RM0433
Reference manual

STM32H743/753 and STM32H750 advanced ARM®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32H743/53xx and STM32H750xB microcontroller memory and peripherals.

The STM32H743/753 and STM32H750 are lines of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics please refer to the corresponding datasheets.

For information on the Arm® Cortex®-M7 with FPU core, please refer to the corresponding Arm *Technical Reference Manuals*.

Related documents

- Arm® Cortex®-M7 Technical Reference Manual, available from www.arm.com.
- Cortex®-M7 programming manual (PM0253).
- STM32H743xx and STM32H753xx datasheets
- STM32H750xB datasheet

Contents

1 Documentation conventions 97

1.1 General information 97

1.2 List of abbreviations for registers 97

1.3 Glossary 98

1.4 Availability of peripherals 98

1.5 Availability of security features 98

2 Memory and bus architecture 99

2.1 System architecture 99

2.1.1 Bus matrices 101

2.1.2 TCM buses 101

2.1.3 Bus-to-bus bridges 101

2.1.4 Inter-domain buses 102

2.1.5 CPU buses 102

2.1.6 Bus master peripherals 103

2.1.7 Clocks to functional blocks 104

2.2 AXI interconnect matrix (AXIM) 104

2.2.1 AXI introduction 104

2.2.2 AXI interconnect main features 104

2.2.3 AXI interconnect functional description 105

2.2.4 AXI interconnect registers 107

2.2.5 AXI interconnect register map 116

2.3 Memory organization 124

2.3.1 Introduction 124

2.3.2 Memory map and register boundary addresses 124

2.4 Embedded SRAM 130

2.5 Flash memory overview 131

2.6 Boot configuration 131

3 Embedded Flash memory (FLASH) 133

3.1 Introduction 133

3.2 FLASH main features 133

3.3 FLASH functional description 134

3.3.1 FLASH block diagram . . . 134
3.3.2 FLASH internal signals . . . 135
3.3.3 FLASH architecture and integration in the system . . . 135
3.3.4 Flash memory architecture and usage . . . 137
3.3.5 FLASH system performance enhancements . . . 140
3.3.6 FLASH data protection schemes . . . 140
3.3.7 Overview of FLASH operations . . . 141
3.3.8 FLASH read operations . . . 142
3.3.9 FLASH program operations . . . 145
3.3.10 FLASH erase operations . . . 149
3.3.11 FLASH parallel operations (STM32H743/753 devices only) . . . 152
3.3.12 Flash memory error protections . . . 152
3.3.13 Flash bank and register swapping (STM32H743/753 devices only) . . . 154
3.3.14 FLASH reset and clocks . . . 157
3.4 FLASH option bytes . . . 157
3.4.1 About option bytes . . . 157
3.4.2 Option byte loading . . . 158
3.4.3 Option byte modification . . . 158
3.4.4 Option bytes overview . . . 161
3.4.5 Description of user and system option bytes . . . 163
3.4.6 Description of data protection option bytes . . . 164
3.4.7 Description of boot address option bytes . . . 165
3.5 FLASH protection mechanisms . . . 165
3.5.1 FLASH configuration protection . . . 165
3.5.2 Write protection . . . 167
3.5.3 Readout protection (RDP) . . . 168
3.5.4 Proprietary code readout protection (PCROP) . . . 172
3.5.5 Secure access mode . . . 173
3.6 FLASH low-power modes . . . 174
3.7 FLASH error management . . . 176
3.7.1 Introduction . . . 176
3.7.2 Write protection error (WRPERR) . . . 176
3.7.3 Programming sequence error (PGSERR) . . . 177
3.7.4 Strobe error (STRBERR) . . . 178
3.7.5 Inconsistency error (INCERR) . . . 178
3.7.6 Operation error (OPERR) . . . 179
3.7.7 Error correction code error (SNECCERR/DBECCERR) . . . 179

3.7.8 Read protection error (RDPERR) . . . 180
3.7.9 Read secure error (RDSERR) . . . 180
3.7.10 Option byte change error (OPTCHANGEERR) . . . 180
3.7.11 Miscellaneous HardFault errors . . . 181
3.8 FLASH interrupts . . . 181
3.9 FLASH registers . . . 184
3.9.1 FLASH access control register (FLASH_ACR) . . . 184
3.9.2 FLASH key register for bank 1 (FLASH_KEYR1) . . . 184
3.9.3 FLASH option key register (FLASH_OPTKEYR) . . . 185
3.9.4 FLASH control register for bank 1 (FLASH_CR1) . . . 185
3.9.5 FLASH status register for bank 1 (FLASH_SR1) . . . 190
3.9.6 FLASH clear control register for bank 1 (FLASH_CCR1) . . . 193
3.9.7 FLASH option control register (FLASH_OPTCR) . . . 194
3.9.8 FLASH option status register (FLASH_OPTSR_CUR) . . . 195
3.9.9 FLASH option status register (FLASH_OPTSR_PRG) . . . 198
3.9.10 FLASH option clear control register (FLASH_OPTCCR) . . . 200
3.9.11 FLASH protection address for bank 1 (FLASH_PRAR_CUR1) . . . 200
3.9.12 FLASH protection address for bank 1 (FLASH_PRAR_PRG1) . . . 201
3.9.13 FLASH secure address for bank 1 (FLASH_SCAR_CUR1) . . . 202
3.9.14 FLASH secure address for bank 1 (FLASH_SCAR_PRG1) . . . 202
3.9.15 FLASH write sector protection for bank 1 (FLASH_WPSN_CUR1R) . . . 203
3.9.16 FLASH write sector protection for bank 1 (FLASH_WPSN_PRG1R) . . . 204
3.9.17 FLASH register boot address FLASH_BOOT_CURR) . . . 204
3.9.18 FLASH register boot address FLASH_BOOT_PRGR) . . . 204
3.9.19 FLASH CRC control register for bank 1 (FLASH_CRCCR1) . . . 205
3.9.20 FLASH CRC start address register for bank 1 (FLASH_CRCSADD1R) . . . 206
3.9.21 FLASH CRC end address register for bank 1 (FLASH_CRCEADD1R) . . . 207
3.9.22 FLASH CRC data register (FLASH_CRCDATAR) . . . 207
3.9.23 FLASH ECC fail address for bank 1 (FLASH_ECC_FA1R) . . . 208
3.9.24 FLASH key register for bank 2 (FLASH_KEYR2) . . . 208
3.9.25 FLASH control register for bank 2 (FLASH_CR2) . . . 209
3.9.26 FLASH status register for bank 2 (FLASH_SR2) . . . 213
3.9.27 FLASH clear control register for bank 2 (FLASH_CCR2) . . . 216
3.9.28 FLASH protection address for bank 2 (FLASH_PRAR_CUR2) . . . 217

3.9.29 FLASH protection address for bank 2 (FLASH_PRAR_PRG2) 218
3.9.30 FLASH secure address for bank 2 (FLASH_SCAR_CUR2) 218
3.9.31 FLASH secure address for bank 2 (FLASH_SCAR_PRG2) 219
3.9.32 FLASH write sector protection for bank 2 (FLASH_WPSN_CUR2R) 220
3.9.33 FLASH write sector protection for bank 2 (FLASH_WPSN_PRG2R) 220
3.9.34 FLASH CRC control register for bank 2 (FLASH_CRCCR2) 221
3.9.35 FLASH CRC start address register for bank 2 (FLASH_CRCSADD2R) 222
3.9.36 FLASH CRC end address register for bank 2 (FLASH_CRCEADD2R) 223
3.9.37 FLASH ECC fail address for bank 2 (FLASH_ECC_FA2R) 223
3.10 FLASH register map and reset values 224

4 Secure memory management 229
4.1 Introduction 229
4.2 Glossary 229
4.3 Flash protections 230
4.4 Secure access mode 231
4.4.1 Associated features 232
4.4.2 Boot state machine 232
4.4.3 Secure access mode configuration 233
4.5 Root secure services (RSS) 234
4.5.1 Calling root secure services 234
4.5.2 Root secure services description 235
4.6 Secure user software 235
4.6.1 Access rules 236
4.6.2 Setting secure user memory areas 236
4.6.3 Removing secure user memory areas 236
4.6.4 Selecting secure user software 237
4.7 Summary of Flash protection mechanisms 238

5 Power control (PWR) 239
5.1 Introduction 239
5.2 PWR main features 239
5.3 PWR block diagram 240

5.3.1 PWR pins and internal signals . . . 241
5.4 Power supplies . . . 243
5.4.1 System supply startup . . . 246
5.4.2 Core domain . . . 247
5.4.3 PWR external supply . . . 248
5.4.4 Backup domain . . . 249
5.4.5 VBAT battery charging . . . 251
5.4.6 Analog supply . . . 251
5.4.7 USB regulator . . . 252
5.5 Power supply supervision . . . 252
5.5.1 Power-on reset (POR)/power-down reset (PDR) . . . 253
5.5.2 Brownout reset (BOR) . . . 253
5.5.3 Programmable voltage detector (PVD) . . . 254
5.5.4 Analog voltage detector (AVD) . . . 256
5.5.5 Battery voltage thresholds . . . 257
5.5.6 Temperature thresholds . . . 258
5.6 Power management . . . 259
5.6.1 Operating modes . . . 260
5.6.2 Voltage scaling . . . 263
5.6.3 Power control modes . . . 264
5.6.4 Power management examples . . . 267
5.7 Low-power modes . . . 273
5.7.1 Slowing down system clocks . . . 273
5.7.2 Controlling peripheral clocks . . . 273
5.7.3 Entering low-power modes . . . 273
5.7.4 Exiting from low-power modes . . . 274
5.7.5 CSleep mode . . . 275
5.7.6 CStop mode . . . 276
5.7.7 DStop mode . . . 277
5.7.8 Stop mode . . . 278
5.7.9 DStandby mode . . . 280
5.7.10 Standby mode . . . 282
5.8 PWR register description . . . 284
5.8.1 PWR control register 1 (PWR_CR1) . . . 284
5.8.2 PWR control status register 1 (PWR_CSR1) . . . 285
5.8.3 PWR control register 2 (PWR_CR2) . . . 286

5.8.4 PWR control register 3 (PWR_CR3) 287
5.8.5 PWR CPU control register (PWR_CPUCR) 289
5.8.6 PWR D3 domain control register (PWR_D3CR) 290
5.8.7 PWR wakeup clear register (PWR_WKUPCR) 291
5.8.8 PWR wakeup flag register (PWR_WKUPFR) 291
5.8.9 PWR wakeup enable and polarity register (PWR_WKUPEPR) 292
5.8.10 PWR register map 293

6 Low-power D3 domain 294
6.1 Introduction 294
6.2 EXTI, RCC and PWR interconnections 294
6.2.1 Interrupts and wakeup 296
6.2.2 Block interactions 296
6.2.3 Role of D3 domain DMAMUX2 297
6.3 Low-power application example based on LPUART1 transmission 298
6.3.1 Memory retention 298
6.3.2 Memory-to-peripheral transfer using LPUART1 interface 298
6.3.3 Overall description of the low-power application example based on LPUART1 transmission 303
6.3.4 Alternate implementations 304
6.4 Other low-power applications 305

7 Reset and Clock Control (RCC) 306
7.1 RCC main features 306
7.2 RCC block diagram 307
7.3 RCC pins and internal signals 307
7.4 RCC reset block functional description 309
7.4.1 Power-on/off reset 309
7.4.2 System reset 310
7.4.3 Local resets 311
7.4.4 Reset source identification 313
7.4.5 Low-power mode security reset (lpwr_rst) 314
7.4.6 Backup domain reset 314
7.4.7 Power-on and wakeup sequences 314
7.5 RCC clock block functional description 317
7.5.1 Clock naming convention 319

7.5.2 Oscillators description . . . 319
7.5.3 Clock Security System (CSS) . . . 324
7.5.4 Clock output generation (MCO1/MCO2) . . . 325
7.5.5 PLL description . . . 326
7.5.6 System clock (sys_ck) . . . 330
7.5.7 Handling clock generators in Stop and Standby mode . . . 332
7.5.8 Kernel clock selection . . . 334
7.5.9 General clock concept overview . . . 346
7.5.10 Peripheral allocation . . . 350
7.5.11 Peripheral clock gating control . . . 352
7.5.12 CPU and bus matrix clock gating control . . . 357
7.6 RCC Interrupts . . . 359
7.7 RCC register description . . . 360
7.7.1 Register mapping overview . . . 360
7.7.2 RCC Source Control Register (RCC_CR) . . . 361
7.7.3 RCC Internal Clock Source Calibration Register (RCC_ICSCR) . . . 365
7.7.4 RCC Clock Recovery RC Register (RCC_CRRCR) . . . 366
7.7.5 RCC Clock Configuration Register (RCC_CFGR) . . . 367
7.7.6 RCC Domain 1 Clock Configuration Register (RCC_D1CFGR) . . . 370
7.7.7 RCC Domain 2 Clock Configuration Register (RCC_D2CFGR) . . . 372
7.7.8 RCC Domain 3 Clock Configuration Register (RCC_D3CFGR) . . . 373
7.7.9 RCC PLLs Clock Source Selection Register (RCC_PLLCKSELR) . . . 374
7.7.10 RCC PLLs Configuration Register (RCC_PLLCFGR) . . . 376
7.7.11 RCC PLL1 Dividers Configuration Register (RCC_PLL1DIVR) . . . 379
7.7.12 RCC PLL1 Fractional Divider Register (RCC_PLL1FRACR) . . . 381
7.7.13 RCC PLL2 Dividers Configuration Register (RCC_PLL2DIVR) . . . 382
7.7.14 RCC PLL2 Fractional Divider Register (RCC_PLL2FRACR) . . . 384
7.7.15 RCC PLL3 Dividers Configuration Register (RCC_PLL3DIVR) . . . 385
7.7.16 RCC PLL3 Fractional Divider Register (RCC_PLL3FRACR) . . . 387
7.7.17 RCC Domain 1 Kernel Clock Configuration Register (RCC_D1CCIPR) . . . 388
7.7.18 RCC Domain 2 Kernel Clock Configuration Register (RCC_D2CCIP1R) . . . 389
7.7.19 RCC Domain 2 Kernel Clock Configuration Register (RCC_D2CCIP2R) . . . 392
7.7.20 RCC Domain 3 Kernel Clock Configuration Register (RCC_D3CCIPR) . . . 394
7.7.21 RCC Clock Source Interrupt Enable Register (RCC_CIER) . . . 397

7.7.22 RCC Clock Source Interrupt Flag Register (RCC_CIFR) 399
7.7.23 RCC Clock Source Interrupt Clear Register (RCC_CICR) 401
7.7.24 RCC Backup Domain Control Register (RCC_BDCR) 403
7.7.25 RCC Clock Control and Status Register (RCC_CSR) 405
7.7.26 RCC AHB3 Reset Register (RCC_AHB3RSTR) 406
7.7.27 RCC AHB1 Peripheral Reset Register(RCC_AHB1RSTR) 408
7.7.28 RCC AHB2 Peripheral Reset Register (RCC_AHB2RSTR) 410
7.7.29 RCC AHB4 Peripheral Reset Register (RCC_AHB4RSTR) 411
7.7.30 RCC APB3 Peripheral Reset Register (RCC_APB3RSTR) 413
7.7.31 RCC APB1 Peripheral Reset Register (RCC_APB1LRSTR) 414
7.7.32 RCC APB1 Peripheral Reset Register (RCC_APB1HRSTR) 417
7.7.33 RCC APB2 Peripheral Reset Register (RCC_APB2RSTR) 418
7.7.34 RCC APB4 Peripheral Reset Register (RCC_APB4RSTR) 420
7.7.35 RCC Global Control Register (RCC_GCR) 422
7.7.36 RCC D3 Autonomous mode Register (RCC_D3AMR) 423
7.7.37 RCC Reset Status Register (RCC_RSR) 426
7.7.38 RCC AHB3 Clock Register (RCC_AHB3ENR) 428
7.7.39 RCC AHB1 Clock Register (RCC_AHB1ENR) 430
7.7.40 RCC AHB2 Clock Register (RCC_AHB2ENR) 432
7.7.41 RCC AHB4 Clock Register (RCC_AHB4ENR) 434
7.7.42 RCC APB3 Clock Register (RCC_APB3ENR) 437
7.7.43 RCC APB1 Clock Register (RCC_APB1LENR) 438
7.7.44 RCC APB1 Clock Register (RCC_APB1HENR) 442
7.7.45 RCC APB2 Clock Register (RCC_APB2ENR) 444
7.7.46 RCC APB4 Clock Register (RCC_APB4ENR) 447
7.7.47 RCC AHB3 Sleep Clock Register (RCC_AHB3LPENR) 450
7.7.48 RCC AHB1 Sleep Clock Register (RCC_AHB1LPENR) 452
7.7.49 RCC AHB2 Sleep Clock Register (RCC_AHB2LPENR) 454
7.7.50 RCC AHB4 Sleep Clock Register (RCC_AHB4LPENR) 456
7.7.51 RCC APB3 Sleep Clock Register (RCC_APB3LPENR) 459
7.7.52 RCC APB1 Low Sleep Clock Register (RCC_APB1LLPENR) 460
7.7.53 RCC APB1 High Sleep Clock Register (RCC_APB1HLPENR) 464
7.7.54 RCC APB2 Sleep Clock Register (RCC_APB2LPENR) 466
7.7.55 RCC APB4 Sleep Clock Register (RCC_APB4LPENR) 469
7.8 RCC register map 472

8 Clock recovery system (CRS) 482

8.1 Introduction . . . 482
8.2 CRS main features . . . 482
8.3 CRS functional description . . . 483
8.3.1 CRS block diagram . . . 483
8.4 CRS internal signals . . . 483
8.4.1 Synchronization input . . . 484
8.4.2 Frequency error measurement . . . 484
8.4.3 Frequency error evaluation and automatic trimming . . . 486
8.4.4 CRS initialization and configuration . . . 486
8.5 CRS low-power modes . . . 487
8.6 CRS interrupts . . . 487
8.7 CRS registers . . . 488
8.7.1 CRS control register (CRS_CR) . . . 488
8.7.2 CRS configuration register (CRS_CFGR) . . . 489
8.7.3 CRS interrupt and status register (CRS_ISR) . . . 490
8.7.4 CRS interrupt flag clear register (CRS_ICR) . . . 492
8.7.5 CRS register map . . . 493

9 Hardware semaphore (HSEM) . . . 494
9.1 Hardware semaphore introduction . . . 494
9.2 Hardware semaphore main features . . . 494
9.3 HSEM functional description . . . 495
9.3.1 HSEM block diagram . . . 495
9.3.2 HSEM internal signals . . . 495
9.3.3 HSEM lock procedures . . . 496
9.3.4 HSEM Write/Read/ReadLock register address . . . 497
9.3.5 HSEM Clear procedures . . . 497
9.3.6 HSEM MasterID semaphore clear . . . 498
9.3.7 HSEM interrupts . . . 498
9.3.8 AHB bus master ID verification . . . 500
9.4 HSEM registers . . . 502
9.4.1 HSEM register semaphore x (HSEM_Rx) . . . 502
9.4.2 HSEM read lock register semaphore x (HSEM_RLRx) . . . 503
9.4.3 HSEM interrupt enable register (HSEM_CnIER) . . . 504
9.4.4 HSEM interrupt clear register (HSEM_CnICR) . . . 504
9.4.5 HSEM interrupt status register (HSEM_CnISR) . . . 504

9.4.6 HSEM masked interrupt status register (HSEM_CnMISR) 505
9.4.7 HSEM clear register (HSEM_CR) 505
9.4.8 HSEM interrupt clear register (HSEM_KEYR) 506
9.4.9 HSEM register map 507

10 General-purpose I/Os (GPIO) 508
10.1 Introduction 508
10.2 GPIO main features 508
10.3 GPIO functional description 508
10.3.1 General-purpose I/O (GPIO) 511
10.3.2 I/O pin alternate function multiplexer and mapping 511
10.3.3 I/O port control registers 512
10.3.4 I/O port data registers 512
10.3.5 I/O data bitwise handling 512
10.3.6 GPIO locking mechanism 513
10.3.7 I/O alternate function input/output 513
10.3.8 External interrupt/wakeup lines 513
10.3.9 Input configuration 514
10.3.10 Output configuration 514
10.3.11 I/O compensation cell 515
10.3.12 Alternate function configuration 515
10.3.13 Analog configuration 516
10.3.14 Using the HSE or LSE oscillator pins as GPIOs 517
10.3.15 Using the GPIO pins in the backup supply domain 517
10.4 GPIO registers 517
10.4.1 GPIO port mode register (GPIOx_MODER) (x =A to K) 518
10.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to K) 518
10.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to K) 519
10.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to K) 519
10.4.5 GPIO port input data register (GPIOx_IDR) (x = A to K) 520
10.4.6 GPIO port output data register (GPIOx_ODR) (x = A to K) 520
10.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to K) 520
10.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to K) 521
10.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to K) 522

10.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to J) . . . 523
10.4.11 GPIO register map . . . 524

11 System configuration controller (SYSCFG) . . . 526

11.1 Introduction . . . 526
11.2 SYSCFG main features . . . 526
11.3 SYSCFG registers . . . 526
11.3.1 SYSCFG peripheral mode configuration register (SYSCFG_PMCR) . 526
11.3.2 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1) . . . 528
11.3.3 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2) . . . 529
11.3.4 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3) . . . 530
11.3.5 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4) . . . 531
11.3.6 SYSCFG configuration register (SYSCFG_CFGR) . . . 532
11.3.7 SYSCFG compensation cell control/status register (SYSCFG_CCCSR) . . . 535
11.3.8 SYSCFG compensation cell value register (SYSCFG_CCVR) . . . 536
11.3.9 SYSCFG compensation cell code register (SYSCFG_CCCR) . . . 536
11.3.10 SYSCFG package register (SYSCFG_PKGR) . . . 537
11.3.11 SYSCFG user register 0 (SYSCFG_UR0) . . . 538
11.3.12 SYSCFG user register 2 (SYSCFG_UR2) . . . 538
11.3.13 SYSCFG user register 3 (SYSCFG_UR3) . . . 539
11.3.14 SYSCFG user register 4 (SYSCFG_UR4) . . . 539
11.3.15 SYSCFG user register 5 (SYSCFG_UR5) . . . 540
11.3.16 SYSCFG user register 6 (SYSCFG_UR6) . . . 540
11.3.17 SYSCFG user register 7 (SYSCFG_UR7) . . . 541
11.3.18 SYSCFG user register 8 (SYSCFG_UR8) . . . 541
11.3.19 SYSCFG user register 9 (SYSCFG_UR9) . . . 542
11.3.20 SYSCFG user register 10 (SYSCFG_UR10) . . . 542
11.3.21 SYSCFG user register 11 (SYSCFG_UR11) . . . 543
11.3.22 SYSCFG user register 12 (SYSCFG_UR12) . . . 543
11.3.23 SYSCFG user register 13 (SYSCFG_UR13) . . . 544
11.3.24 SYSCFG user register 14 (SYSCFG_UR14) . . . 545
11.3.25 SYSCFG user register 15 (SYSCFG_UR15) . . . 546
11.3.26 SYSCFG user register 16 (SYSCFG_UR16) . . . 547

11.3.27 SYSCFG user register 17 (SYSCFG_UR17) 547
11.3.28 SYSCFG register maps 548

12 Block interconnect 551
12.1 Peripheral interconnect 551
12.1.1 Introduction 551
12.1.2 Connection overview 551
12.2 Wakeup from low power modes 570
12.3 DMA 575
12.3.1 MDMA (D1 domain) 576
12.3.2 DMAMUX1, DMA1 and DMA2 (D2 domain) 578
12.3.3 DMAMUX2, BDMA (D3 domain) 583

13 MDMA controller (MDMA) 586
13.1 MDMA introduction 586
13.2 MDMA main features 586
13.3 MDMA functional description 588
13.3.1 MDMA block diagram 588
13.3.2 MDMA internal signals 588
13.3.3 MDMA overview 589
13.3.4 MDMA channel 590
13.3.5 Source, destination and transfer modes 590
13.3.6 Pointer update 590
13.3.7 MDMA buffer transfer 591
13.3.8 Request arbitration 592
13.3.9 FIFO 592
13.3.10 Block transfer 593
13.3.11 Block repeat mode 593
13.3.12 Linked list mode 593
13.3.13 MDMA transfer completion 593
13.3.14 MDMA transfer suspension 594
13.3.15 Error management 594
13.4 MDMA interrupts 594
13.5 MDMA registers 595
13.5.1 MDMA global interrupt/status register (MDMA_GISR0) 595
13.5.2 MDMA channel x interrupt/status register (MDMA_CxISR) 596

13.5.3 MDMA channel x interrupt flag clear register (MDMA_CxIFCR) 597
13.5.4 MDMA channel x error status register (MDMA_CxESR) 598
13.5.5 MDMA channel x control register (MDMA_CxCR) 600
13.5.6 MDMA channel x transfer configuration register (MDMA_CxTCR) . . . 602
13.5.7 MDMA channel x block number of data register (MDMA_CxBNDTR) . 605
13.5.8 MDMA channel x source address register (MDMA_CxSAR) 607
13.5.9 MDMA channel x destination address register (MDMA_CxDAR) 607
13.5.10 MDMA channel x block repeat address update register (MDMA_CxBRUR) 609
13.5.11 MDMA channel x link address register (MDMA_CxLAR) 610
13.5.12 MDMA channel x trigger and bus selection register (MDMA_CxTBR) 611
13.5.13 MDMA channel x mask address register (MDMA_CxMAR) 612
13.5.14 MDMA channel x mask data register (MDMA_CxMDR) 612
13.5.15 MDMA register map 613

14 Direct memory access controller (DMA) 615

14.1 DMA introduction 615
14.2 DMA main features 615
14.3 DMA functional description 617
14.3.1 DMA block diagram 617
14.3.2 DMA internal signals 617
14.3.3 DMA overview 618
14.3.4 DMA transactions 618
14.3.5 DMA request mapping 618
14.3.6 Arbiter 619
14.3.7 DMA streams 619
14.3.8 Source, destination and transfer modes 619
14.3.9 Pointer incrementation 622
14.3.10 Circular mode 623
14.3.11 Double-buffer mode 623
14.3.12 Programmable data width, packing/unpacking, endianness 624
14.3.13 Single and burst transfers 626
14.3.14 FIFO 627
14.3.15 DMA transfer completion 629
14.3.16 DMA transfer suspension 630
14.3.17 Flow controller 631
14.3.18 Summary of the possible DMA configurations 632

14.3.19 Stream configuration procedure . . . 632
14.3.20 Error management . . . 633
14.4 DMA interrupts . . . 634
14.5 DMA registers . . . 635
14.5.1 DMA low interrupt status register (DMA_LISR) . . . 635
14.5.2 DMA high interrupt status register (DMA_HISR) . . . 636
14.5.3 DMA low interrupt flag clear register (DMA_LIFCR) . . . 637
14.5.4 DMA high interrupt flag clear register (DMA_HIFCR) . . . 637
14.5.5 DMA stream x configuration register (DMA_SxCR) . . . 638
14.5.6 DMA stream x number of data register (DMA_SxNDTR) . . . 641
14.5.7 DMA stream x peripheral address register (DMA_SxPAR) . . . 642
14.5.8 DMA stream x memory 0 address register (DMA_SxM0AR) . . . 642
14.5.9 DMA stream x memory 1 address register (DMA_SxM1AR) . . . 642
14.5.10 DMA stream x FIFO control register (DMA_SxFCR) . . . 643
14.5.11 DMA register map . . . 645

15 Basic direct memory access controller (BDMA) . . . 649
15.1 Introduction . . . 649
15.2 BDMA main features . . . 649
15.3 BDMA implementation . . . 649
15.3.1 BDMA . . . 649
15.3.2 BDMA request mapping . . . 650
15.4 BDMA functional description . . . 650
15.4.1 BDMA block diagram . . . 650
15.4.2 BDMA pins and internal signals . . . 651
15.4.3 BDMA transfers . . . 651
15.4.4 BDMA arbitration . . . 652
15.4.5 BDMA channels . . . 652
15.4.6 BDMA data width, alignment and endianness . . . 656
15.4.7 BDMA error management . . . 657
15.5 BDMA interrupts . . . 658
15.6 BDMA registers . . . 658
15.6.1 BDMA interrupt status register (BDMA_ISR) . . . 658
15.6.2 BDMA interrupt flag clear register (BDMA_IFCR) . . . 661
15.6.3 BDMA channel x configuration register (BDMA_CCRx) . . . 662

15.6.4 BDMA channel x number of data to transfer register (BDMA_CNDTRx) . . . 666
15.6.5 BDMA channel x peripheral address register (BDMA_CPARx) . . . 666
15.6.6 BDMA channel x memory 0 address register (BDMA_CM0ARx) . . . 667
15.6.7 BDMA channel x memory 1 address register (BDMA_CM1ARx) . . . 668
15.6.8 BDMA register map and reset values . . . 668

16 DMA request multiplexer (DMAMUX) . . . 671
16.1 Introduction . . . 671
16.2 DMAMUX main features . . . 671
16.3 DMAMUX implementation . . . 672
16.3.1 DMAMUX1 and DMAMUX2 instantiation . . . 672
16.3.2 DMAMUX1 mapping . . . 672
16.3.3 DMAMUX2 mapping . . . 674
16.4 DMAMUX functional description . . . 676
16.4.1 DMAMUX block diagram . . . 676
16.4.2 DMAMUX signals . . . 677
16.4.3 DMAMUX channels . . . 677
16.4.4 DMAMUX request line multiplexer . . . 677
16.4.5 DMAMUX request generator . . . 680
16.5 DMAMUX interrupts . . . 681
16.6 DMAMUX registers . . . 682
16.6.1 DMAMUX1 request line multiplexer channel x configuration register (DMAMUX1_CxCR) . . . 682
16.6.2 DMAMUX2 request line multiplexer channel x configuration register (DMAMUX2_CxCR) . . . 683
16.6.3 DMAMUX1 request line multiplexer interrupt channel status register (DMAMUX1_CSR) . . . 684
16.6.4 DMAMUX2 request line multiplexer interrupt channel status register (DMAMUX2_CSR) . . . 684
16.6.5 DMAMUX1 request line multiplexer interrupt clear flag register (DMAMUX1_CFR) . . . 685
16.6.6 DMAMUX2 request line multiplexer interrupt clear flag register (DMAMUX2_CFR) . . . 685
16.6.7 DMAMUX1 request generator channel x configuration register (DMAMUX1_RGxCR) . . . 686
16.6.8 DMAMUX2 request generator channel x configuration register (DMAMUX2_RGxCR) . . . 686

16.6.9 DMAMUX1 request generator interrupt status register (DMAMUX1_RGSR) . . . 688
16.6.10 DMAMUX2 request generator interrupt status register (DMAMUX2_RGSR) . . . 688
16.6.11 DMAMUX1 request generator interrupt clear flag register (DMAMUX1_RGCFR) . . . 689
16.6.12 DMAMUX2 request generator interrupt clear flag register (DMAMUX2_RGCFR) . . . 689
16.6.13 DMAMUX register map . . . 690

17 Chrom-Art Accelerator™ controller (DMA2D) . . . 692
17.1 DMA2D introduction . . . 692
17.2 DMA2D main features . . . 692
17.3 DMA2D functional description . . . 693
17.3.1 General description . . . 693
17.4 DMA2D pins and internal signals . . . 694
17.4.1 DMA2D control . . . 695
17.4.2 DMA2D foreground and background FIFOs . . . 695
17.4.3 DMA2D foreground and background pixel format converter (PFC) . . . 695
17.4.4 DMA2D foreground and background CLUT interface . . . 698
17.4.5 DMA2D blender . . . 699
17.4.6 DMA2D output PFC . . . 699
17.4.7 DMA2D output FIFO . . . 699
17.4.8 DMA2D output FIFO byte reordering . . . 700
17.4.9 DMA2D AXI master port timer . . . 702
17.4.10 DMA2D transactions . . . 702
17.4.11 DMA2D configuration . . . 702
17.4.12 YCbCr support . . . 706
17.4.13 DMA2D transfer control (start, suspend, abort and completion) . . . 707
17.4.14 Watermark . . . 707
17.4.15 Error management . . . 707
17.4.16 AXI dead time . . . 707
17.5 DMA2D interrupts . . . 708
17.6 DMA2D registers . . . 709
17.6.1 DMA2D control register (DMA2D_CR) . . . 709
17.6.2 DMA2D Interrupt Status Register (DMA2D_ISR) . . . 711
17.6.3 DMA2D interrupt flag clear register (DMA2D_IFCR) . . . 712
17.6.4 DMA2D foreground memory address register (DMA2D_FGMAR) . . . 713

17.6.5 DMA2D foreground offset register (DMA2D_FGOR) 713
17.6.6 DMA2D background memory address register (DMA2D_BGMAR) . . 714
17.6.7 DMA2D background offset register (DMA2D_BGOR) 714
17.6.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR) 715
17.6.9 DMA2D foreground color register (DMA2D_FGCOLR) 717
17.6.10 DMA2D background PFC control register (DMA2D_BGPFCCR) 718
17.6.11 DMA2D background color register (DMA2D_BGCOLR) 720
17.6.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR) 720
17.6.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR) 721
17.6.14 DMA2D output PFC control register (DMA2D_OPFCCR) 721
17.6.15 DMA2D output color register (DMA2D_OCOLR) 722
17.6.16 DMA2D output memory address register (DMA2D_OMAR) 724
17.6.17 DMA2D output offset register (DMA2D_OOR) 724
17.6.18 DMA2D number of line register (DMA2D_NLR) 725
17.6.19 DMA2D line watermark register (DMA2D_LWR) 725
17.6.20 DMA2D AXI master timer configuration register (DMA2D_AMTCR) . . 726
17.6.21 DMA2D register map 727

18 Nested Vectored Interrupt Controllers 729

18.1 NVIC features 729
18.1.1 SysTick calibration value register 729
18.1.2 Interrupt and exception vectors 730

19 Extended interrupt and event controller (EXTI) 738

19.1 EXTI main features 738
19.2 EXTI block diagram 738
19.2.1 EXTI connections between peripherals, CPU, and D3 domain 739
19.3 EXTI functional description 740
19.3.1 EXTI Configurable event input CPU wakeup 741
19.3.2 EXTI configurable event input Any wakeup 742
19.3.3 EXTI direct event input CPU wakeup 744
19.3.4 EXTI direct event input Any wakeup 745
19.3.5 EXTI D3 pending request clear selection 746
19.4 EXTI event input mapping 746
19.5 EXTI functional behavior 749

19.5.1 EXTI CPU interrupt procedure 750
19.5.2 EXTI CPU event procedure 750
19.5.3 EXTI CPU wakeup procedure 751
19.5.4 EXTI D3 domain wakeup for autonomous Run mode procedure 751
19.5.5 EXTI software interrupt/event trigger procedure 751
19.6 EXTI registers 752
19.6.1 EXTI rising trigger selection register (EXTI_RTSR1) 752
19.6.2 EXTI falling trigger selection register (EXTI_FTSR1) 752
19.6.3 EXTI software interrupt event register (EXTI_SWIER1) 753
19.6.4 EXTI D3 pending mask register (EXTI_D3PMR1) 753
19.6.5 EXTI D3 pending clear selection register low (EXTI_D3PCR1L) 754
19.6.6 EXTI D3 pending clear selection register high (EXTI_D3PCR1H) . . . 754
19.6.7 EXTI rising trigger selection register (EXTI_RTSR2) 755
19.6.8 EXTI falling trigger selection register (EXTI_FTSR2) 756
19.6.9 EXTI software interrupt event register (EXTI_SWIER2) 756
19.6.10 EXTI D3 pending mask register (EXTI_D3PMR2) 757
19.6.11 EXTI D3 pending clear selection register low (EXTI_D3PCR2L) 758
19.6.12 EXTI D3 pending clear selection register high (EXTI_D3PCR2H) . . . 758
19.6.13 EXTI rising trigger selection register (EXTI_RTSR3) 759
19.6.14 EXTI falling trigger selection register (EXTI_FTSR3) 759
19.6.15 EXTI software interrupt event register (EXTI_SWIER3) 760
19.6.16 EXTI D3 pending mask register (EXTI_D3PMR3) 760
19.6.17 EXTI D3 pending clear selection register low (EXTI_D3PCR3L) 761
19.6.18 EXTI D3 pending clear selection register high (EXTI_D3PCR3H) . . . 761
19.6.19 EXTI interrupt mask register (EXTI_CPUIMR1) 762
19.6.20 EXTI event mask register (EXTI_CPUEMR1) 762
19.6.21 EXTI pending register (EXTI_CPUPR1) 763
19.6.22 EXTI interrupt mask register (EXTI_CPUIMR2) 763
19.6.23 EXTI event mask register (EXTI_CPUEMR2) 764
19.6.24 EXTI pending register (EXTI_CPUPR2) 764
19.6.25 EXTI interrupt mask register (EXTI_CPUIMR3) 765
19.6.26 EXTI event mask register (EXTI_CPUEMR3) 766
19.6.27 EXTI pending register (EXTI_CPUPR3) 766
19.6.28 EXTI register map 767

20 Cyclic redundancy check calculation unit (CRC) 770

20.1 Introduction 770

20.2 CRC main features . . . 770
20.3 CRC functional description . . . 771
20.3.1 CRC block diagram . . . 771
20.3.2 CRC internal signals . . . 771
20.3.3 CRC operation . . . 771
20.4 CRC registers . . . 773
20.4.1 Data register (CRC_DR) . . . 773
20.4.2 Independent data register (CRC_IDR) . . . 773
20.4.3 Control register (CRC_CR) . . . 774
20.4.4 Initial CRC value (CRC_INIT) . . . 774
20.4.5 CRC polynomial (CRC_POL) . . . 775
20.4.6 CRC register map . . . 775

21 Flexible memory controller (FMC) . . . 776
21.1 FMC main features . . . 776
21.2 FMC block diagram . . . 777
21.3 FMC internal signals . . . 779
21.4 AHB interface . . . 779
21.5 AXI interface . . . 779
21.5.1 Supported memories and transactions . . . 780
21.6 External device address mapping . . . 781
21.6.1 NOR/PSRAM address mapping . . . 783
21.6.2 NAND Flash memory address mapping . . . 783
21.6.3 SDRAM address mapping . . . 784
21.7 NOR Flash/PSRAM controller . . . 787
21.7.1 External memory interface signals . . . 789
21.7.2 Supported memories and transactions . . . 791
21.7.3 General timing rules . . . 792
21.7.4 NOR Flash/PSRAM controller asynchronous transactions . . . 792
21.7.5 Synchronous transactions . . . 812
21.7.6 NOR/PSRAM controller registers . . . 818
21.8 NAND Flash controller . . . 827
21.8.1 External memory interface signals . . . 827
21.8.2 NAND Flash supported memories and transactions . . . 828
21.8.3 Timing diagrams for NAND Flash memories . . . 829
21.8.4 NAND Flash operations . . . 830

21.8.5 NAND Flash prewait feature . . . 831
21.8.6 Computation of the error correction code (ECC) in NAND Flash memory . . . 832
21.8.7 NAND Flash controller registers . . . 833
21.9 SDRAM controller . . . 839
21.9.1 SDRAM controller main features . . . 839
21.9.2 SDRAM External memory interface signals . . . 839
21.9.3 SDRAM controller functional description . . . 840
21.9.4 Low-power modes . . . 847
21.9.5 SDRAM controller registers . . . 850
21.10 FMC register map . . . 857

22 Quad-SPI interface (QUADSPI) . . . 859
22.1 Introduction . . . 859
22.2 QUADSPI main features . . . 859
22.3 QUADSPI functional description . . . 860
22.3.1 QUADSPI block diagram . . . 860
22.3.2 QUADSPI pins and internal signals . . . 861
22.3.3 QUADSPI command sequence . . . 861
22.3.4 QUADSPI signal interface protocol modes . . . 864
22.3.5 QUADSPI indirect mode . . . 866
22.3.6 QUADSPI status flag polling mode . . . 868
22.3.7 QUADSPI memory-mapped mode . . . 868
22.3.8 QUADSPI Free running clock mode . . . 869
22.3.9 QUADSPI Flash memory configuration . . . 869
22.3.10 QUADSPI delayed data sampling . . . 869
22.3.11 QUADSPI configuration . . . 870
22.3.12 QUADSPI usage . . . 870
22.3.13 Sending the instruction only once . . . 872
22.3.14 QUADSPI error management . . . 873
22.3.15 QUADSPI busy bit and abort functionality . . . 873
22.3.16 nCS behavior . . . 873
22.4 QUADSPI interrupts . . . 875
22.5 QUADSPI registers . . . 876
22.5.1 QUADSPI control register (QUADSPI_CR) . . . 876
22.5.2 QUADSPI device configuration register (QUADSPI_DCR) . . . 879
22.5.3 QUADSPI status register (QUADSPI_SR) . . . 880

22.5.4 QUADSPI flag clear register (QUADSPI_FCR) 881
22.5.5 QUADSPI data length register (QUADSPI_DLR) 881
22.5.6 QUADSPI communication configuration register (QUADSPI_CCR) . . 882
22.5.7 QUADSPI address register (QUADSPI_AR) 884
22.5.8 QUADSPI alternate bytes registers (QUADSPI_ABR) 885
22.5.9 QUADSPI data register (QUADSPI_DR) 885
22.5.10 QUADSPI polling status mask register (QUADSPI _PSMKR) 886
22.5.11 QUADSPI polling status match register (QUADSPI _PSMAR) 886
22.5.12 QUADSPI polling interval register (QUADSPI _PIR) 887
22.5.13 QUADSPI low-power timeout register (QUADSPI_LPTR) 887
22.5.14 QUADSPI register map 888

23 Delay block (DLYB) 889

23.1 Introduction 889
23.2 DLYB main features 889
23.3 DLYB functional description 889
23.3.1 DLYB diagram 889
23.3.2 DLYB pins and internal signals 890
23.3.3 General description 890
23.3.4 Delay line length configuration procedure 891
23.3.5 Output clock phase configuration procedure 891
23.4 DLYB registers 892
23.4.1 DLYB control register (DLYB_CR) 892
23.4.2 DLYB configuration register (DLYB_CFGR) 892
23.4.3 DLYB register map 893

24 Analog-to-digital converters (ADC) 894

24.1 Introduction 894
24.2 ADC main features 895
24.3 ADC functional description 897
24.3.1 ADC block diagram 897
24.3.2 ADC pins and internal signals 898
24.3.3 Clocks 899
24.3.4 ADC1/2/3 connectivity 901
24.3.5 Slave AHB interface 904
24.3.6 ADC deep-power-down mode (DEEPPWD) & ADC voltage regulator (ADVREGEN) 904

24.3.7 Single-ended and differential input channels . . . 905
24.3.8 Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT) . . . 905
24.3.9 ADC on-off control (ADEN, ADDIS, ADRDY) . . . 911
24.3.10 Constraints when writing the ADC control bits . . . 912
24.3.11 Channel selection (SQRx, JSQRx) . . . 912
24.3.12 Channel preselection register (ADC_PCSEL) . . . 913
24.3.13 Channel-wise programmable sampling time (SMPR1, SMPR2) . . . 913
24.3.14 Single conversion mode (CONT=0) . . . 914
24.3.15 Continuous conversion mode (CONT=1) . . . 915
24.3.16 Starting conversions (ADSTART, JADSTART) . . . 916
24.3.17 Timing . . . 917
24.3.18 Stopping an ongoing conversion (ADSTP, JADSTP) . . . 917
24.3.19 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) . . . 919
24.3.20 Injected channel management . . . 922
24.3.21 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . 924
24.3.22 Queue of context for injected conversions . . . 925
24.3.23 Programmable resolution (RES) - fast conversion mode . . . 934
24.3.24 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . . 934
24.3.25 End of conversion sequence (EOS, JEOS) . . . 934
24.3.26 Timing diagrams example (single/continuous modes, hardware/software triggers) . . . 935
24.3.27 Data management . . . 936
24.3.28 Managing conversions using the DFSDM . . . 943
24.3.29 Dynamic low-power features . . . 943
24.3.30 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy) . . 948
24.3.31 Oversampler . . . 951
24.3.32 Dual ADC modes . . . 957
24.3.33 Temperature sensor . . . 972
24.3.34 VBAT supply monitoring . . . 973
24.3.35 Monitoring the internal voltage reference . . . 974
24.4 ADC interrupts . . . 976
24.5 ADC registers (for each ADC) . . . 977
24.5.1 ADC interrupt and status register (ADC_ISR) . . . 977
24.5.2 ADC interrupt enable register (ADC_IER) . . . 979
24.5.3 ADC control register (ADC_CR) . . . 981
24.5.4 ADC configuration register (ADC_CFGR) . . . 986

24.5.5 ADC configuration register 2 (ADC_CFGR2) 990
24.5.6 ADC sample time register 1 (ADC_SMPR1) 992
24.5.7 ADC sample time register 2 (ADC_SMPR2) 993
24.5.8 ADC channel preselection register (ADC_PCSEL) 994
24.5.9 ADC watchdog threshold register 1 (ADC_LTR1) 994
24.5.10 ADC watchdog threshold register 1 (ADC_HTR1) 995
24.5.11 ADC regular sequence register 1 (ADC_SQR1) 996
24.5.12 ADC regular sequence register 2 (ADC_SQR2) 997
24.5.13 ADC regular sequence register 3 (ADC_SQR3) 998
24.5.14 ADC regular sequence register 4 (ADC_SQR4) 999
24.5.15 ADC regular Data Register (ADC_DR) 1000
24.5.16 ADC injected sequence register (ADC_JSQR) 1001
24.5.17 ADC offset register (ADC_OFRy) 1003
24.5.18 ADC injected data register (ADC_JDRy) 1004
24.5.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) 1004
24.5.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) 1005
24.5.21 ADC watchdog lower threshold register 2 (ADC_LTR2) 1005
24.5.22 ADC watchdog higher threshold register 2 (ADC_HTR2) 1006
24.5.23 ADC watchdog lower threshold register 3 (ADC_LTR3) 1006
24.5.24 ADC watchdog higher threshold register 3 (ADC_HTR3) 1007
24.5.25 ADC differential mode selection register (ADC_DIFSEL) 1007
24.5.26 ADC calibration factors register (ADC_CALFACT) 1008
24.5.27 ADC calibration factor register 2 (ADC_CALFACT2) 1008
24.6 ADC common registers 1010
24.6.1 ADC x common status register (ADCx_CSR) (x=1/2 or 3) 1010
24.6.2 ADC x common control register (ADCx_CCR) (x=1/2 or 3) 1012
24.6.3 ADC x common regular data register for dual mode (ADCx_CDR) (x=1/2 or 3) 1015
24.6.4 ADC x common regular data register for 32-bit dual mode (ADCx_CDR2) (x=1/2 or 3) 1015
24.6.5 ADC register map 1016

25 Digital-to-analog converter (DAC) 1020
25.1 Introduction 1020
25.2 DAC main features 1020
25.3 DAC implementation 1021

25.4 DAC functional description 1022
25.4.1 DAC block diagram 1022
25.4.2 DAC pins and internal signals 1023
25.4.3 DAC channel enable 1024
25.4.4 DAC data format 1024
25.4.5 DAC conversion 1025
25.4.6 DAC output voltage 1025
25.4.7 DAC trigger selection 1026
25.4.8 DMA requests 1027
25.4.9 Noise generation 1027
25.4.10 Triangle-wave generation 1029
25.4.11 DAC channel modes 1029
25.4.12 DAC channel buffer calibration 1033
25.4.13 Dual DAC channel conversion (if available) 1034
25.5 DAC low-power modes 1039
25.6 DAC interrupts 1039
25.7 DAC registers 1040
25.7.1 DAC control register (DAC_CR) 1040
25.7.2 DAC software trigger register (DAC_SWTRGR) 1043
25.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) 1044
25.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) 1044
25.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) 1045
25.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) 1045
25.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) 1046
25.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) 1046
25.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) 1047
25.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) 1047
25.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) 1048
25.7.12 DAC channel1 data output register (DAC_DOR1) 1048
25.7.13 DAC channel2 data output register (DAC_DOR2) 1049

25.7.14 DAC status register (DAC_SR) 1049
25.7.15 DAC calibration control register (DAC_CCR) 1051
25.7.16 DAC mode control register (DAC_MCR) 1051
25.7.17 DAC channel 1 sample and hold sample time register (DAC_SHSR1) . . . 1053
25.7.18 DAC channel 2 sample and hold sample time register (DAC_SHSR2) . . . 1053
25.7.19 DAC sample and hold time register (DAC_SHHR) 1054
25.7.20 DAC sample and hold refresh time register (DAC_SHRR) 1054
25.7.21 DAC register map 1056

26 Voltage reference buffer (VREFBUF) 1058
26.1 Introduction 1058
26.2 VREFBUF functional description 1058
26.3 VREFBUF registers 1059
26.3.1 VREFBUF control and status register (VREFBUF_CSR) 1059
26.3.2 VREFBUF calibration control register (VREFBUF_CCR) 1060
26.3.3 VREFBUF register map 1060

27 Comparator (COMP) 1061
27.1 Introduction 1061
27.2 COMP main features 1061
27.3 COMP functional description 1062
27.3.1 COMP block diagram 1062
27.3.2 COMP pins and internal signals 1062
27.3.3 COMP reset and clocks 1064
27.3.4 Comparator LOCK mechanism 1064
27.3.5 Window comparator 1064
27.3.6 Hysteresis 1064
27.3.7 Comparator output blanking function 1065
27.3.8 Comparator output on GPIOs 1066
27.3.9 Comparator output redirection 1067
27.3.10 COMP power and speed modes 1067
27.4 COMP low-power modes 1068
27.5 COMP interrupts 1068
27.5.1 Interrupt through EXTI block 1068
27.5.2 Interrupt through NVIC of the CPU 1069

27.6 SCALER function 1069
27.7 COMP registers 1070
27.7.1 Comparator status register (COMP_SR) 1070
27.7.2 Comparator interrupt clear flag register (COMP_ICFR) 1070
27.7.3 Comparator option register (COMP_OR) 1071
27.7.4 Comparator configuration register 1 (COMP_CFGR1) 1072
27.7.5 Comparator configuration register 2 (COMP_CFGR2) 1074
27.7.6 COMP register map 1077

28 Operational amplifiers (OPAMP) 1078
28.1 Introduction 1078
28.2 OPAMP main features 1078
28.3 OPAMP functional description 1078
28.3.1 OPAMP reset and clocks 1078
28.3.2 Initial configuration 1079
28.3.3 Signal routing 1079
28.3.4 OPAMP modes 1080
28.3.5 Calibration 1086
28.4 OPAMP low-power modes 1088
28.5 OPAMP PGA gain 1088
28.6 OPAMP registers 1088
28.6.1 OPAMP1 control/status register (OPAMP1_CSR) 1088
28.6.2 OPAMP1 trimming register in normal mode (OPAMP1_OTR) 1090
28.6.3 OPAMP1 trimming register in high-speed mode (OPAMP1_HSOTR) 1091
28.6.4 OPAMP option register (OPAMP_OR) 1091
28.6.5 OPAMP2 control/status register (OPAMP2_CSR) 1091
28.6.6 OPAMP2 trimming register in normal mode (OPAMP2_OTR) 1093
28.6.7 OPAMP2 trimming register in high-speed mode (OPAMP2_HSOTR) 1094
28.6.8 OPAMP register map 1095

29 Digital filter for sigma delta modulators (DFSDM) 1096
29.1 Introduction 1096
29.2 DFSDM main features 1097
29.3 DFSDM implementation 1098
29.4 DFSDM functional description 1099
29.4.1 DFSDM block diagram 1099

29.4.2 DFSDM pins and internal signals 1100
29.4.3 DFSDM reset and clocks 1101
29.4.4 Serial channel transceivers 1102
29.4.5 Configuring the input serial interface 1111
29.4.6 Parallel data inputs 1111
29.4.7 Channel selection 1113
29.4.8 Digital filter configuration 1114
29.4.9 Integrator unit 1115
29.4.10 Analog watchdog 1116
29.4.11 Short-circuit detector 1118
29.4.12 Extreme detector 1119
29.4.13 Data unit block 1119
29.4.14 Signed data format 1120
29.4.15 Launching conversions 1121
29.4.16 Continuous and fast continuous modes 1121
29.4.17 Request precedence 1122
29.4.18 Power optimization in run mode 1123
29.5 DFSDM interrupts 1123
29.6 DFSDM DMA transfer 1125
29.7 DFSDM channel y registers (y=0..7) 1125
29.7.1 DFSDM channel y configuration register (DFSDM_CHyCFGR1) . . . 1125
29.7.2 DFSDM channel y configuration register (DFSDM_CHyCFGR2) . . . 1127
29.7.3 DFSDM channel y analog watchdog and short-circuit detector register (DFSDM_CHyAWSCDR) 1128
29.7.4 DFSDM channel y watchdog filter data register (DFSDM_CHyWDATR) 1129
29.7.5 DFSDM channel y data input register (DFSDM_CHyDATINR) 1129
29.8 DFSDM filter x module registers (x=0..3) 1130
29.8.1 DFSDM filter x control register 1 (DFSDM_FLTxCR1) 1130
29.8.2 DFSDM filter x control register 2 (DFSDM_FLTxCR2) 1133
29.8.3 DFSDM filter x interrupt and status register (DFSDM_FLTxISR) 1134
29.8.4 DFSDM filter x interrupt flag clear register (DFSDM_FLTxICR) 1136
29.8.5 DFSDM filter x injected channel group selection register (DFSDM_FLTxJCHGR) 1137
29.8.6 DFSDM filter x control register (DFSDM_FLTxFCR) 1137
29.8.7 DFSDM filter x data register for injected group (DFSDM_FLTxJDATAR) 1138

29.8.8 DFSDM filter x data register for the regular channel (DFSDM_FLTxRDATAR) . . . 1139
29.8.9 DFSDM filter x analog watchdog high threshold register (DFSDM_FLTxAWHTR) . . . 1140
29.8.10 DFSDM filter x analog watchdog low threshold register (DFSDM_FLTxAWLTR) . . . 1140
29.8.11 DFSDM filter x analog watchdog status register (DFSDM_FLTxAWSR) . . . 1141
29.8.12 DFSDM filter x analog watchdog clear flag register (DFSDM_FLTxAWCFR) . . . 1142
29.8.13 DFSDM filter x extremes detector maximum register (DFSDM_FLTxEXMAX) . . . 1142
29.8.14 DFSDM filter x extremes detector minimum register (DFSDM_FLTxEXMIN) . . . 1143
29.8.15 DFSDM filter x conversion timer register (DFSDM_FLTxCNVTIMR) . 1143
29.8.16 DFSDM register map . . . 1144

30 Digital camera interface (DCMI) . . . 1154

30.1 DCMI introduction . . . 1154
30.2 DCMI main features . . . 1154
30.3 DCMI clocks . . . 1154
30.4 DCMI functional overview . . . 1154
30.4.1 DCMI block diagram . . . 1155
30.4.2 DCMI internal signals . . . 1156
30.4.3 DMA interface . . . 1156
30.4.4 DCMI physical interface . . . 1156
30.4.5 Synchronization . . . 1158
30.4.6 Capture modes . . . 1161
30.4.7 Crop feature . . . 1162
30.4.8 JPEG format . . . 1163
30.4.9 FIFO . . . 1163
30.5 Data format description . . . 1164
30.5.1 Data formats . . . 1164
30.5.2 Monochrome format . . . 1164
30.5.3 RGB format . . . 1165
30.5.4 YCbCr format . . . 1165
30.5.5 YCbCr format - Y only . . . 1165
30.5.6 Half resolution image extraction . . . 1166

30.6 DCMI interrupts 1166
30.7 DCMI register description 1166
30.7.1 DCMI control register (DCMI_CR) 1166
30.7.2 DCMI status register (DCMI_SR) 1170
30.7.3 DCMI raw interrupt status register (DCMI_RIS) 1171
30.7.4 DCMI interrupt enable register (DCMI_IER) 1172
30.7.5 DCMI masked interrupt status register (DCMI_MIS) 1173
30.7.6 DCMI interrupt clear register (DCMI_ICR) 1174
30.7.7 DCMI embedded synchronization code register (DCMI_ESCR) 1175
30.7.8 DCMI embedded synchronization unmask register (DCMI_ESUR) . 1176
30.7.9 DCMI crop window start (DCMI_CWSTRT) 1177
30.7.10 DCMI crop window size (DCMI_CWSIZE) 1177
30.7.11 DCMI data register (DCMI_DR) 1178
30.7.12 DCMI register map 1179

31 LCD-TFT display controller (LTDC) 1180
31.1 Introduction 1180
31.2 LTDC main features 1180
31.3 LTDC functional description 1181
31.3.1 LTDC block diagram 1181
31.3.2 LCD-TFT internal signals 1181
31.3.3 LTDC pins and external signal interface 1182
31.3.4 LTDC reset and clocks 1182
31.4 LTDC programmable parameters 1184
31.4.1 LTDC global configuration parameters 1184
31.4.2 Layer programmable parameters 1186
31.5 LTDC interrupts 1191
31.6 LTDC programming procedure 1192
31.7 LTDC registers 1193
31.7.1 LTDC synchronization size configuration register (LTDC_SSCR) . . . 1193
31.7.2 LTDC back porch configuration register (LTDC_BPCR) 1193
31.7.3 LTDC active width configuration register (LTDC_AWCR) 1194
31.7.4 LTDC total width configuration register (LTDC_TWCR) 1195
31.7.5 LTDC global control register (LTDC_GCR) 1195
31.7.6 LTDC shadow reload configuration register (LTDC_SRCR) 1197
31.7.7 LTDC background color configuration register (LTDC_BCCR) 1197

31.7.8 LTDC interrupt enable register (LTDC_IER) 1198
31.7.9 LTDC interrupt status register (LTDC_ISR) 1199
31.7.10 LTDC Interrupt Clear Register (LTDC_ICR) 1199
31.7.11 LTDC line interrupt position configuration register (LTDC_LIPCR) . . 1200
31.7.12 LTDC current position status register (LTDC_CPSR) 1200
31.7.13 LTDC current display status register (LTDC_CDSR) 1201
31.7.14 LTDC layer x control register (LTDC_LxCR) 1202
31.7.15 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR) 1203
31.7.16 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR) 1204
31.7.17 LTDC layer x color keying configuration register (LTDC_LxCKCR) 1205
31.7.18 LTDC layer x pixel format configuration register (LTDC_LxPFCR) 1205
31.7.19 LTDC layer x constant alpha configuration register (LTDC_LxCACR) 1206
31.7.20 LTDC layer x default color configuration register (LTDC_LxDCCR) 1207
31.7.21 LTDC layer x blending factors configuration register (LTDC_LxBFCR) 1208
31.7.22 LTDC layer x color frame buffer address register (LTDC_LxCFBAR) 1209
31.7.23 LTDC layer x color frame buffer length register (LTDC_LxCFBLR) 1209
31.7.24 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR) 1210
31.7.25 LTDC layer x CLUT write register (LTDC_LxCLUTWR) 1211
31.7.26 LTDC register map 1212

32 JPEG codec (JPEG) 1215

32.1 Introduction 1215
32.2 JPEG codec main features 1215
32.3 JPEG codec block functional description 1216
32.3.1 General description 1216
32.3.2 JPEG internal signals 1216
32.3.3 JPEG decoding procedure 1217
32.3.4 JPEG encoding procedure 1218
32.4 JPEG codec interrupts 1220

32.5 JPEG codec registers . . . 1221
32.5.1 JPEG codec control register (JPEG_CONFR0) . . . 1221
32.5.2 JPEG codec configuration register 1 (JPEG_CONFR1) . . . 1221
32.5.3 JPEG codec configuration register 2 (JPEG_CONFR2) . . . 1222
32.5.4 JPEG codec configuration register 3 (JPEG_CONFR3) . . . 1223
32.5.5 JPEG codec configuration register 4-7 (JPEG_CONFR4-7) . . . 1223
32.5.6 JPEG control register (JPEG_CR) . . . 1224
32.5.7 JPEG status register (JPEG_SR) . . . 1225
32.5.8 JPEG clear flag register (JPEG_CFR) . . . 1226
32.5.9 JPEG data input register (JPEG_DIR) . . . 1227
32.5.10 JPEG data output register (JPEG_DOR) . . . 1227
32.5.11 JPEG codec register map . . . 1228

33 True random number generator (RNG) . . . 1230
33.1 Introduction . . . 1230
33.2 RNG main features . . . 1230
33.3 RNG functional description . . . 1231
33.3.1 RNG block diagram . . . 1231
33.3.2 RNG internal signals . . . 1231
33.3.3 Random number generation . . . 1232
33.3.4 RNG initialization . . . 1234
33.3.5 RNG operation . . . 1235
33.3.6 RNG clocking . . . 1236
33.3.7 Error management . . . 1236
33.4 RNG low-power usage . . . 1237
33.5 RNG interrupts . . . 1238
33.6 RNG processing time . . . 1238
33.7 Entropy source validation . . . 1238
33.7.1 Introduction . . . 1238
33.7.2 Validation conditions . . . 1239
33.7.3 Data collection . . . 1239
33.8 RNG registers . . . 1240
33.8.1 RNG control register (RNG_CR) . . . 1240
33.8.2 RNG status register (RNG_SR) . . . 1241
33.8.3 RNG data register (RNG_DR) . . . 1242
33.8.4 RNG register map . . . 1243

34 Cryptographic processor (CRYP) 1244

34.1 Introduction 1244

34.2 CRYP main features 1244

34.3 CRYP functional description 1246

34.3.1 CRYP block diagram 1246

34.3.2 CRYP internal signals 1247

34.3.3 CRYP DES/TDES cryptographic core 1247

34.3.4 CRYP AES cryptographic core 1248

34.3.5 CRYP procedure to perform a cipher operation 1254

34.3.6 CRYP busy state 1257

34.3.7 Preparing the CRYP AES key for decryption 1258

34.3.8 CRYP stealing and data padding 1258

34.3.9 CRYP suspend/resume operations 1260

34.3.10 CRYP DES/TDES basic chaining modes (ECB, CBC) 1261

34.3.11 CRYP AES basic chaining modes (ECB, CBC) 1266

34.3.12 CRYP AES counter mode (AES-CTR) 1271

34.3.13 CRYP AES Galois/counter mode (GCM) 1275

34.3.14 CRYP AES Galois message authentication code (GMAC) 1280

34.3.15 CRYP AES Counter with CBC-MAC (CCM) 1281

34.3.16 CRYP data registers and data swapping 1287

34.3.17 CRYP key registers 1291

34.3.18 CRYP initialization vector registers 1292

34.3.19 CRYP DMA interface 1293

34.3.20 CRYP error management 1295

34.4 CRYP interrupts 1295

34.5 CRYP processing time 1297

34.6 CRYP registers 1298

34.6.1 CRYP control register (CRYP_CR) 1298

34.6.2 CRYP status register (CRYP_SR) 1300

34.6.3 CRYP data input register (CRYP_DIN) 1300

34.6.4 CRYP data output register (CRYP_DOUT) 1301

34.6.5 CRYP DMA control register (CRYP_DMACR) 1302

34.6.6 CRYP interrupt mask set/clear register (CRYP_IMSCR) 1302

34.6.7 CRYP raw interrupt status register (CRYP_RISR) 1303

34.6.8 CRYP masked interrupt status register (CRYP_MISR) 1303

34.6.9 CRYP key register 0L (CRYP_K0LR) 1304

34.6.10 CRYP key register 0R (CRYP_K0RR) 1305
34.6.11 CRYP key register 1L (CRYP_K1LR) 1305
34.6.12 CRYP key register 1R (CRYP_K1RR) 1306
34.6.13 CRYP key register 2L (CRYP_K2LR) 1306
34.6.14 CRYP key register 2R (CRYP_K2RR) 1306
34.6.15 CRYP key register 3L (CRYP_K3LR) 1307
34.6.16 CRYP key register 3R (CRYP_K3RR) 1307
34.6.17 CRYP initialization vector register 0L (CRYP_IV0LR) 1308
34.6.18 CRYP initialization vector register 0R (CRYP_IV0RR) 1308
34.6.19 CRYP initialization vector register 1L (CRYP_IV1LR) 1309
34.6.20 CRYP initialization vector register 1R (CRYP_IV1RR) 1309
34.6.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR) 1309
34.6.22 CRYP context swap GCM registers (CRYP_CSGCMxR) 1310
34.6.23 CRYP register map 1311

35 Hash processor (HASH) 1314

35.1 Introduction 1314
35.2 HASH main features 1314
35.3 HASH functional description 1315
35.3.1 HASH block diagram 1315
35.3.2 HASH internal signals 1315
35.3.3 About secure hash algorithms 1316
35.3.4 Message data feeding 1316
35.3.5 Message digest computing 1318
35.3.6 Message padding 1319
35.3.7 HMAC operation 1321
35.3.8 Context swapping 1323
35.3.9 HASH DMA interface 1325
35.3.10 HASH error management 1325
35.4 HASH interrupts 1325
35.5 HASH processing time 1326
35.6 HASH registers 1327
35.6.1 HASH control register (HASH_CR) 1327
35.6.2 HASH data input register (HASH_DIN) 1330
35.6.3 HASH start register (HASH_STR) 1331
35.6.4 HASH digest registers (HASH_HR0..7) 1332

35.6.5 HASH interrupt enable register (HASH_IMR) 1335
35.6.6 HASH status register (HASH_SR) 1336
35.6.7 HASH context swap registers (HASH_CSRx) 1337
35.6.8 HASH register map 1338

36 High-Resolution Timer (HRTIM) 1339
36.1 Introduction 1339
36.2 Main features 1340
36.3 Functional description 1341
36.3.1 General description 1341
36.3.2 HRTIM pins and internal signals 1343
36.3.3 Clocks 1344
36.3.4 Timer A..E timing units 1347
36.3.5 Master timer 1364
36.3.6 Set/reset events priorities and narrow pulses management 1365
36.3.7 External events global conditioning 1366
36.3.8 External event filtering in timing units 1371
36.3.9 Delayed Protection 1376
36.3.10 Register preload and update management 1382
36.3.11 Events propagation within or across multiple timers 1385
36.3.12 Output management 1389
36.3.13 Burst mode controller 1391
36.3.14 Chopper 1400
36.3.15 Fault protection 1401
36.3.16 Auxiliary outputs 1404
36.3.17 Synchronizing the HRTIM with other timers or HRTIM instances . . . 1407
36.3.18 ADC triggers 1410
36.3.19 DAC triggers 1411
36.3.20 HRTIM Interrupts 1413
36.3.21 DMA 1415
36.3.22 HRTIM initialization 1418
36.3.23 Debug 1419
36.4 Application use cases 1420
36.4.1 Buck converter 1420
36.4.2 Buck converter with synchronous rectification 1421
36.4.3 Multiphase converters 1422
36.4.4 Transition mode Power Factor Correction 1424

36.5 HRTIM registers 1426
36.5.1 HRTIM Master Timer Control Register (HRTIM_MCR) 1426
36.5.2 HRTIM Master Timer Interrupt Status Register (HRTIM_MISR) 1429
36.5.3 HRTIM Master Timer Interrupt Clear Register (HRTIM_MICR) 1430
36.5.4 HRTIM Master Timer DMA / Interrupt Enable Register (HRTIM_MDIER) 1431
36.5.5 HRTIM Master Timer Counter Register (HRTIM_MCNTR) 1433
36.5.6 HRTIM Master Timer Period Register (HRTIM_MPER) 1433
36.5.7 HRTIM Master Timer Repetition Register (HRTIM_MREP) 1434
36.5.8 HRTIM Master Timer Compare 1 Register (HRTIM_MCMP1R) 1434
36.5.9 HRTIM Master Timer Compare 2 Register (HRTIM_MCMP2R) 1435
36.5.10 HRTIM Master Timer Compare 3 Register (HRTIM_MCMP3R) 1435
36.5.11 HRTIM Master Timer Compare 4 Register (HRTIM_MCMP4R) 1436
36.5.12 HRTIM Timerx Control Register (HRTIM_TIMxCR) 1437
36.5.13 HRTIM Timerx Interrupt Status Register (HRTIM_TIMxISR) 1441
36.5.14 HRTIM Timerx Interrupt Clear Register (HRTIM_TIMxICR) 1443
36.5.15 HRTIM Timerx DMA / Interrupt Enable Register (HRTIM_TIMxDIER) 1444
36.5.16 HRTIM Timerx Counter Register (HRTIM_CNTxR) 1447
36.5.17 HRTIM Timerx Period Register (HRTIM_PERxR) 1447
36.5.18 HRTIM Timerx Repetition Register (HRTIM_REPxR) 1448
36.5.19 HRTIM Timerx Compare 1 Register (HRTIM_CMP1xR) 1448
36.5.20 HRTIM Timerx Compare 1 Compound Register (HRTIM_CMP1CxR) 1449
36.5.21 HRTIM Timerx Compare 2 Register (HRTIM_CMP2xR) 1449
36.5.22 HRTIM Timerx Compare 3 Register (HRTIM_CMP3xR) 1450
36.5.23 HRTIM Timerx Compare 4 Register (HRTIM_CMP4xR) 1450
36.5.24 HRTIM Timerx Capture 1 Register (HRTIM_CPT1xR) 1451
36.5.25 HRTIM Timerx Capture 2 Register (HRTIM_CPT2xR) 1451
36.5.26 HRTIM Timerx Deadtime Register (HRTIM_DTxR) 1452
36.5.27 HRTIM Timerx Output1 Set Register (HRTIM_SETx1R) 1454
36.5.28 HRTIM Timerx Output1 Reset Register (HRTIM_RSTx1R) 1456
36.5.29 HRTIM Timerx Output2 Set Register (HRTIM_SETx2R) 1456
36.5.30 HRTIM Timerx Output2 Reset Register (HRTIM_RSTx2R) 1457
36.5.31 HRTIM Timerx External Event Filtering Register 1 (HRTIM_EEFxR1) 1458
36.5.32 HRTIM Timerx External Event Filtering Register 2 (HRTIM_EEFxR2) 1460

36.5.33 HRTIM Timerx Reset Register (HRTIM_RSTxR) 1461
36.5.34 HRTIM Timerx Chopper Register (HRTIM_CHPxR) 1464
36.5.35 HRTIM Timerx Capture 1 Control Register (HRTIM_CPT1xCR) 1466
36.5.36 HRTIM Timerx Capture 2 Control Register (HRTIM_CPT2xCR) 1467
36.5.37 HRTIM Timerx Output Register (HRTIM_OUTxR) 1470
36.5.38 HRTIM Timerx Fault Register (HRTIM_FLTxR) 1473
36.5.39 HRTIM Control Register 1 (HRTIM_CR1) 1474
36.5.40 HRTIM Control Register 2 (HRTIM_CR2) 1476
36.5.41 HRTIM Interrupt Status Register (HRTIM_ISR) 1477
36.5.42 HRTIM Interrupt Clear Register (HRTIM_ICR) 1478
36.5.43 HRTIM Interrupt Enable Register (HRTIM_IER) 1479
36.5.44 HRTIM Output Enable Register (HRTIM_OENR) 1480
36.5.45 HRTIM Output Disable Register (HRTIM_ODISR) 1481
36.5.46 HRTIM Output Disable Status Register (HRTIM_ODSR) 1482
36.5.47 HRTIM Burst Mode Control Register (HRTIM_BMCR) 1483
36.5.48 HRTIM Burst Mode Trigger Register (HRTIM_BMTRGR) 1485
36.5.49 HRTIM Burst Mode Compare Register (HRTIM_BMCMPR) 1487
36.5.50 HRTIM Burst Mode Period Register (HRTIM_BMPER) 1487
36.5.51 HRTIM Timer External Event Control Register 1 (HRTIM_EECR1) . 1488
36.5.52 HRTIM Timer External Event Control Register 2 (HRTIM_EECR2) . 1490
36.5.53 HRTIM Timer External Event Control Register 3 (HRTIM_EECR3) . 1491
36.5.54 HRTIM ADC Trigger 1 Register (HRTIM_ADC1R) 1492
36.5.55 HRTIM ADC Trigger 2 Register (HRTIM_ADC2R) 1493
36.5.56 HRTIM ADC Trigger 3 Register (HRTIM_ADC3R) 1494
36.5.57 HRTIM ADC Trigger 4 Register (HRTIM_ADC4R) 1496
36.5.58 HRTIM Fault Input Register 1 (HRTIM_FLTINR1) 1498
36.5.59 HRTIM Fault Input Register 2 (HRTIM_FLTINR2) 1500
36.5.60 HRTIM Burst DMA Master timer update Register (HRTIM_BDMUPR) 1502
36.5.61 HRTIM Burst DMA Timerx update Register (HRTIM_BDTxUPR) . . . 1503
36.5.62 HRTIM Burst DMA Data Register (HRTIM_BDMADR) 1504
36.5.63 HRTIM register map 1505

37 Advanced-control timers (TIM1/TIM8) 1514

37.1 TIM1/TIM8 introduction 1514
37.2 TIM1/TIM8 main features 1514
37.3 TIM1/TIM8 functional description 1516

37.3.1 Time-base unit . . . 1516
37.3.2 Counter modes . . . 1518
37.3.3 Repetition counter . . . 1529
37.3.4 External trigger input . . . 1531
37.3.5 Clock selection . . . 1532
37.3.6 Capture/compare channels . . . 1536
37.3.7 Input capture mode . . . 1538
37.3.8 PWM input mode . . . 1539
37.3.9 Forced output mode . . . 1540
37.3.10 Output compare mode . . . 1541
37.3.11 PWM mode . . . 1542
37.3.12 Asymmetric PWM mode . . . 1545
37.3.13 Combined PWM mode . . . 1546
37.3.14 Combined 3-phase PWM mode . . . 1547
37.3.15 Complementary outputs and dead-time insertion . . . 1548
37.3.16 Using the break function . . . 1550
37.3.17 Bidirectional break inputs . . . 1556
37.3.18 Clearing the OCxREF signal on an external event . . . 1556
37.3.19 6-step PWM generation . . . 1558
37.3.20 One-pulse mode . . . 1559
37.3.21 Retriggerable one pulse mode (OPM) . . . 1560
37.3.22 Encoder interface mode . . . 1561
37.3.23 UIF bit remapping . . . 1563
37.3.24 Timer input XOR function . . . 1564
37.3.25 Interfacing with Hall sensors . . . 1564
37.3.26 Timer synchronization . . . 1567
37.3.27 ADC synchronization . . . 1571
37.3.28 DMA burst mode . . . 1571
37.3.29 Debug mode . . . 1572
37.4 TIM1/TIM8 registers . . . 1573
37.4.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8) . . . 1573
37.4.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8) . . . 1574
37.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8) . . . 1577
37.4.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8) . . . 1579
37.4.5 TIMx status register (TIMx_SR)(x = 1, 8) . . . 1581
37.4.6 TIMx event generation register (TIMx_EGR)(x = 1, 8) . . . 1583
37.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 1, 8) . . . 1584

37.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 1, 8) . . 1588
37.4.9 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8) 1590
37.4.10 TIMx counter (TIMx_CNT)(x = 1, 8) 1594
37.4.11 TIMx prescaler (TIMx_PSC)(x = 1, 8) 1594
37.4.12 TIMx auto-reload register (TIMx_ARR)(x = 1, 8) 1594
37.4.13 TIMx repetition counter register (TIMx_RCR)(x = 1, 8) 1595
37.4.14 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8) 1595
37.4.15 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8) 1596
37.4.16 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8) 1596
37.4.17 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8) 1597
37.4.18 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8) 1597
37.4.19 TIMx DMA control register (TIMx_DCR)(x = 1, 8) 1600
37.4.20 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8) 1601
37.4.21 TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8) . . 1602
37.4.22 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8) 1603
37.4.23 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8) 1604
37.4.24 TIM1 alternate function option register 1 (TIM1_AF1) 1604
37.4.25 TIM1 Alternate function register 2 (TIM1_AF2) 1606
37.4.26 TIM8 Alternate function option register 1 (TIM8_AF1) 1607
37.4.27 TIM8 Alternate function option register 2 (TIM8_AF2) 1609
37.4.28 TIM1 timer input selection register (TIM1_TISEL) 1611
37.4.29 TIM8 timer input selection register (TIM8_TISEL) 1611
37.4.30 TIM1 register map 1613
37.4.31 TIM8 register map 1616

38 General-purpose timers (TIM2/TIM3/TIM4/TIM5) 1619

38.1 TIM2/TIM3/TIM4/TIM5 introduction 1619
38.2 TIM2/TIM3/TIM4/TIM5 main features 1619
38.3 TIM2/TIM3/TIM4/TIM5 functional description 1621
38.3.1 Time-base unit 1621
38.3.2 Counter modes 1623
38.3.3 Clock selection 1633
38.3.4 Capture/Compare channels 1637
38.3.5 Input capture mode 1639
38.3.6 PWM input mode 1640
38.3.7 Forced output mode 1641
38.3.8 Output compare mode 1641

38.3.9 PWM mode . . . 1642
38.3.10 Asymmetric PWM mode . . . 1646
38.3.11 Combined PWM mode . . . 1646
38.3.12 Clearing the OCxREF signal on an external event . . . 1647
38.3.13 One-pulse mode . . . 1649
38.3.14 Retriggerable one pulse mode (OPM) . . . 1650
38.3.15 Encoder interface mode . . . 1651
38.3.16 UIF bit remapping . . . 1653
38.3.17 Timer input XOR function . . . 1653
38.3.18 Timers and external trigger synchronization . . . 1654
38.3.19 Timer synchronization . . . 1657
38.3.20 DMA burst mode . . . 1661
38.3.21 Debug mode . . . 1662
38.4 TIM2/TIM3/TIM4/TIM5 registers . . . 1664
38.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5) . . . 1664
38.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5) . . . 1665
38.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5) . . . 1667
38.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5) . . . 1670
38.4.5 TIMx status register (TIMx_SR)(x = 2 to 5) . . . 1671
38.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5) . . . 1672
38.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5) . 1673
38.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5) . 1677
38.4.9 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5) . . . 1679
38.4.10 TIMx counter (TIMx_CNT)(x = 2 to 5) . . . 1680
38.4.11 TIMx prescaler (TIMx_PSC)(x = 2 to 5) . . . 1681
38.4.12 TIMx auto-reload register (TIMx_ARR)(x = 2 to 5) . . . 1681
38.4.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5) . . . 1682
38.4.14 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5) . . . 1682
38.4.15 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5) . . . 1683
38.4.16 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5) . . . 1683
38.4.17 TIMx DMA control register (TIMx_DCR)(x = 2 to 5) . . . 1684
38.4.18 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5) . . . 1684
38.4.19 TIM2 alternate function option register 1 (TIM2_AF1) . . . 1685
38.4.20 TIM3 alternate function option register 1 (TIM3_AF1) . . . 1685
38.4.21 TIM4 alternate function option register 1 (TIM4_AF1) . . . 1686
38.4.22 TIM5 alternate function option register 1 (TIM5_AF1) . . . 1686
38.4.23 TIM2 timer input selection register (TIM2_TISEL) . . . 1687

38.4.24 TIM3 timer input selection register (TIM3_TISEL) 1687
38.4.25 TIM4 timer input selection register (TIM4_TISEL) 1688
38.4.26 TIM5 timer input selection register (TIM5_TISEL) 1689
38.4.27 TIMx register map 1691

39 General-purpose timers (TIM12/TIM13/TIM14) 1694

39.1 TIM12/TIM13/TIM14 introduction 1694
39.2 TIM12/TIM13/TIM14 main features 1694
39.2.1 TIM12 main features 1694
39.2.2 TIM13/TIM14 main features 1695
39.3 TIM12/TIM13/TIM14 functional description 1697
39.3.1 Time-base unit 1697
39.3.2 Counter modes 1699
39.3.3 Clock selection 1702
39.3.4 Capture/compare channels 1704
39.3.5 Input capture mode 1706
39.3.6 PWM input mode (only for TIM12) 1707
39.3.7 Forced output mode 1708
39.3.8 Output compare mode 1709
39.3.9 PWM mode 1710
39.3.10 Combined PWM mode (TIM12 only) 1711
39.3.11 One-pulse mode 1712
39.3.12 Retriggerable one pulse mode (OPM) (TIM12 only) 1714
39.3.13 UIF bit remapping 1715
39.3.14 Timer input XOR function 1715
39.3.15 TIM12 external trigger synchronization 1715
39.3.16 Slave mode – combined reset + trigger mode 1718
39.3.17 Timer synchronization (TIM12) 1719
39.3.18 Debug mode 1719
39.4 TIM12 registers 1719
39.4.1 TIM12 control register 1 (TIM12_CR1) 1719
39.4.2 TIM12 control register 2 (TIM12_CR2) 1720
39.4.3 TIM12 slave mode control register (TIM12_SMCR) 1721
39.4.4 TIM12 Interrupt enable register (TIM12_DIER) 1723
39.4.5 TIM12 status register (TIM12_SR) 1723
39.4.6 TIM12 event generation register (TIM12_EGR) 1724
39.4.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1) 1725

39.4.8 TIM12 capture/compare enable register (TIM12_CCER) 1729
39.4.9 TIM12 counter (TIM12_CNT) 1730
39.4.10 TIM12 prescaler (TIM12_PSC) 1730
39.4.11 TIM12 auto-reload register (TIM12_ARR) 1731
39.4.12 TIM12 capture/compare register 1 (TIM12_CCR1) 1731
39.4.13 TIM12 capture/compare register 2 (TIM12_CCR2) 1731
39.4.14 TIM12 timer input selection register (TIM12_TISEL) 1732
39.4.15 TIM12 register map 1733
39.5 TIM13/TIM14 registers 1735
39.5.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14) 1735
39.5.2 TIMx Interrupt enable register (TIMx_DIER)(x = 13, 14) 1736
39.5.3 TIMx status register (TIMx_SR)(x = 13, 14) 1736
39.5.4 TIMx event generation register (TIMx_EGR)(x = 13, 14) 1737
39.5.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13, 14) 1738
39.5.6 TIMx capture/compare enable register (TIMx_CCER)(x = 13, 14) 1740
39.5.7 TIMx counter (TIMx_CNT)(x = 13, 14) 1741
39.5.8 TIMx prescaler (TIMx_PSC)(x = 13, 14) 1742
39.5.9 TIMx auto-reload register (TIMx_ARR)(x = 13, 14) 1742
39.5.10 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13, 14) 1742
39.5.11 TIM13 timer input selection register (TIM13_TISEL) 1743
39.5.12 TIM14 timer input selection register (TIM14_TISEL) 1743
39.5.13 TIM13/TIM14 register map 1744

40 General-purpose timers (TIM15/TIM16/TIM17) 1746
40.1 TIM15/TIM16/TIM17 introduction 1746
40.2 TIM15 main features 1746
40.3 TIM16/TIM17 main features 1747
40.4 TIM15/TIM16/TIM17 functional description 1750
40.4.1 Time-base unit 1750
40.4.2 Counter modes 1752
40.4.3 Repetition counter 1756
40.4.4 Clock selection 1757
40.4.5 Capture/compare channels 1759
40.4.6 Input capture mode 1762
40.4.7 PWM input mode (only for TIM15) 1763

40.4.8 Forced output mode . . . 1764
40.4.9 Output compare mode . . . 1764
40.4.10 PWM mode . . . 1765
40.4.11 Combined PWM mode (TIM15 only) . . . 1766
40.4.12 Complementary outputs and dead-time insertion . . . 1768
40.4.13 Using the break function . . . 1770
40.4.14 One-pulse mode . . . 1774
40.4.15 Retriggerable one pulse mode (OPM) (TIM15 only) . . . 1775
40.4.16 UIF bit remapping . . . 1776
40.4.17 Timer input XOR function (TIM15 only) . . . 1777
40.4.18 External trigger synchronization (TIM15 only) . . . 1778
40.4.19 Slave mode – combined reset + trigger mode . . . 1780
40.4.20 DMA burst mode . . . 1780
40.4.21 Timer synchronization (TIM15) . . . 1782
40.4.22 Debug mode . . . 1782
40.5 TIM15 registers . . . 1783
40.5.1 TIM15 control register 1 (TIM15_CR1) . . . 1783
40.5.2 TIM15 control register 2 (TIM15_CR2) . . . 1784
40.5.3 TIM15 slave mode control register (TIM15_SMCR) . . . 1786
40.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER) . . . 1787
40.5.5 TIM15 status register (TIM15_SR) . . . 1788
40.5.6 TIM15 event generation register (TIM15_EGR) . . . 1790
40.5.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) . . . 1791
40.5.8 TIM15 capture/compare enable register (TIM15_CCER) . . . 1795
40.5.9 TIM15 counter (TIM15_CNT) . . . 1798
40.5.10 TIM15 prescaler (TIM15_PSC) . . . 1798
40.5.11 TIM15 auto-reload register (TIM15_ARR) . . . 1798
40.5.12 TIM15 repetition counter register (TIM15_RCR) . . . 1799
40.5.13 TIM15 capture/compare register 1 (TIM15_CCR1) . . . 1799
40.5.14 TIM15 capture/compare register 2 (TIM15_CCR2) . . . 1800
40.5.15 TIM15 break and dead-time register (TIM15_BDTR) . . . 1800
40.5.16 TIM15 DMA control register (TIM15_DCR) . . . 1803
40.5.17 TIM15 DMA address for full transfer (TIM15_DMAR) . . . 1803
40.5.18 TIM15 alternate register 1 (TIM15_AF1) . . . 1804
40.5.19 TIM15 input selection register (TIM15_TISEL) . . . 1805
40.5.20 TIM15 register map . . . 1806
40.6 TIM16/TIM17 registers . . . 1808

40.6.1 TIMx control register 1 (TIMx_CR1)(x = 16, 17) 1808
40.6.2 TIMx control register 2 (TIMx_CR2)(x = 16, 17) 1809
40.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16, 17) 1810
40.6.4 TIMx status register (TIMx_SR)(x = 16, 17) 1811
40.6.5 TIMx event generation register (TIMx_EGR)(x = 16, 17) 1812
40.6.6 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16, 17) 1813
40.6.7 TIMx capture/compare enable register (TIMx_CCER)(x = 16, 17) . . 1815
40.6.8 TIMx counter (TIMx_CNT)(x = 16, 17) 1817
40.6.9 TIMx prescaler (TIMx_PSC)(x = 16, 17) 1818
40.6.10 TIMx auto-reload register (TIMx_ARR)(x = 16, 17) 1818
40.6.11 TIMx repetition counter register (TIMx_RCR)(x = 16, 17) 1819
40.6.12 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16, 17) 1819
40.6.13 TIMx break and dead-time register (TIMx_BDTR)(x = 16, 17) 1820
40.6.14 TIMx DMA control register (TIMx_DCR)(x = 16, 17) 1822
40.6.15 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16, 17) 1823
40.6.16 TIM16 alternate function register 1 (TIM16_AF1) 1824
40.6.17 TIM16 input selection register (TIM16_TISEL) 1825
40.6.18 TIM17 alternate function register 1 (TIM17_AF1) 1826
40.6.19 TIM17 input selection register (TIM17_TISEL) 1827
40.6.20 TIM16/TIM17 register map 1828

41 Basic timers (TIM6/TIM7) 1830

41.1 TIM6/TIM7 introduction 1830
41.2 TIM6/TIM7 main features 1830
41.3 TIM6/TIM7 functional description 1831
41.3.1 Time-base unit 1831
41.3.2 Counting mode 1833
41.3.3 UIF bit remapping 1836
41.3.4 Clock source 1836
41.3.5 Debug mode 1837
41.4 TIM6/TIM7 registers 1837
41.4.1 TIM6/TIM7 control register 1 (TIMx_CR1) 1837
41.4.2 TIM6/TIM7 control register 2 (TIMx_CR2) 1839
41.4.3 TIM6/TIM7 DMA/Interrupt enable register (TIMx_DIER) 1839
41.4.4 TIM6/TIM7 status register (TIMx_SR) 1840
41.4.5 TIM6/TIM7 event generation register (TIMx_EGR) 1840
41.4.6 TIM6/TIM7 counter (TIMx_CNT) 1840

41.4.7 TIM6/TIM7 prescaler (TIMx_PSC) . . . 1841
41.4.8 TIM6/TIM7 auto-reload register (TIMx_ARR) . . . 1841
41.4.9 TIM6/TIM7 register map . . . 1842

42 Low-power timer (LPTIM) . . . 1843
42.1 Introduction . . . 1843
42.2 LPTIM main features . . . 1843
42.3 LPTIM implementation . . . 1843
42.4 LPTIM functional description . . . 1844
42.4.1 LPTIM block diagram . . . 1844
42.4.2 LPTIM pins and internal signals . . . 1846
42.4.3 LPTIM input and trigger mapping . . . 1846
42.4.4 LPTIM reset and clocks . . . 1849
42.4.5 Glitch filter . . . 1849
42.4.6 Prescaler . . . 1850
42.4.7 Trigger multiplexer . . . 1851
42.4.8 Operating mode . . . 1851
42.4.9 Timeout function . . . 1853
42.4.10 Waveform generation . . . 1853
42.4.11 Register update . . . 1854
42.4.12 Counter mode . . . 1855
42.4.13 Timer enable . . . 1856
42.4.14 Timer counter reset . . . 1856
42.4.15 Encoder mode . . . 1857
42.4.16 Debug mode . . . 1858
42.5 LPTIM low-power modes . . . 1858
42.6 LPTIM interrupts . . . 1859
42.7 LPTIM registers . . . 1859
42.7.1 LPTIM interrupt and status register (LPTIM_ISR) . . . 1859
42.7.2 LPTIM interrupt clear register (LPTIM_ICR) . . . 1861
42.7.3 LPTIM interrupt enable register (LPTIM_IER) . . . 1862
42.7.4 LPTIM configuration register (LPTIM_CFGR) . . . 1863
42.7.5 LPTIM control register (LPTIM_CR) . . . 1866
42.7.6 LPTIM compare register (LPTIM_CMP) . . . 1868
42.7.7 LPTIM autoreload register (LPTIM_ARR) . . . 1868
42.7.8 LPTIM counter register (LPTIM_CNT) . . . 1869

42.7.9 LPTIM configuration register 2 (LPTIM_CFGR2) 1869
42.7.10 LPTIM3 configuration register 2 (LPTIM3_CFGR2) 1870
42.7.11 LPTIM register map 1871

43 System window watchdog (WWDG) 1873
43.1 Introduction 1873
43.2 WWDG main features 1873
43.3 WWDG functional description 1873
43.3.1 WWDG block diagram 1874
43.3.2 WWDG internal signals 1874
43.3.3 Enabling the watchdog 1874
43.3.4 Controlling the downcounter 1874
43.3.5 Advanced watchdog interrupt feature 1875
43.3.6 How to program the watchdog timeout 1875
43.3.7 Debug mode 1876
43.4 WWDG registers 1877
43.4.1 Control register (WWDG_CR) 1877
43.4.2 Configuration register (WWDG_CFR) 1877
43.4.3 Status register (WWDG_SR) 1878
43.4.4 WWDG register map 1879

44 Independent watchdog (IWDG) 1880
44.1 Introduction 1880
44.2 IWDG main features 1880
44.3 IWDG functional description 1880
44.3.1 IWDG block diagram 1880
44.3.2 IWDG internal signals 1881
44.3.3 Window option 1881
44.3.4 Hardware watchdog 1882
44.3.5 Low-power freeze 1882
44.3.6 Behavior in Stop and Standby modes 1882
44.3.7 Register access protection 1882
44.3.8 Debug mode 1882
44.4 IWDG registers 1883
44.4.1 Key register (IWDG_KR) 1883
44.4.2 Prescaler register (IWDG_PR) 1884

44.4.3 Reload register (IWDG_RLR) . . . 1885
44.4.4 Status register (IWDG_SR) . . . 1886
44.4.5 Window register (IWDG_WINR) . . . 1887
44.4.6 IWDG register map . . . 1888

45 Real-time clock (RTC) . . . 1889
45.1 Introduction . . . 1889
45.2 RTC main features . . . 1890
45.3 RTC functional description . . . 1890
45.3.1 RTC block diagram . . . 1890
45.3.2 RTC pins and internal signals . . . 1892
45.3.3 GPIOs controlled by the RTC . . . 1893
45.3.4 Clock and prescalers . . . 1895
45.3.5 Real-time clock and calendar . . . 1896
45.3.6 Programmable alarms . . . 1896
45.3.7 Periodic auto-wakeup . . . 1896
45.3.8 RTC initialization and configuration . . . 1897
45.3.9 Reading the calendar . . . 1899
45.3.10 Resetting the RTC . . . 1900
45.3.11 RTC synchronization . . . 1900
45.3.12 RTC reference clock detection . . . 1901
45.3.13 RTC smooth digital calibration . . . 1902
45.3.14 Time-stamp function . . . 1904
45.3.15 Tamper detection . . . 1904
45.3.16 Calibration clock output . . . 1906
45.3.17 Alarm output . . . 1907
45.4 RTC low-power modes . . . 1907
45.5 RTC interrupts . . . 1907
45.6 RTC registers . . . 1908
45.6.1 RTC time register (RTC_TR) . . . 1908
45.6.2 RTC date register (RTC_DR) . . . 1909
45.6.3 RTC control register (RTC_CR) . . . 1911
45.6.4 RTC initialization and status register (RTC_ISR) . . . 1914
45.6.5 RTC prescaler register (RTC_PRER) . . . 1917
45.6.6 RTC wakeup timer register (RTC_WUTR) . . . 1918
45.6.7 RTC alarm A register (RTC_ALRMAR) . . . 1919

45.6.8 RTC alarm B register (RTC_ALRMBR) 1920
45.6.9 RTC write protection register (RTC_WPR) 1921
45.6.10 RTC sub second register (RTC_SSR) 1921
45.6.11 RTC shift control register (RTC_SHIFTR) 1922
45.6.12 RTC timestamp time register (RTC_TSTR) 1923
45.6.13 RTC timestamp date register (RTC_TSDR) 1924
45.6.14 RTC time-stamp sub second register (RTC_TSSSR) 1925
45.6.15 RTC calibration register (RTC_CALR) 1926
45.6.16 RTC tamper configuration register (RTC_TAMPCR) 1927
45.6.17 RTC alarm A sub second register (RTC_ALRMASSR) 1930
45.6.18 RTC alarm B sub second register (RTC_ALRMBSSR) 1931
45.6.19 RTC option register (RTC_OR) 1932
45.6.20 RTC backup registers (RTC_BKPxR) 1932
45.6.21 RTC register map 1933

46 Inter-integrated circuit (I2C) interface 1935

46.1 Introduction 1935
46.2 I2C main features 1935
46.3 I2C implementation 1936
46.4 I2C functional description 1936
46.4.1 I2C block diagram 1937
46.4.2 I2C clock requirements 1938
46.4.3 Mode selection 1938
46.4.4 I2C initialization 1939
46.4.5 Software reset 1943
46.4.6 Data transfer 1944
46.4.7 I2C slave mode 1946
46.4.8 I2C master mode 1955
46.4.9 I2C_TIMINGR register configuration examples 1967
46.4.10 SMBus specific features 1968
46.4.11 SMBus initialization 1971
46.4.12 SMBus: I2C_TIMEOUTR register configuration examples 1973
46.4.13 SMBus slave mode 1973
46.4.14 Wakeup from Stop mode on address match 1981
46.4.15 Error conditions 1981
46.4.16 DMA requests 1983
46.4.17 Debug mode 1984

46.5 I2C low-power modes 1984
46.6 I2C interrupts 1985
46.7 I2C registers 1986
46.7.1 I2C2 control register 1 (I2C_CR1) 1986
46.7.2 I2C2 control register 2 (I2C_CR2) 1989
46.7.3 I2C2 own address 1 register (I2C_OAR1) 1992
46.7.4 I2C2 own address 2 register (I2C_OAR2) 1993
46.7.5 I2C2 timing register (I2C_TIMINGR) 1994
46.7.6 I2C2 timeout register (I2C_TIMEOUTR) 1995
46.7.7 I2C2 interrupt and status register (I2C_ISR) 1996
46.7.8 I2C2 interrupt clear register (I2C_ICR)(. 1998
46.7.9 I2C2 PEC register (I2C_PECR) 1999
46.7.10 I2C2 receive data register (I2C_RXDR) 2000
46.7.11 I2C2 transmit data register (I2C_TXDR) 2000
46.7.12 I2C register map 2001

47 Universal synchronous receiver transmitter (USART) /universal asynchronous receiver transmitter (UART) 2003

47.1 USART introduction 2003
47.2 USART main features 2004
47.3 USART extended features 2005
47.4 USART implementation 2005
47.5 USART functional description 2006
47.5.1 USART block diagram 2006
47.5.2 USART signals 2007
47.5.3 USART character description 2008
47.5.4 USART FIFOs and thresholds 2010
47.5.5 USART transmitter 2010
47.5.6 USART receiver 2014
47.5.7 USART baud rate generation 2022
47.5.8 Tolerance of the USART receiver to clock deviation 2023
47.5.9 USART Auto baud rate detection 2024
47.5.10 USART multiprocessor communication 2026
47.5.11 USART Modbus communication 2028
47.5.12 USART parity control 2029
47.5.13 USART LIN (local interconnection network) mode 2030

47.5.14 USART synchronous mode 2032
47.5.15 USART single-wire Half-duplex communication 2036
47.5.16 USART receiver timeout 2036
47.5.17 USART Smartcard mode 2037
47.5.18 USART IrDA SIR ENDEC block 2041
47.5.19 Continuous communication using USART and DMA 2044
47.5.20 RS232 Hardware flow control and RS485 Driver Enable 2046
47.5.21 USART low-power management 2049
47.6 USART interrupts 2052
47.7 USART registers 2055
47.7.1 USART control register 1 (USART_CR1) 2055
47.7.2 USART control register 2 (USART_CR2) 2059
47.7.3 USART control register 3 (USART_CR3) 2062
47.7.4 USART baud rate register (USART_BRR) 2067
47.7.5 USART guard time and prescaler register (USART_GTPR) 2067
47.7.6 USART receiver timeout register (USART_RTOR) 2068
47.7.7 USART request register (USART_RQR) 2069
47.7.8 USART interrupt and status register (USART_ISR) 2070
47.7.9 USART interrupt flag clear register (USART_ICR) 2076
47.7.10 USART receive data register (USART_RDR) 2078
47.7.11 USART transmit data register (USART_TDR) 2078
47.7.12 USART prescaler register (USART_PRESC) 2079
47.7.13 USART register map 2080

48 Low-power universal asynchronous receiver transmitter (LPUART) 2082

48.1 LPUART introduction 2082
48.2 LPUART main features 2083
48.3 LPUART functional description 2084
48.3.1 LPUART block diagram 2084
48.3.2 LPUART signals 2085
48.3.3 LPUART character description 2085
48.3.4 LPUART FIFOs and thresholds 2086
48.3.5 LPUART transmitter 2087
48.3.6 LPUART receiver 2090
48.3.7 LPUART baud rate generation 2094
48.3.8 Tolerance of the LPUART receiver to clock deviation 2096

48.3.9 LPUART multiprocessor communication 2096
48.3.10 LPUART parity control 2098
48.3.11 LPUART single-wire Half-duplex communication 2099
48.3.12 Continuous communication using DMA and LPUART 2100
48.3.13 RS232 Hardware flow control and RS485 Driver Enable 2102
48.3.14 LPUART low-power management 2105
48.4 LPUART interrupts 2108
48.5 LPUART registers 2110
48.5.1 Control register 1 (LPUART_CR1) 2110
48.5.2 Control register 2 (LPUART_CR2) 2113
48.5.3 Control register 3 (LPUART_CR3) 2115
48.5.4 Baud rate register (LPUART_BRR) 2118
48.5.5 Request register (LPUART_RQR) 2119
48.5.6 Interrupt and status register (LPUART_ISR) 2119
48.5.7 Interrupt flag clear register (LPUART_ICR) 2123
48.5.8 Receive data register (LPUART_RDR) 2124
48.5.9 Transmit data register (LPUART_TDR) 2125
48.5.10 Prescaler register (LPUART_PRESC) 2125
48.5.11 LPUART register map 2127

49 Serial peripheral interface (SPI) 2129
49.1 Introduction 2129
49.2 SPI main features 2130
49.3 SPI implementation 2130
49.4 SPI functional description 2131
49.4.1 SPI block diagram 2131
49.4.2 SPI signals 2132
49.4.3 SPI communication general aspects 2132
49.4.4 Communications between one master and one slave 2132
49.4.5 Standard multi-slave communication 2135
49.4.6 Multi-master communication 2138
49.4.7 Slave select (SS) pin management 2138
49.4.8 Communication formats 2142
49.4.9 Configuration of SPI 2144
49.4.10 Procedure for enabling SPI 2145
49.4.11 SPI data transmission and reception procedures 2145

49.4.12 Procedure for disabling the SPI . . . 2149
49.4.13 Data packing . . . 2150
49.4.14 Communication using DMA (direct memory addressing) . . . 2151
49.5 SPI specific modes and control . . . 2153
49.5.1 TI mode . . . 2153
49.5.2 SPI error flags . . . 2153
49.5.3 CRC computation . . . 2155
49.6 Low-power mode management . . . 2157
49.7 SPI wakeup and interrupts . . . 2159
49.8 I2S main features . . . 2160
49.9 I2S functional description . . . 2161
49.9.1 I2S general description . . . 2161
49.9.2 Pin sharing with SPI function . . . 2161
49.9.3 Bits and fields usable in I2S/PCM mode . . . 2162
49.9.4 Slave and master modes . . . 2163
49.9.5 Supported audio protocols . . . 2163
49.9.6 Additional Serial Interface Flexibility . . . 2169
49.9.7 Start-up sequence . . . 2171
49.9.8 Stop sequence . . . 2172
49.9.9 Clock generator . . . 2173
49.9.10 Internal FIFOs . . . 2175
49.9.11 FIFOs status flags . . . 2176
49.9.12 Handling of underrun situation . . . 2177
49.9.13 Handling of overrun situation . . . 2178
49.9.14 Frame error detection . . . 2178
49.9.15 DMA Interface . . . 2180
49.9.16 Programing examples . . . 2181
49.9.17 Slave I2S Philips standard, receive . . . 2183
49.10 I2S wakeup and interrupts . . . 2184
49.11 SPI/I2S registers . . . 2185
49.11.1 SPI/I2S control register 1 (SPI2S_CR1) . . . 2185
49.11.2 SPI control register 2 (SPI_CR2) . . . 2186
49.11.3 SPI configuration register 1 (SPI_CFG1) . . . 2187
49.11.4 SPI configuration register 2 (SPI_CFG2) . . . 2190
49.11.5 SPI/I2S interrupt enable register (SPI2S_IER) . . . 2192
49.11.6 SPI/I2S status register (SPI2S_SR) . . . 2193

49.11.7 SPI/I2S interrupt/status flags clear register (SPI2S_IFCR) 2195
49.11.8 SPI/I2S transmit data register (SPI2S_TXDR) 2196
49.11.9 SPI/I2S receive data register (SPI2S_RXDR) 2197
49.11.10 SPI polynomial register (SPI_CRCPOLY) 2197
49.11.11 SPI transmitter CRC register (SPI_TXCRC) 2198
49.11.12 SPI receiver CRC register (SPI_RXCRC) 2198
49.11.13 SPI underrun data register (SPI_UDRDR) 2199
49.11.14 SPI/I2S configuration register (SPI_I2SCFGR) 2199
49.12 SPI register map and reset values 2202

50 Serial audio interface (SAI) 2204
50.1 Introduction 2204
50.2 SAI main features 2205
50.3 SAI functional description 2206
50.3.1 SAI block diagram 2206
50.3.2 SAI pins and internal signals 2207
50.3.3 Main SAI modes 2207
50.3.4 SAI synchronization mode 2208
50.3.5 Audio data size 2209
50.3.6 Frame synchronization 2210
50.3.7 Slot configuration 2213
50.3.8 SAI clock generator 2215
50.3.9 Internal FIFOs 2217
50.3.10 PDM Interface 2219
50.3.11 AC’97 link controller 2227
50.3.12 SPDIF output 2229
50.3.13 Specific features 2232
50.3.14 Error flags 2236
50.3.15 Disabling the SAI 2239
50.3.16 SAI DMA interface 2239
50.4 SAI interrupts 2240
50.5 SAI registers 2241
50.5.1 Global configuration register (SAI_GCR) 2241
50.5.2 Configuration register 1 (SAI_ACR1) 2241
50.5.3 Configuration register 1 (SAI_BCR1) 2244
50.5.4 Configuration register 2 (SAI_ACR2) 2247

50.5.5 Configuration register 2 (SAI_BCR2) . . . 2249
50.5.6 Frame configuration register (SAI_AFRCR) . . . 2251
50.5.7 Frame configuration register (SAI_BFRCR) . . . 2252
50.5.8 Slot register (SAI_ASLOTR) . . . 2253
50.5.9 Slot register (SAI_BSLOTR) . . . 2254
50.5.10 Interrupt mask register (SAI_AIM) . . . 2255
50.5.11 Interrupt mask register (SAI_BIM) . . . 2257
50.5.12 Status register (SAI_ASR) . . . 2258
50.5.13 Status register (SAI_BSR) . . . 2260
50.5.14 Clear flag register (SAI_ACLRFR) . . . 2262
50.5.15 Clear flag register (SAI_BCLRFR) . . . 2263
50.5.16 Data register (SAI_ADR) . . . 2264
50.5.17 Data register (SAI_BDR) . . . 2265
50.5.18 PDM control register (SAI_PDMCR) . . . 2265
50.5.19 PDM delay register (SAI_PDMDLY) . . . 2266
50.5.20 SAI register map . . . 2269

51 SPDIF receiver interface (SPDIFRX) . . . 2271

51.1 SPDIFRX interface introduction . . . 2271
51.2 SPDIFRX main features . . . 2271
51.3 SPDIFRX functional description . . . 2271
51.3.1 SPDIFRX pins and internal signals . . . 2272
51.3.2 S/PDIF protocol (IEC-60958) . . . 2273
51.3.3 SPDIFRX decoder (SPDIFRX_DC) . . . 2275
51.3.4 SPDIFRX tolerance to clock deviation . . . 2279
51.3.5 SPDIFRX synchronization . . . 2279
51.3.6 SPDIFRX handling . . . 2281
51.3.7 Data reception management . . . 2283
51.3.8 Dedicated control flow . . . 2285
51.3.9 Reception errors . . . 2286
51.3.10 Clocking strategy . . . 2288
51.3.11 DMA Interface . . . 2288
51.3.12 Interrupt Generation . . . 2289
51.3.13 Register protection . . . 2290
51.4 Programming procedures . . . 2291
51.4.1 Initialization phase . . . 2291
51.4.2 Handling of interrupts coming from SPDIFRX . . . 2292

51.4.3 Handling of interrupts coming from DMA 2293
51.5 SPDIFRX interface registers 2293
51.5.1 Control register (SPDIFRX_CR) 2293
51.5.2 Interrupt mask register (SPDIFRX_IMR) 2296
51.5.3 Status register (SPDIFRX_SR) 2297
51.5.4 Interrupt flag clear register (SPDIFRX_IFCR) 2299
51.5.5 Data input register (SPDIFRX_FMT0_DR) 2300
51.5.6 Data input register (SPDIFRX_FMT1_DR) 2301
51.5.7 Data input register (SPDIFRX_FMT2_DR) 2302
51.5.8 Channel status register (SPDIFRX_CSR) 2303
51.5.9 Debug information register (SPDIFRX_DIR) 2303
51.5.10 SPDIFRX interface register map 2305

52 Single Wire Protocol Master Interface (SWPMI) 2306
52.1 Introduction 2306
52.2 SWPMI main features 2307
52.3 SWPMI functional description 2308
52.3.1 SWPMI block diagram 2308
52.3.2 SWPMI pins and internal signals 2308
52.3.3 SWP initialization and activation 2309
52.3.4 SWP bus states 2309
52.3.5 SWPMI_IO (internal transceiver) bypass 2311
52.3.6 SWPMI Bit rate 2311
52.3.7 SWPMI frame handling 2312
52.3.8 Transmission procedure 2312
52.3.9 Reception procedure 2317
52.3.10 Error management 2321
52.3.11 Loopback mode 2323
52.4 SWPMI low-power modes 2323
52.5 SWPMI interrupts 2324
52.6 SWPMI registers 2325
52.6.1 SWPMI Configuration/Control register (SWPMI_CR) 2325
52.6.2 SWPMI Bitrate register (SWPMI_BRR) 2326
52.6.3 SWPMI Interrupt and Status register (SWPMI_ISR) 2327
52.6.4 SWPMI Interrupt Flag Clear register (SWPMI_ICR) 2328
52.6.5 SWPMI Interrupt Enable register (SMPMI_IER) 2329

52.6.6 SWPMI Receive Frame Length register (SWPMI_RFL) 2331
52.6.7 SWPMI Transmit data register (SWPMI_TDR) 2331
52.6.8 SWPMI Receive data register (SWPMI_RDR) 2331
52.6.9 SWPMI Option register (SWPMI_OR) 2332
52.6.10 SWPMI register map and reset value table 2333

53 Management data input/output (MDIOS) 2334
53.1 MDIOS introduction 2334
53.2 MDIOS main features 2334
53.3 MDIOS functional description 2335
53.3.1 MDIOS block diagram 2335
53.3.2 MDIOS pins and internal signals 2335
53.3.3 MDIOS protocol 2336
53.3.4 MDIOS enabling and disabling 2337
53.3.5 MDIOS data 2337
53.3.6 MDIOS APB frequency 2339
53.3.7 Write/read flags and interrupts 2339
53.3.8 MDIOS error management 2339
53.3.9 MDIOS in Stop mode 2340
53.3.10 MDIOS interrupts 2341
53.4 MDIOS registers 2342
53.4.1 MDIOS configuration register (MDIOS_CR) 2342
53.4.2 MDIOS write flag register (MDIOS_WRFR) 2343
53.4.3 MDIOS clear write flag register (MDIOS_CWRFR) 2343
53.4.4 MDIOS read flag register (MDIOS_RDFR) 2344
53.4.5 MDIOS clear read flag register (MDIOS_CRDFR) 2344
53.4.6 MDIOS status register (MDIOS_SR) 2345
53.4.7 MDIOS clear flag register (MDIOS_CLRFR) 2346
53.4.8 MDIOS input data register x (MDIOS_DINRx) 2347
53.4.9 MDIOS output data register x (MDIOS_DOUTRx) 2347
53.4.10 MDIOS register map 2348

54 Secure digital input/output MultiMediaCard interface (SDMMC) . . 2350
54.1 SDMMC main features 2350
54.2 SDMMC bus topology 2350
54.3 SDMMC operation modes 2352

54.4 SDMMC functional description 2353
54.4.1 SDMMC diagram 2354
54.4.2 SDMMC pins and internal signals 2354
54.4.3 General description 2355
54.4.4 SDMMC adapter 2357
54.4.5 SDMMC AHB slave interface 2378
54.4.6 SDMMC AHB master interface 2379
54.4.7 MDMA request generation 2381
54.4.8 AHB and SDMMC_CK clock relation 2381
54.5 Card functional description 2382
54.5.1 SD I/O mode 2382
54.5.2 CMD12 send timing 2390
54.5.3 Sleep (CMD5) 2393
54.5.4 Interrupt mode (Wait-IRQ) 2394
54.5.5 Boot operation 2395
54.5.6 Response R1b handling 2398
54.5.7 Reset and card cycle power 2399
54.6 Hardware flow control 2400
54.7 Ultra-high-speed phase I (UHS-I) voltage switch 2401
54.8 SDMMC registers 2404
54.8.1 SDMMC power control register (SDMMC_POWER) 2404
54.8.2 SDMMC clock control register (SDMMC_CLKCR) 2405
54.8.3 SDMMC argument register (SDMMC_ARGR) 2407
54.8.4 SDMMC command register (SDMMC_CMDR) 2407
54.8.5 SDMMC command response register (SDMMC_RESPCMDR) 2409
54.8.6 SDMMC response x register (SDMMC_RESPxR) 2410
54.8.7 SDMMC data timer register (SDMMC_DTIMER) 2410
54.8.8 SDMMC data length register (SDMMC_DLENR) 2411
54.8.9 SDMMC data control register (SDMMC_DCTRL) 2412
54.8.10 SDMMC data counter register (SDMMC_DCNTR) 2413
54.8.11 SDMMC status register (SDMMC_STAR) 2414
54.8.12 SDMMC interrupt clear register (SDMMC_ICR) 2416
54.8.13 SDMMC mask register (SDMMC_MASKR) 2419
54.8.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) . . . 2421
54.8.15 SDMMC data FIFO register (SDMMC_FIFOR) 2422
54.8.16 SDMMC DMA control register (SDMMC_IDMACTRLR) 2422

54.8.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER) 2423
54.8.18 SDMMC IDMA buffer 0 base address register (SDMMC_IDMABASE0R) 2424
54.8.19 SDMMC IDMA buffer 1 base address register (SDMMC_IDMABASE1R) 2424
54.8.20 SDMMC register map 2425

55 FD controller area network (FDCAN) 2428
55.1 Introduction 2428
55.2 FDCAN main features 2431
55.3 FDCAN functional description 2432
55.3.1 Operating modes 2433
55.3.2 Message RAM 2442
55.3.3 FIFO acknowledge handling 2453
55.3.4 Bit timing 2454
55.3.5 Clock calibration on CAN 2455
55.3.6 TTCAN operations (FDCAN1 only) 2460
55.3.7 TTCAN configuration 2461
55.3.8 Message scheduling 2463
55.3.9 TTCAN gap control 2470
55.3.10 Stop watch 2471
55.3.11 Local time, cycle time, global time, and external clock synchronization 2471
55.3.12 TTCAN error level 2474
55.3.13 TTCAN message handling 2475
55.3.14 TTCAN interrupt and error handling 2478
55.3.15 Level 0 2479
55.3.16 Synchronization to external time schedule 2481
55.3.17 FDCAN Rx buffer and FIFO element 2482
55.3.18 FDCAN Tx buffer element 2484
55.3.19 FDCAN Tx event FIFO element 2486
55.3.20 FDCAN standard message ID filter element 2487
55.3.21 FDCAN extended message ID filter element 2489
55.3.22 FDCAN trigger memory element 2490
55.4 FDCAN registers 2492
55.4.1 FDCAN core release register (FDCAN_CREL) 2492
55.4.2 FDCAN Endian register (FDCAN_ENDN) 2492
55.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 2492

55.4.4 FDCAN test register (FDCAN_TEST) 2493
55.4.5 FDCAN RAM watchdog register (FDCAN_RWD) 2494
55.4.6 FDCAN CC control register (FDCAN_CCCR) 2495
55.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 2497
55.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 2498
55.4.9 FDCAN timestamp counter value register (FDCAN_TSCV) 2498
55.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 2499
55.4.11 FDCAN timeout counter value register (FDCAN_TOCV) 2500
55.4.12 FDCAN error counter register (FDCAN_ECR) 2500
55.4.13 FDCAN protocol status register (FDCAN_PSR) 2501
55.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 2504
55.4.15 FDCAN interrupt register (FDCAN_IR) 2504
55.4.16 FDCAN interrupt enable register (FDCAN_IE) 2507
55.4.17 FDCAN interrupt line select register (FDCAN_ILS) 2510
55.4.18 FDCAN interrupt line enable register (FDCAN_ILE) 2511
55.4.19 FDCAN global filter configuration register (FDCAN_GFC) 2512
55.4.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC) . . 2513
55.4.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC) . . 2513
55.4.22 FDCAN extended ID and mask register (FDCAN_XIDAM) 2514
55.4.23 FDCAN high priority message status register (FDCAN_HPMS) 2515
55.4.24 FDCAN new data 1 register (FDCAN_NDAT1) 2515
55.4.25 FDCAN new data 2 register (FDCAN_NDAT2) 2516
55.4.26 FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C) 2516
55.4.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 2517
55.4.28 FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 2518
55.4.29 FDCAN Rx buffer configuration register (FDCAN_RXBC) 2518
55.4.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C) 2519
55.4.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 2520
55.4.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 2521
55.4.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC) 2521
55.4.34 FDCAN Tx buffer configuration register (FDCAN_TXBC) 2522
55.4.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 2523
55.4.36 FDCAN Tx buffer element size configuration register (FDCAN_TXESC) 2524
55.4.37 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 2525
55.4.38 FDCAN Tx buffer add request register (FDCAN_TXBAR) 2526
55.4.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 2526

55.4.40 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 2527
55.4.41 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 2527
55.4.42 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 2527
55.4.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_ TXBCIE) 2528
55.4.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC) . . . 2528
55.4.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 2529
55.4.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 2530
55.4.47 FDCAN TT trigger memory configuration register (FDCAN_TTTMC) 2530
55.4.48 FDCAN TT reference message configuration register (FDCAN_TTRMC) 2531
55.4.49 FDCAN TT operation configuration register (FDCAN_TTOCF) 2532
55.4.50 FDCAN TT matrix limits register (FDCAN_TTMLM) 2533
55.4.51 FDCAN TUR configuration register (FDCAN_TURCF) 2534
55.4.52 FDCAN TT operation control register (FDCAN_TTOCN) 2536
55.4.53 FDCAN TT global time preset register (FDCAN_TTGTP) 2537
55.4.54 FDCAN TT time mark register (FDCAN_TTTMK) 2538
55.4.55 FDCAN TT interrupt register (FDCAN_TTIR) 2539
55.4.56 FDCAN TT interrupt enable register (FDCAN_TTIE) 2541
55.4.57 FDCAN TT interrupt line select register (FDCAN_TTILS) 2543
55.4.58 FDCAN TT operation status register (FDCAN_TTOST) 2544
55.4.59 FDCAN TUR numerator actual register (FDCAN_TURNA) 2546
55.4.60 FDCAN TT local and global time register (FDCAN_TTLGT) 2547
55.4.61 FDCAN TT cycle time and count register (FDCAN_TTCTC) 2547
55.4.62 FDCAN TT capture time register (FDCAN_TTCPT) 2548
55.4.63 FDCAN TT cycle sync mark register (FDCAN_TTCSM) 2548
55.4.64 FDCAN TT trigger select register (FDCAN_TTTS) 2549
55.4.65 FDCAN register map and reset value table 2550
55.5 CCU registers 2556
55.5.1 Clock calibration unit core release register (FCCAN_CCU_CREL) . . 2556
55.5.2 Calibration configuration register (FCCAN_CCU_CCFG) 2556
55.5.3 Calibration status register (FCCAN_CCU_CSTAT) 2558
55.5.4 Calibration watchdog register (FCCAN_CCU_CWD) 2558
55.5.5 Clock calibration unit interrupt register (FCCAN_CCU_IR) 2559
55.5.6 Clock calibration unit interrupt enable register (FCCAN_CCU_IE) . . 2560
55.5.7 CCU register map and reset value table 2561

56 USB on-the-go high-speed (OTG_HS) 2562

56.1 Introduction 2562

56.2 OTG main features 2563

56.2.1 General features 2564

56.2.2 Host-mode features 2565

56.2.3 Peripheral-mode features 2565

56.3 OTG implementation 2566

56.4 OTG functional description 2566

56.4.1 OTG block diagram 2566

56.4.2 USB OTG pin and internal signals 2568

56.4.3 OTG core 2568

56.4.4 Embedded full speed OTG PHY 2569

56.4.5 High-speed OTG PHY 2569

56.5 OTG dual role device (DRD) 2569

56.5.1 ID line detection 2569

56.5.2 HNP dual role device 2570

56.5.3 SRP dual role device 2570

56.6 USB peripheral 2570

56.6.1 SRP-capable peripheral 2571

56.6.2 Peripheral states 2571

56.6.3 Peripheral endpoints 2572

56.7 USB host 2574

56.7.1 SRP-capable host 2574

56.7.2 USB host states 2575

56.7.3 Host channels 2576

56.7.4 Host scheduler 2577

56.8 SOF trigger 2578

56.8.1 Host SOFs 2579

56.8.2 Peripheral SOFs 2579

56.9 OTG low-power modes 2579

56.10 Dynamic update of the OTG_HFIR register 2580

56.11 USB data FIFOs 2581

56.11.1 Peripheral FIFO architecture 2582

56.11.2 Host FIFO architecture 2583

56.11.3 FIFO RAM allocation 2584

56.12 **OTG_HS** interrupts 2586
56.13 **OTG_HS** control and status registers 2588
56.13.1 CSR memory map 2588
56.14 **OTG_HS** registers 2593
56.14.1 OTG control and status register (OTG_GOTGCTL) 2593
56.14.2 OTG interrupt register (OTG_GOTGINT) 2596
56.14.3 OTG AHB configuration register (OTG_GAHBCFG) 2598
56.14.4 OTG USB configuration register (OTG_GUSBCFG) 2599
56.14.5 OTG reset register (OTG_GRSTCTL) 2603
56.14.6 OTG core interrupt register (OTG_GINTSTS) 2605
56.14.7 OTG interrupt mask register (OTG_GINTMSK) 2609
56.14.8 OTG receive status debug read register (OTG_GRXSTSR) 2612
56.14.9 OTG receive status debug read [alternate] (OTG_GRXSTSR) 2613
56.14.10 OTG status read and pop registers (OTG_GRXSTSP) 2614
56.14.11 OTG status read and pop registers [alternate] (OTG_GRXSTSP) . . 2615
56.14.12 OTG receive FIFO size register (OTG_GRXFSIZ) 2616
56.14.13 OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0) 2616
56.14.14 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS) 2617
56.14.15 OTG general core configuration register (OTG_GCCFG) 2618
56.14.16 OTG core ID register (OTG_CID) 2620
56.14.17 OTG core LPM configuration register (OTG_GLPMCFG) 2620
56.14.18 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ) 2624
56.14.19 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx) 2624
56.14.20 Host-mode registers 2624
56.14.21 OTG host configuration register (OTG_HCFG) 2625
56.14.22 OTG host frame interval register (OTG_HFIR) 2625
56.14.23 OTG host frame number/frame time remaining register (OTG_HFNUM) 2626
56.14.24 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS) 2627
56.14.25 OTG host all channels interrupt register (OTG_HAINT) 2628
56.14.26 OTG host all channels interrupt mask register (OTG_HAINTMSK) 2628

56.14.27 OTG host port control and status register (OTG_HPRT) 2629
56.14.28 OTG host channel x characteristics register (OTG_HCCHARx) 2631
56.14.29 OTG host channel x split control register (OTG_HCSPLTx) 2632
56.14.30 OTG host channel x interrupt register (OTG_HCINTx) 2633
56.14.31 OTG host channel x interrupt mask register (OTG_HCINTMSKx) . . 2634
56.14.32 OTG host channel x transfer size register (OTG_HCTSIZx) 2635
56.14.33 OTG host channel x DMA address register (OTG_HCDMAx) (x = 0..15, where x = Channel number) 2636
56.14.34 Device-mode registers 2636
56.14.35 OTG device configuration register (OTG_DCFG) 2637
56.14.36 OTG device control register (OTG_DCTL) 2638
56.14.37 OTG device status register (OTG_DSTS) 2640
56.14.38 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK) 2641
56.14.39 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK) 2642
56.14.40 OTG device all endpoints interrupt register (OTG_DAINT) 2644
56.14.41 OTG all endpoints interrupt mask register (OTG_DAINTMSK) 2644
56.14.42 OTG device V_{BUS} discharge time register (OTG_DVBUSDIS) 2645
56.14.43 OTG device V_{BUS} pulsing time register (OTG_DVBUSPULSE) 2645
56.14.44 OTG device threshold control register (OTG_DTHRCTL) 2646
56.14.45 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) 2647
56.14.46 OTG device each endpoint interrupt register (OTG_DEACHINT) . . . 2647
56.14.47 OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK) 2648
56.14.48 OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHMSK1) 2648
56.14.49 OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHMSK1) 2649
56.14.50 OTG device IN endpoint x control register (OTG_DIEPCTLx) 2651
56.14.51 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) 2653
56.14.52 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) 2655
56.14.53 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx) . . . 2655
56.14.54 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) 2656

56.14.55 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) . 2656
56.14.56 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) 2657
56.14.57 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) . . 2659
56.14.58 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0) 2661
56.14.59 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx) . 2662
56.14.60 OTG device OUT endpoint x control register (OTG_DOEPCTLx) . . . 2662
56.14.61 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx) . . 2665
56.14.62 OTG power and clock gating control register (OTG_PCGCCTL) . . . 2666
56.14.63 **OTG_HS** register map 2667
56.15 **OTG_HS** programming model 2677
56.15.1 Core initialization 2677
56.15.2 Host initialization 2677
56.15.3 Device initialization 2678
56.15.4 DMA mode 2679
56.15.5 Host programming model 2679
56.15.6 Device programming model 2711
56.15.7 Worst case response time 2729
56.15.8 OTG programming model 2731

57 Ethernet (ETH): media access control (MAC) with DMA controller 2737
57.1 Ethernet introduction 2737
57.2 Ethernet main features 2737
57.2.1 MAC core features 2737
57.2.2 DMA features 2740
57.2.3 Bus interface features 2740
57.3 Ethernet pins and internal signals 2741
57.4 Ethernet architecture 2743
57.4.1 DMA controller 2744
57.4.2 MTL 2750
57.4.3 MAC 2751
57.5 Ethernet functional description: MAC 2756
57.5.1 Double VLAN processing 2756
57.5.2 Source Address and VLAN insertion, replacement, or deletion 2757

57.5.3 Packet filtering . . . 2759
57.5.4 IEEE 1588 timestamps . . . 2766
57.5.5 IPv4 ARP offload . . . 2779
57.5.6 TCP segmentation offload . . . 2779
57.5.7 Loopback . . . 2783
57.5.8 Flow control . . . 2783
57.5.9 Checksum offload engine . . . 2785
57.5.10 MAC management counters . . . 2790
57.5.11 Interrupts generated by the MAC . . . 2790
57.5.12 MAC and MMC register descriptions . . . 2790
57.6 Ethernet functional description: PHY interfaces . . . 2791
57.6.1 Station management agent (SMA) . . . 2791
57.6.2 Media Independent Interface (MII) . . . 2794
57.6.3 Reduced media independent interface (RMII) . . . 2796
57.7 Ethernet low-power modes . . . 2799
57.7.1 Energy Efficient Ethernet . . . 2799
57.7.2 Power management . . . 2800
57.7.3 Power-down and wakeup sequence . . . 2802
57.8 Ethernet interrupts . . . 2803
57.8.1 DMA interrupts . . . 2803
57.8.2 MTL interrupts . . . 2804
57.8.3 MAC Interrupts . . . 2805
57.9 Ethernet programming model . . . 2806
57.9.1 DMA initialization . . . 2806
57.9.2 MTL initialization . . . 2807
57.9.3 MAC initialization . . . 2807
57.9.4 Performing normal receive and transmit operation . . . 2808
57.9.5 Stopping and starting transmission . . . 2809
57.9.6 Programming guidelines for MII link state transitions . . . 2809
57.9.7 Programming guidelines for IEEE 1588 timestamping . . . 2810
57.9.8 Programming guidelines for Energy Efficient Ethernet (EEE) . . . 2811
57.9.9 Programming guidelines for flexible pulse-per-second (PPS) output 2812
57.9.10 Programming guidelines for TSO . . . 2814
57.9.11 Programming guidelines to perform VLAN filtering on the receive . . 2814
57.10 Descriptors . . . 2815
57.10.1 Descriptor overview . . . 2815

57.10.2 Descriptor structure . . . 2815
57.10.3 Transmit descriptor . . . 2817
57.10.4 Receive descriptor . . . 2829
57.11 Ethernet registers . . . 2841
57.11.1 Ethernet registers maps . . . 2841
57.11.2 Ethernet DMA registers . . . 2841
57.11.3 Ethernet MTL registers . . . 2867
57.11.4 Ethernet MAC and MMC registers . . . 2879

58 HDMI-CEC controller (HDMI-CEC) . . . 2978
58.1 Introduction . . . 2978
58.2 HDMI-CEC controller main features . . . 2978
58.3 HDMI-CEC functional description . . . 2979
58.3.1 HDMI-CEC pin and internal signals . . . 2979
58.3.2 HDMI-CEC block diagram . . . 2980
58.3.3 Message description . . . 2980
58.3.4 Bit timing . . . 2981
58.4 Arbitration . . . 2981
58.4.1 SFT option bit . . . 2983
58.5 Error handling . . . 2983
58.5.1 Bit error . . . 2983
58.5.2 Message error . . . 2984
58.5.3 Bit Rising Error (BRE) . . . 2984
58.5.4 Short Bit Period Error (SBPE) . . . 2984
58.5.5 Long Bit Period Error (LBPE) . . . 2984
58.5.6 Transmission Error Detection (TXERR) . . . 2987
58.6 HDMI-CEC interrupts . . . 2988
58.7 HDMI-CEC registers . . . 2989
58.7.1 CEC control register (CEC_CR) . . . 2989
58.7.2 CEC configuration register (CEC_CFGR) . . . 2990
58.7.3 CEC Tx data register (CEC_TXDR) . . . 2993
58.7.4 CEC Rx data register (CEC_RXDR) . . . 2993
58.7.5 CEC Interrupt and Status Register (CEC_ISR) . . . 2993
58.7.6 CEC interrupt enable register (CEC_IER) . . . 2995
58.7.7 HDMI-CEC register map . . . 2997

59 Debug infrastructure 2998
59.1 Introduction 2998
59.2 Debug infrastructure features 2999
59.3 Debug infrastructure functional description 2999
59.3.1 Debug infrastructure block diagram 2999
59.3.2 Debug infrastructure pins and internal signals 3000
59.3.3 Debug infrastructure powering, clocking and reset 3001
59.4 Debug access port functional description 3003
59.4.1 Serial-wire and JTAG debug port (SWJ-DP) 3003
59.4.2 Access ports 3017
59.5 Trace and debug subsystem functional description 3023
59.5.1 System ROM tables 3023
59.5.2 Global timestamp generator (TSG) 3031
59.5.3 Cross trigger interfaces (CTI) and matrix (CTM) 3038
59.5.4 Trace funnel (CSTF) 3058
59.5.5 Embedded trace FIFO (ETF) 3068
59.5.6 Trace port interface unit (TPIU) 3090
59.5.7 Serial wire output (SWO) and SWO trace funnel (SWTF) 3108
59.5.8 Microcontroller debug unit (DBGMCU) 3131
59.6 Cortex-M7 debug functional description 3138
59.6.1 Cortex-M7 ROM tables 3139
59.6.2 Cortex-M7 data watchpoint and trace unit (DWT) 3151
59.6.3 Cortex-M7 instrumentation trace macrocell (ITM) 3164
59.6.4 Cortex-M7 breakpoint unit (FPB) 3173
59.6.5 Cortex-M7 embedded trace macrocell (ETM) 3181
59.6.6 Cortex-M7 cross trigger interface (CTI) 3213
59.7 References for debug infrastructure 3213

60 Device electronic signature 3214
60.1 Unique device ID register (96 bits) 3214
60.2 Flash size 3215
60.3 Package data register 3215

61 Revision history 3216

List of tables

Table 1. Availability of security features . . . 98
Table 2. Bus-master-to-bus-slave interconnect . . . 99
Table 3. ASIB configuration . . . 105
Table 4. AMIB configuration . . . 106
Table 5. AXI interconnect register map and reset values . . . 116
Table 6. Memory map and default device memory area attributes . . . 124
Table 7. Register boundary addresses . . . 126
Table 8. Boot modes . . . 131
Table 9. FLASH internal input/output signals . . . 135
Table 10. Flash memory organization on STM32H750xB devices . . . 138
Table 11. Flash memory organization on STM32H743/753 devices . . . 138
Table 12. FLASH recommended number of wait states and programming delay . . . 144
Table 13. FLASH parallelism parameter . . . 148
Table 14. FLASH AXI interface memory map vs swapping option . . . 154
Table 15. Flash register map vs swapping option . . . 156
Table 16. Option byte organization . . . 161
Table 17. Flash interface register protection summary . . . 166
Table 18. RDP value vs readout protection level . . . 168
Table 19. Protection vs RDP Level . . . 169
Table 20. RDP transition and its effects . . . 170
Table 21. Effect of low-power modes on the embedded Flash memory . . . 174
Table 22. Flash interrupt request . . . 182
Table 23. Register map and reset value table . . . 224
Table 24. List of preferred terms . . . 229
Table 25. Flash protection mechanisms . . . 231
Table 26. Secure user software selection use-cases . . . 237
Table 27. Summary of Flash protected areas access rights . . . 238
Table 28. PWR input/output signals connected to package pins or balls . . . 241
Table 29. PWR internal input/output signals . . . 241
Table 30. Supply configuration control . . . 245
Table 31. Low-power mode summary . . . 262
Table 32. PDDS_Dn low-power mode control . . . 264
Table 33. Low-power exit mode flags . . . 266
Table 34. CSleep mode . . . 275
Table 35. CStop mode . . . 276
Table 36. DStop mode overview . . . 277
Table 37. DStop mode . . . 278
Table 38. Stop mode operation . . . 279
Table 39. Stop mode . . . 280
Table 40. DStandby mode . . . 281
Table 41. Standby and Stop flags . . . 283
Table 42. Standby mode . . . 283
Table 43. Power control register map and reset values . . . 293
Table 44. BDMA and DMAMUX2 interconnection . . . 301
Table 45. BDMA and DMAMUX2 initialization sequence (DMAMUX2_INIT) . . . 301
Table 46. LPUART1 Initial programming (LPUART1_INIT) . . . 303
Table 47. LPUART1 Initial programming (LPUART1_Start) . . . 303
Table 48. RCC input/output signals connected to package pins or balls . . . 307

Table 49. RCC internal input/output signals . . . 308
Table 50. Reset distribution summary . . . 311
Table 51. Reset source identification (RCC_RSR) . . . 313
Table 52. Ratio between clock timer and pclk . . . 331
Table 53. STOPWUCK and STOPKERWUCK description . . . 332
Table 54. HSIKERON and CSIKERON behavior . . . 333
Table 55. Kernel clock distribution overview . . . 335
Table 56. System states overview . . . 349
Table 57. Peripheral clock enabling for D1 and D2 peripherals . . . 354
Table 58. Peripheral clock enabling for D3 peripherals . . . 355
Table 59. Interrupt sources and control . . . 359
Table 60. RCC_RSR address offset and reset value . . . 426
Table 61. RCC_AHB3ENR address offset and reset value . . . 428
Table 62. RCC_AHB1ENR address offset and reset value . . . 430
Table 63. RCC_AHB2ENR address offset and reset value . . . 432
Table 64. RCC_AHB4ENR address offset and reset value . . . 434
Table 65. RCC_APB3ENR address offset and reset value . . . 437
Table 66. RCC_APB1ENR address offset and reset value . . . 438
Table 67. RCC_APB1ENR address offset and reset value . . . 442
Table 68. RCC_APB2ENR address offset and reset value . . . 444
Table 69. RCC_APB4ENR address offset and reset value . . . 447
Table 70. RCC_AHB3LPENR address offset and reset value . . . 450
Table 71. RCC_AHB1LPENR address offset and reset value . . . 452
Table 72. RCC_AHB2LPENR address offset and reset value . . . 454
Table 73. RCC_AHB4LPENR address offset and reset value . . . 456
Table 74. RCC_APB3LPENR address offset and reset value . . . 459
Table 75. RCC_APB1LLPENR address offset and reset value . . . 460
Table 76. RCC_APB1HLPENR address offset and reset value . . . 464
Table 77. RCC_APB2LPENR address offset and reset value . . . 466
Table 78. RCC_APB4LPENR address offset and reset value . . . 469
Table 79. RCC register map and reset values . . . 472
Table 80. CRS internal input/output signals . . . 483
Table 81. Effect of low-power modes on CRS . . . 487
Table 82. Interrupt control bits . . . 487
Table 83. CRS register map and reset values . . . 493
Table 84. HSEM internal input/output signals . . . 495
Table 85. Authorized AHB bus master ID . . . 501
Table 86. HSEM register map and reset values . . . 507
Table 87. Port bit configuration table . . . 510
Table 88. GPIO register map and reset values . . . 524
Table 89. SYSCFG register map and reset values . . . 548
Table 90. Peripherals interconnect matrix (D2 domain) . . . 552
Table 91. Peripherals interconnect matrix (D3 domain) . . . 553
Table 92. Peripherals interconnect matrix details . . . 554
Table 93. EXTI wakeup inputs . . . 571
Table 94. EXTI pending requests clear inputs . . . 574
Table 95. MDMA . . . 576
Table 96. DMAMUX1, DMA1 and DMA2 connections . . . 578
Table 97. DMAMUX2 and BDMA connections . . . 583
Table 98. MDMA internal input/output signals . . . 588
Table 99. MDMA interrupt requests . . . 595
Table 100. MDMA register map and reset values . . . 613

Table 101. DMA internal input/output signals . . . 617
Table 102. Source and destination address . . . 619
Table 103. Source and destination address registers in double-buffer mode (DBM = 1) . . . 624
Table 104. Packing/unpacking and endian behavior (bit PINC = MINC = 1) . . . 625
Table 105. Restriction on NDT versus PSIZE and MSIZE . . . 626
Table 106. FIFO threshold configurations . . . 628
Table 107. Possible DMA configurations . . . 632
Table 108. DMA interrupt requests . . . 634
Table 109. DMA register map and reset values . . . 645
Table 110. BDMA implementation . . . 649
Table 111. BDMA internal input/output signals . . . 651
Table 112. Programmable data width and endian behavior (when PINC = MINC = 1) . . . 656
Table 113. BDMA interrupt requests . . . 658
Table 114. BDMA register map and reset values . . . 668
Table 115. DMAMUX1 and DMAMUX2 instantiation . . . 672
Table 116. DMAMUX1: assignment of multiplexer inputs to resources . . . 672
Table 117. DMAMUX1: assignment of trigger inputs to resources . . . 673
Table 118. DMAMUX1: assignment of synchronization inputs to resources . . . 674
Table 119. DMAMUX2: assignment of multiplexer inputs to resources . . . 674
Table 120. DMAMUX2: assignment of trigger inputs to resources . . . 675
Table 121. DMAMUX2: assignment of synchronization inputs to resources . . . 675
Table 122. DMAMUX signals . . . 677
Table 123. DMAMUX interrupts . . . 681
Table 124. DMAMUX register map and reset values . . . 690
Table 125. DMA2D internal input/output signals . . . 694
Table 126. Supported color mode in input . . . 695
Table 127. Data order in memory . . . 697
Table 128. Alpha mode configuration . . . 697
Table 129. Supported CLUT color mode . . . 698
Table 130. CLUT data order in memory . . . 698
Table 131. Supported color mode in output . . . 699
Table 132. Data order in memory . . . 700
Table 133. Standard data order in memory . . . 700
Table 134. Output FIFO byte reordering steps . . . 701
Table 135. MCU order in memory . . . 706
Table 136. DMA2D interrupt requests . . . 708
Table 137. DMA2D register map and reset values . . . 727
Table 138. NVIC . . . 730
Table 139. EXTI Event input configurations and register control . . . 740
Table 140. Configurable Event input Asynchronous Edge detector reset . . . 742
Table 141. EXTI Event input mapping . . . 747
Table 142. Masking functionality . . . 749
Table 143. Asynchronous interrupt/event controller register map and reset values . . . 767
Table 144. CRC internal input/output signals . . . 771
Table 145. CRC register map and reset values . . . 775
Table 146. FMC pins . . . 779
Table 147. FMC bank mapping options . . . 782
Table 148. NOR/PSRAM bank selection . . . 783
Table 149. NOR/PSRAM External memory address . . . 783
Table 150. NAND memory mapping and timing registers . . . 783
Table 151. NAND bank selection . . . 784
Table 152. SDRAM bank selection . . . 784

Table 153. SDRAM address mapping . . . 784
Table 154. SDRAM address mapping with 8-bit data bus width . . . 785
Table 155. SDRAM address mapping with 16-bit data bus width . . . 786
Table 156. SDRAM address mapping with 32-bit data bus width . . . 787
Table 157. Programmable NOR/PSRAM access parameters . . . 788
Table 158. Non-multiplexed I/O NOR Flash memory . . . 789
Table 159. 16-bit multiplexed I/O NOR Flash memory . . . 789
Table 160. Non-multiplexed I/Os PSRAM/SRAM . . . 790
Table 161. 16-Bit multiplexed I/O PSRAM . . . 790
Table 162. NOR Flash/PSRAM: Example of supported memories and transactions . . . 791
Table 163. FMC_BCRx bit fields . . . 794
Table 164. FMC_BTRx bit fields . . . 795
Table 165. FMC_BCRx bit fields . . . 797
Table 166. FMC_BTRx bit fields . . . 798
Table 167. FMC_BWTRx bit fields . . . 798
Table 168. FMC_BCRx bit fields . . . 800
Table 169. FMC_BTRx bit fields . . . 801
Table 170. FMC_BWTRx bit fields . . . 801
Table 171. FMC_BCRx bit fields . . . 803
Table 172. FMC_BTRx bit fields . . . 804
Table 173. FMC_BWTRx bit fields . . . 804
Table 174. FMC_BCRx bit fields . . . 806
Table 175. FMC_BTRx bit fields . . . 806
Table 176. FMC_BWTRx bit fields . . . 807
Table 177. FMC_BCRx bit fields . . . 809
Table 178. FMC_BTRx bit fields . . . 809
Table 179. FMC_BCRx bit fields . . . 815
Table 180. FMC_BTRx bit fields . . . 815
Table 181. FMC_BCRx bit fields . . . 816
Table 182. FMC_BTRx bit fields . . . 817
Table 183. Programmable NAND Flash access parameters . . . 827
Table 184. 8-bit NAND Flash memory . . . 827
Table 185. 16-bit NAND Flash memory . . . 828
Table 186. Supported memories and transactions . . . 828
Table 187. ECC result relevant bits . . . 838
Table 188. SDRAM signals . . . 839
Table 189. FMC register map . . . 857
Table 190. QUADSPI internal signals . . . 861
Table 191. QUADSPI pins . . . 861
Table 192. QUADSPI interrupt requests . . . 875
Table 193. QUADSPI register map and reset values . . . 888
Table 194. DLYB internal input/output signals . . . 890
Table 195. Delay block control . . . 890
Table 196. DLYB register map and reset values . . . 893
Table 197. ADC internal input/output signals . . . 898
Table 198. ADC input/output pins . . . 898
Table 199. Configuring the trigger polarity for regular external triggers . . . 919
Table 200. Configuring the trigger polarity for injected external triggers . . . 919
Table 201. ADC1, ADC2 and ADC3 - External triggers for regular channels . . . 920
Table 202. ADC1, ADC2 and ADC3 - External triggers for injected channels . . . 921
Table 203. TSAR timings depending on resolution . . . 934
Table 204. Offset computation versus data resolution . . . 937

Table 205. 16-bit data formats . . . 940
Table 206. Numerical examples for 16-bit format (bold indicates saturation) . . . 940
Table 207. Analog watchdog channel selection . . . 948
Table 208. Analog watchdog 1,2,3 comparison . . . 949
Table 209. Oversampler operating modes summary . . . 957
Table 210. ADC interrupts per each ADC . . . 976
Table 211. DELAY bits versus ADC resolution . . . 1014
Table 212. ADC global register map . . . 1016
Table 213. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) . . . 1016
Table 214. ADC register map and reset values (master and slave ADC common registers) offset =0x300) . . . 1019
Table 215. DAC implementation . . . 1021
Table 216. DAC input/output pins . . . 1023
Table 217. DAC internal input/output signals . . . 1023
Table 218. DAC trigger selection . . . 1026
Table 219. Sample and refresh timings . . . 1031
Table 220. Channel output modes summary . . . 1032
Table 221. Effect of low-power modes on DAC . . . 1039
Table 222. DAC interrupts . . . 1039
Table 223. DAC register map and reset values . . . 1056
Table 224. VREF buffer modes . . . 1058
Table 225. VREFBUF register map and reset values . . . 1060
Table 226. COMP input/output internal signals . . . 1063
Table 227. COMP input/output pins . . . 1063
Table 228. COMP1_OUT assignment to GPIOs . . . 1066
Table 229. COMP2_OUT assignment to GPIOs . . . 1066
Table 230. Comparator behavior in the low-power modes . . . 1068
Table 231. Interrupt control bits . . . 1068
Table 232. Interrupt control bits . . . 1069
Table 233. COMP register map and reset values . . . 1077
Table 234. Operational amplifier possible connections . . . 1079
Table 235. Operating modes and calibration . . . 1086
Table 236. Effect of low-power modes on the OPAMP . . . 1088
Table 237. OPAMP register map and reset values . . . 1095
Table 238. DFSDM1 implementation . . . 1098
Table 239. DFSDM external pins . . . 1100
Table 240. DFSDM internal signals . . . 1100
Table 241. DFSDM triggers connection . . . 1100
Table 242. DFSDM break connection . . . 1101
Table 243. Filter maximum output resolution (peak data values from filter output) for some FOSR values . . . 1115
Table 244. Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and Sinc3 filter type (largest data) . . . 1116
Table 245. DFSDM interrupt requests . . . 1124
Table 246. DFSDM register map and reset values . . . 1144
Table 247. DCMI internal signals . . . 1156
Table 248. DCMI external signals . . . 1156
Table 249. Positioning of captured data bytes in 32-bit words (8-bit width) . . . 1157
Table 250. Positioning of captured data bytes in 32-bit words (10-bit width) . . . 1157
Table 251. Positioning of captured data bytes in 32-bit words (12-bit width) . . . 1158
Table 252. Positioning of captured data bytes in 32-bit words (14-bit width) . . . 1158

Table 253. Data storage in monochrome progressive video format . . . 1164
Table 254. Data storage in RGB progressive video format . . . 1165
Table 255. Data storage in YCbCr progressive video format . . . 1165
Table 256. Data storage in YCbCr progressive video format - Y extraction mode . . . 1166
Table 257. DCMI interrupts . . . 1166
Table 258. DCMI register map and reset values . . . 1179
Table 259. LCD-TFT internal signals . . . 1181
Table 260. LTDC pins and signal interface . . . 1182
Table 261. Clock domain for each register . . . 1183
Table 262. LTDC register access and update durations . . . 1183
Table 263. Pixel data mapping versus color format . . . 1187
Table 264. LTDC interrupt requests . . . 1191
Table 265. LTDC register map and reset values . . . 1212
Table 266. JPEG internal signals . . . 1216
Table 267. JPEG codec interrupt requests . . . 1220
Table 268. JPEG codec register map and reset values . . . 1228
Table 269. RNG internal input/output signals . . . 1231
Table 270. RNG interrupt requests . . . 1238
Table 271. RNG register map and reset map . . . 1243
Table 272. CRYP internal input/output signals . . . 1247
Table 273. Counter mode initialization vector . . . 1273
Table 274. GCM last block definition . . . 1276
Table 275. GCM mode IV registers initialization . . . 1276
Table 276. CCM mode IV registers initialization . . . 1283
Table 277. DES/TDES data swapping feature . . . 1289
Table 278. AES data swapping feature . . . 1291
Table 279. Key registers CRYP_KxR/LR endianness (TDES K1/2/3 and AES 128/192/256-bit keys) . . . 1291
Table 280. Initialization vector registers CRYP_IVxR endianness . . . 1292
Table 281. Cryptographic processor configuration for memory-to-peripheral DMA transfers . . . 1293
Table 282. Cryptographic processor configuration for peripheral to memory DMA transfers . . . 1294
Table 283. CRYP interrupt requests . . . 1296
Table 284. Processing time (in clock cycle) for ECB, CBC and CTR per 128-bit block . . . 1297
Table 285. Processing time (in clock cycle) for GCM and CCM per 128-bit block . . . 1297
Table 286. CRYP register map and reset values . . . 1311
Table 287. HASH internal input/output signals . . . 1315
Table 288. Hash processor outputs . . . 1319
Table 289. HASH interrupt requests . . . 1326
Table 290. Processing time (in clock cycle) . . . 1326
Table 291. HASH register map and reset values . . . 1338
Table 292. HRTIM Input/output summary . . . 1343
Table 293. Timer resolution and min. PWM frequency for f_{HRTIM} = 400 MHz . . . 1345
Table 294. Period and Compare registers min and max values . . . 1347
Table 295. Timer operating modes . . . 1348
Table 296. Events mapping across Timer A to E . . . 1353
Table 297. Deadtime resolution and max absolute values . . . 1361
Table 298. External events mapping and associated features . . . 1368
Table 299. Output set/reset latency and jitter vs external event operating mode . . . 1369
Table 300. Filtering signals mapping per time . . . 1372
Table 301. Windowing signals mapping per timer (EEFLTR[3:0] = 1111) . . . 1374

Table 302. HRTIM preloadable control registers and associated update sources . . . 1383
Table 303. Update enable inputs and sources . . . 1384
Table 304. Master timer update event propagation . . . 1386
Table 305. TIMx update event propagation . . . 1386
Table 306. Reset events able to generate an update . . . 1387
Table 307. Update event propagation for a timer reset . . . 1388
Table 308. Output state programming, x= A..E, y = 1 or 2 . . . 1389
Table 309. Timer output programming for burst mode . . . 1392
Table 310. Burst mode clock sources from general purpose timer . . . 1394
Table 311. Fault inputs . . . 1402
Table 312. Sampling rate and filter length vs FLTFxF[3:0] and clock setting . . . 1403
Table 313. Effect of sync event vs timer operating modes . . . 1408
Table 314. HRTIM interrupt summary . . . 1414
Table 315. HRTIM DMA request summary . . . 1415
Table 316. RTIM global register map . . . 1505
Table 317. HRTIM Register map and reset values: Master timer . . . 1505
Table 318. HRTIM Register map and reset values: TIMx (x= A..E) . . . 1507
Table 319. HRTIM Register map and reset values: Common functions . . . 1511
Table 320. Behavior of timer outputs versus BRK/BRK2 inputs . . . 1555
Table 321. Counting direction versus encoder signals . . . 1562
Table 322. TIMx internal trigger connection . . . 1579
Table 323. Output control bits for complementary OCx and OCxN channels with break feature . . . 1593
Table 324. TIM1 register map and reset values . . . 1613
Table 325. TIM8 register map and reset values . . . 1616
Table 326. Counting direction versus encoder signals . . . 1652
Table 327. TIMx internal trigger connection . . . 1670
Table 328. Output control bit for standard OCx channels . . . 1680
Table 329. TIM2/TIM3/TIM4/TIM5 register map and reset values . . . 1691
Table 330. TIMx internal trigger connection . . . 1722
Table 331. Output control bit for standard OCx channels . . . 1730
Table 332. TIM12 register map and reset values . . . 1733
Table 333. Output control bit for standard OCx channels . . . 1741
Table 334. TIM13/TIM14 register map and reset values . . . 1744
Table 335. TIMx Internal trigger connection . . . 1787
Table 336. Output control bits for complementary OCx and OCxN channels with break feature (TIM15) . . . 1797
Table 337. TIM15 register map and reset values . . . 1806
Table 338. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) . . . 1817
Table 339. TIM16/TIM17 register map and reset values . . . 1828
Table 340. TIM6/TIM7 register map and reset values . . . 1842
Table 341. STM32H743/753 and STM32H750 LPTIM features . . . 1843
Table 342. LPTIM input/output pins . . . 1846
Table 343. LPTIM internal signals . . . 1846
Table 344. LPTIM1 external trigger connection . . . 1846
Table 345. LPTIM2 external trigger connection . . . 1847
Table 346. LPTIM3 external trigger connection . . . 1847
Table 347. LPTIM4 external trigger connection . . . 1847
Table 348. LPTIM5 external trigger connection . . . 1848
Table 349. LPTIM1 input 1 connection . . . 1848
Table 350. LPTIM1 input 2 connection . . . 1848
Table 351. LPTIM2 input 1 connection . . . 1848

Table 352. LPTIM2 input 2 connection . . . 1849
Table 353. LPTIM3 input 1 connection . . . 1849
Table 354. Prescaler division ratios . . . 1850
Table 355. Encoder counting scenarios . . . 1857
Table 356. Effect of low-power modes on the LPTIM . . . 1858
Table 357. Interrupt events . . . 1859
Table 358. LPTIM register map and reset values . . . 1871
Table 359. WWDG internal input/output signals . . . 1874
Table 360. WWDG register map and reset values . . . 1879
Table 361. IWDG internal input/output signals . . . 1881
Table 362. IWDG register map and reset values . . . 1888
Table 363. RTC pins and internal signals . . . 1892
Table 364. RTC pin PC13 configuration . . . 1893
Table 365. RTC_OUT mapping . . . 1894
Table 366. RTC functions over modes . . . 1894
Table 367. Effect of low-power modes on RTC . . . 1907
Table 368. Interrupt control bits . . . 1908
Table 369. RTC register map and reset values . . . 1933
Table 370. STM32H743/753 and STM32H750 I2C implementation . . . 1936
Table 371. Comparison of analog vs. digital filters . . . 1939
Table 372. I2C-SMBUS specification data setup and hold times . . . 1942
Table 373. I2C configuration . . . 1946
Table 374. I2C-SMBUS specification clock timings . . . 1957
Table 375. Examples of timing settings for fI2CCLK = 8 MHz . . . 1967
Table 376. Examples of timings settings for fI2CCLK = 16 MHz . . . 1967
Table 377. Examples of timings settings for fI2CCLK = 48 MHz . . . 1968
Table 378. SMBus timeout specifications . . . 1970
Table 379. SMBUS with PEC configuration . . . 1972
Table 380. Examples of TIMEOUTA settings for various i2c_ker_ck frequencies (max $t_{TIMEOUT}$ = 25 ms) . . . 1973
Table 381. Examples of TIMEOUTB settings for various i2c_ker_ck frequencies . . . 1973
Table 382. Examples of TIMEOUTA settings for various i2c_ker_ck frequencies (max t_{IDLE} = 50 µs) . . . 1973
Table 383. Effect of low-power modes on the I2C . . . 1984
Table 384. I2C Interrupt requests . . . 1985
Table 385. I2C register map and reset values . . . 2001
Table 386. USART/LPUART features . . . 2005
Table 387. Noise detection from sampled data . . . 2020
Table 388. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 2024
Table 389. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 2024
Table 390. USART frame formats . . . 2029
Table 391. USART interrupt requests . . . 2052
Table 392. USART register map and reset values . . . 2080
Table 393. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32,768 KHz . . . 2095
Table 394. Error calculation for programmed baud rates at fCK = 100 MHz . . . 2095
Table 395. Tolerance of the LPUART receiver . . . 2096
Table 397. LPUART interrupt requests . . . 2108
Table 398. LPUART register map and reset values . . . 2127
Table 399. STM32H7xx SPI features . . . 2130
Table 400. SPI wakeup and interrupt requests . . . 2159
Table 401. Bit fields usable in PCM/I2S mode . . . 2162
Table 402. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 2170

Table 403. Serial data line swapping . . . 2170
Table 404. CLKGEN programming examples for usual I2S frequencies . . . 2175
Table 405. I2S interrupt requests . . . 2184
Table 406. SPI register map and reset values . . . 2202
Table 407. SAI internal input/output signals . . . 2207
Table 408. SAI input/output pins . . . 2207
Table 409. External synchronization selection . . . 2209
Table 410. Clock generator programming examples . . . 2217
Table 411. TDM settings . . . 2224
Table 412. Allowed TDM frame configuration . . . 2226
Table 413. SOPD pattern . . . 2230
Table 414. Parity bit calculation . . . 2230
Table 415. Audio sampling frequency versus symbol rates . . . 2231
Table 416. SAI interrupt sources . . . 2240
Table 417. SAI register map and reset values . . . 2269
Table 418. SPDIFRX internal input/output signals . . . 2272
Table 419. SPDIFRX pins . . . 2272
Table 420. Transition sequence for preamble . . . 2278
Table 421. Minimum spdifrx_ker_ck frequency versus audio sampling rate . . . 2288
Table 422. Bit field property versus SPDIFRX state . . . 2290
Table 423. SPDIFRX interface register map and reset values . . . 2305
Table 424. SWPMI input/output signals connected to package pins or balls . . . 2308
Table 425. SWPMI internal input/output signals . . . 2309
Table 426. Effect of low-power modes on SWPMI . . . 2323
Table 427. Interrupt control bits . . . 2324
Table 428. Buffer modes selection for transmission/reception . . . 2326
Table 429. SWPMI register map and reset values . . . 2333
Table 430. MDIOS input/output signals connected to package pins or balls . . . 2335
Table 431. MDIOS internal input/output signals . . . 2335
Table 432. Interrupt control bits . . . 2341
Table 433. MDIOS register map and reset values . . . 2348
Table 434. SDMMC operation modes SD & SDIO . . . 2352
Table 435. SDMMC operation modes e•MMC . . . 2353
Table 436. SDMMC internal input/output signals . . . 2354
Table 437. SDMMC pins . . . 2354
Table 438. SDMMC Command and data phase selection . . . 2356
Table 439. Command token format . . . 2362
Table 440. Short response with CRC token format . . . 2363
Table 441. Short response without CRC token format . . . 2363
Table 442. Long response with CRC token format . . . 2363
Table 443. Specific Commands overview . . . 2364
Table 444. Command path status flags . . . 2365
Table 445. Command path error handling . . . 2365
Table 446. Data token format . . . 2373
Table 447. Data path status flags and clear bits . . . 2373
Table 448. Data path error handling . . . 2375
Table 449. Data FIFO access . . . 2376
Table 450. Transmit FIFO status flags . . . 2377
Table 451. Receive FIFO status flags . . . 2378
Table 452. SDMMC connections to MDMA . . . 2381
Table 453. AHB and SDMMC_CK clock frequency relation . . . 2381
Table 454. SDIO special operation control . . . 2382

Table 455. 4-bit mode Start, interrupt, and CRC-status Signaling detection . . . 2386
Table 456. CMD12 use cases . . . 2390
Table 457. Response type and SDMMC_RESPxR registers . . . 2410
Table 458. SDMMC register map . . . 2425
Table 459. CAN subsystem I/O signals . . . 2428
Table 460. CAN subsystem I/O pins . . . 2429
Table 461. DLC coding in FDCAN . . . 2436
Table 462. Example of filter configuration for Rx buffers . . . 2448
Table 463. Example of filter configuration for Debug messages . . . 2449
Table 464. Possible configurations for frame transmission . . . 2449
Table 465. Tx buffer/FIFO - queue element size . . . 2450
Table 466. First byte of level 1 reference message . . . 2460
Table 467. First four bytes of level 2 reference message . . . 2461
Table 468. First four bytes of level 0 reference message . . . 2461
Table 469. TUR configuration example . . . 2462
Table 470. System matrix, Node A . . . 2467
Table 471. Trigger list, Node A . . . 2468
Table 472. Number of data bytes transmitted with a reference message . . . 2475
Table 473. Rx buffer and FIFO element . . . 2482
Table 474. Rx buffer and FIFO element description . . . 2482
Table 475. Tx buffer and FIFO element . . . 2484
Table 476. Tx buffer element description . . . 2484
Table 477. Tx Event FIFO element . . . 2486
Table 478. Tx Event FIFO element description . . . 2486
Table 479. Standard message ID filter element . . . 2487
Table 480. Standard message ID filter element field description . . . 2488
Table 481. Extended message ID filter element . . . 2489
Table 482. Extended message ID filter element field description . . . 2489
Table 483. Trigger memory element . . . 2490
Table 484. Trigger memory element description . . . 2490
Table 485. FDCAN register map and reset values . . . 2550
Table 486. CCU register map and reset values . . . 2561
Table 487. OTG_HS speeds supported . . . 2563
Table 488. OTG implementation . . . 2566
Table 489. OTG_FS input/output pins . . . 2568
Table 490. OTG_HS input/output pins . . . 2568
Table 491. OTG_HS input/output signals . . . 2568
Table 492. Compatibility of STM32 low power modes with the OTG . . . 2579
Table 493. Core global control and status registers (CSRs) . . . 2588
Table 494. Host-mode control and status registers (CSRs) . . . 2589
Table 495. Device-mode control and status registers . . . 2590
Table 496. Data FIFO (DFIFO) access register map . . . 2592
Table 497. Power and clock gating control and status registers . . . 2593
Table 498. TRDT values . . . 2602
Table 499. TRDT values (HS) . . . 2602
Table 500. Minimum duration for soft disconnect . . . 2640
Table 501. OTG_HS register map and reset values . . . 2667
Table 502. Ethernet peripheral pins . . . 2741
Table 503. Ethernet internal input/output signals . . . 2742
Table 504. Double VLAN processing features in Tx path . . . 2756
Table 505. Double VLAN processing in Rx path . . . 2757
Table 506. VLAN insertion or replacement based on VLTI bit . . . 2758

Table 507. Destination address filtering . . . 2762
Table 508. Source address filtering . . . 2763
Table 509. VLAN match status . . . 2764
Table 510. Ordinary clock: PTP messages for snapshot . . . 2769
Table 511. End-to-end transparent clock: PTP messages for snapshot . . . 2770
Table 512. Peer-to-peer transparent clock: PTP messages for snapshot . . . 2771
Table 513. PTP message generation criteria . . . 2778
Table 514. TSO: TCP and IP header fields . . . 2781
Table 515. Pause packet fields . . . 2783
Table 516. Tx MAC flow control . . . 2784
Table 517. Rx MAC Flow Control . . . 2785
Table 518. Transmit checksum offload engine functions for different packet types . . . 2787
Table 519. Receive checksum offload engine functions for different packet types . . . 2788
Table 520. MCD clock selection . . . 2792
Table 521. MDIO packet field description . . . 2792
Table 522. RX interface signal encoding . . . 2795
Table 523. Transfer complete interrupt behavior . . . 2804
Table 524. TDES0 normal descriptor (read format) . . . 2817
Table 525. TDES1 normal descriptor (read format) . . . 2818
Table 526. TDES2 normal descriptor (read format) . . . 2818
Table 527. TDES3 normal descriptor (Read format) . . . 2819
Table 528. DES0 normal descriptor (write-back format) . . . 2822
Table 529. TDES1 normal descriptor (write-back format) . . . 2822
Table 530. TDES2 normal descriptor (write-back format) . . . 2822
Table 531. TDES3 normal descriptor (write-back format) . . . 2822
Table 532. TDES0 context descriptor . . . 2826
Table 533. TDES1 context descriptor . . . 2826
Table 534. TDES2 context descriptor . . . 2826
Table 535. TDES3 context descriptor . . . 2827
Table 536. RDES0 normal descriptor (read format) . . . 2830
Table 537. RDES1 normal descriptor (read format) . . . 2830
Table 538. RDES2 normal descriptor (read format) . . . 2830
Table 539. RDES3 normal descriptor (read format) . . . 2831
Table 540. RDES0 normal descriptor (write-back format) . . . 2832
Table 541. RDES1 normal descriptor (write-back format) . . . 2833
Table 542. RDES2 normal descriptor (write-back format) . . . 2835
Table 543. RDES3 normal descriptor (write-back format) . . . 2836
Table 544. RDES0 context descriptor . . . 2839
Table 545. RDES1 context descriptor . . . 2840
Table 546. RDES2 context descriptor . . . 2840
Table 547. RDES3 context descriptor . . . 2840
Table 548. ETH_DMA common register map and reset values . . . 2864
Table 549. ETH_DMA_CH register map and reset values . . . 2864
Table 550. ETH_MTL register map and reset values . . . 2877
Table 551. Giant Packet Status based on S2KP and JE Bits . . . 2884
Table 552. Packet Length based on the CST and ACS bits . . . 2885
Table 553. Remote wakeup packet filter register . . . 2907
Table 554. ETH_MACRWKPFR . . . 2908
Table 555. Remote Wakeup Packet and PMT Interrupt Generation . . . 2909
Table 556. Timestamp Snapshot Dependency on Register Bits . . . 2948
Table 557. Ethernet MAC register map and reset values . . . 2969
Table 558. HDMI pin . . . 2979

Table 559. HDMI-CEC internal input/output signals . . . 2979
Table 560. Error handling timing parameters . . . 2985
Table 561. TXERR timing parameters . . . 2987
Table 562. HDMI-CEC interrupts . . . 2988
Table 563. HDMI-CEC register map and reset values . . . 2997
Table 564. JTAG/Serial-wire debug port pins . . . 3000
Table 565. Trace port pins . . . 3000
Table 566. Serial-wire trace port pins . . . 3000
Table 567. Trigger pins . . . 3000
Table 568. Packet request . . . 3004
Table 569. ACK response . . . 3004
Table 570. Data transfer . . . 3004
Table 571. JTAG-DP data registers . . . 3007
Table 572. Debug port registers . . . 3009
Table 573. MEM-AP registers . . . 3019
Table 574. System ROM table 1 . . . 3023
Table 575. System ROM table 2 . . . 3024
Table 576. System ROM table register map and reset values . . . 3030
Table 577. TSG register map and reset values . . . 3038
Table 578. System CTI inputs . . . 3039
Table 579. System CTI outputs . . . 3040
Table 580. Cortex-M7 CTI inputs . . . 3040
Table 581. Cortex-M7 CTI outputs . . . 3041
Table 582. CTI register map and reset values . . . 3056
Table 583. CSTF register map and reset values . . . 3067
Table 584. ETF register map and reset values . . . 3088
Table 585. TPIU register map and reset values . . . 3106
Table 586. SWO register map and reset values . . . 3118
Table 587. SWTF register map and reset values . . . 3129
Table 588. DBGMCU register map and reset values . . . 3138
Table 589. Cortex-M7 CPU ROM table . . . 3139
Table 590. Cortex-M7 PPB ROM table . . . 3139
Table 591. Cortex-M7 CPU ROM table register map and reset values . . . 3145
Table 592. Cortex-M7 PPB ROM table register map and reset values . . . 3150
Table 593. Cortex-M7 DWT register map and reset values . . . 3162
Table 594. Cortex-M7 ITM register map and reset values . . . 3172
Table 595. Cortex-M7 FPB register map and reset values . . . 3179
Table 596. Cortex-M7 ETM register map and reset values . . . 3209
Table 597. Document revision history . . . 3216

List of figures

Figure 1. System architecture for STM32H743/53xx and STM32H750xB devices . . . 100
Figure 2. AXI interconnect . . . 105
Figure 3. FLASH block diagram . . . 134
Figure 4. Detailed FLASH architecture . . . 136
Figure 5. Embedded Flash memory organization . . . 137
Figure 6. Embedded Flash memory usage . . . 139
Figure 7. FLASH protection mechanisms . . . 140
Figure 8. FLASH read pipeline architecture . . . 143
Figure 9. FLASH write pipeline architecture . . . 146
Figure 10. Flash bank swapping sequence . . . 155
Figure 11. RDP protection transition scheme . . . 170
Figure 12. Example of protected region overlapping . . . 171
Figure 13. Flash memory areas and services in Standard and Secure access modes . . . 232
Figure 14. Secure bootloader state machine . . . 233
Figure 15. Root secure service call . . . 234
Figure 16. Core access to Flash memory areas . . . 238
Figure 17. Power control block diagram . . . 240
Figure 18. Power supply overview . . . 244
Figure 19. System supply configurations . . . 245
Figure 20. Device startup with V_{CORE} supplied from voltage regulator . . . 247
Figure 21. Backup domain . . . 251
Figure 22. USB supply configurations . . . 252
Figure 23. Power-on reset/power-down reset waveform . . . 253
Figure 24. BOR thresholds . . . 254
Figure 25. PVD thresholds . . . 255
Figure 26. AVD thresholds . . . 256
Figure 27. VBAT thresholds . . . 257
Figure 28. Temperature thresholds . . . 258
Figure 29. V_{CORE} voltage scaling versus system power modes . . . 263
Figure 30. Power control modes detailed state diagram . . . 265
Figure 31. Dynamic voltage scaling in Run mode . . . 268
Figure 32. Dynamic voltage scaling behavior with D1, D2 and system in Stop mode . . . 269
Figure 33. Dynamic Voltage Scaling D1, D2, system Standby mode . . . 270
Figure 34. Dynamic voltage scaling behavior with D1 and D2 in DStandby mode and D3 in autonomous mode . . . 272
Figure 35. EXTI, RCC and PWR interconnections . . . 295
Figure 36. Timing diagram of SRAM4-to-LPUART1 transfer with BDMA and D3 domain in Autonomous mode . . . 299
Figure 37. Timing diagram of LPUART1 transmission with D3 domain in Autonomous mode . . . 304
Figure 38. RCC Block diagram . . . 307
Figure 39. System reset circuit . . . 310
Figure 40. Boot sequences versus system states . . . 316
Figure 41. Top-level clock tree . . . 318
Figure 42. HSE/LSE clock source . . . 319
Figure 43. PLL block diagram . . . 326
Figure 44. PLLs Initialization Flowchart . . . 329

Figure 45. Core and bus clock generation . . . 331
Figure 46. Kernel clock distribution for SAIs and DFSDM . . . 337
Figure 47. Kernel clock distribution for SPIs and SPI/I2S . . . 338
Figure 48. Kernel clock distribution for I2Cs . . . 339
Figure 49. Kernel clock distribution for UARTs, USARTs and LPUART1 . . . 339
Figure 50. Kernel clock distribution for LTDC . . . 340
Figure 51. Kernel clock distribution for SDMMC, QUADSPI and FMC . . . 340
Figure 52. Kernel clock distribution for USB (2) . . . 341
Figure 53. Kernel clock distribution for Ethernet . . . 342
Figure 54. Kernel clock distribution For ADCs, SWPMI, RNG and FDCAN (2) . . . 343
Figure 55. Kernel clock distribution for LPTIMs and HDMI-CEC (2) . . . 344
Figure 56. Peripheral allocation example . . . 347
Figure 57. Kernel Clock switching . . . 350
Figure 58. Peripheral kernel clock enable logic details . . . 353
Figure 59. Bus clock enable logic . . . 358
Figure 60. RCC mapping overview . . . 360
Figure 61. CRS block diagram . . . 483
Figure 62. CRS counter behavior . . . 485
Figure 63. HSEM block diagram . . . 495
Figure 64. Procedure state diagram . . . 496
Figure 65. Interrupt state diagram . . . 499
Figure 66. Basic structure of an I/O port bit . . . 509
Figure 67. Basic structure of a 5-Volt tolerant I/O port bit . . . 509
Figure 68. Input floating/pull up/pull down configurations . . . 514
Figure 69. Output configuration . . . 515
Figure 70. Alternate function configuration . . . 516
Figure 71. High impedance-analog configuration . . . 516
Figure 72. Analog inputs connected to ADC inputs . . . 517
Figure 73. MDMA block diagram . . . 588
Figure 74. DMA block diagram . . . 617
Figure 75. Peripheral-to-memory mode . . . 620
Figure 76. Memory-to-peripheral mode . . . 621
Figure 77. Memory-to-memory mode . . . 622
Figure 78. FIFO structure . . . 627
Figure 79. BDMA block diagram . . . 650
Figure 80. DMAMUX block diagram . . . 676
Figure 81. Synchronization mode of the DMAMUX request line multiplexer channel . . . 679
Figure 82. Event generation of the DMA request line multiplexer channel . . . 679
Figure 83. DMA2D block diagram . . . 694
Figure 84. Intel 8080 16-bit mode (RGB565) . . . 701
Figure 85. Intel 8080 18/24-bit mode (RGB888) . . . 701
Figure 86. EXTI block diagram . . . 739
Figure 87. Configurable event triggering logic CPU wakeup . . . 741
Figure 88. Configurable event triggering logic Any wakeup . . . 743
Figure 89. Direct event triggering logic CPU Wakeup . . . 744
Figure 90. Direct event triggering logic Any Wakeup . . . 745
Figure 91. D3 domain Pending request clear logic . . . 746
Figure 92. CRC calculation unit block diagram . . . 771
Figure 93. FMC block diagram . . . 778
Figure 94. FMC memory banks (default mapping) . . . 782
Figure 95. Mode 1 read access waveforms . . . 793
Figure 96. Mode 1 write access waveforms . . . 794

Figure 97. Mode A read access waveforms . . . 796
Figure 98. Mode A write access waveforms . . . 797
Figure 99. Mode 2 and mode B read access waveforms . . . 799
Figure 100. Mode 2 write access waveforms . . . 799
Figure 101. Mode B write access waveforms . . . 800
Figure 102. Mode C read access waveforms . . . 802
Figure 103. Mode C write access waveforms . . . 802
Figure 104. Mode D read access waveforms . . . 805
Figure 105. Mode D write access waveforms . . . 805
Figure 106. Muxed read access waveforms . . . 808
Figure 107. Muxed write access waveforms . . . 808
Figure 108. Asynchronous wait during a read access waveforms . . . 811
Figure 109. Asynchronous wait during a write access waveforms . . . 811
Figure 110. Wait configuration waveforms . . . 814
Figure 111. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) . . . 814
Figure 112. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) . . . 816
Figure 113. NAND Flash controller waveforms for common memory access . . . 830
Figure 114. Access to non ‘CE don’t care’ NAND-Flash . . . 831
Figure 115. Burst write SDRAM access waveforms . . . 841
Figure 116. Burst read SDRAM access . . . 842
Figure 117. Logic diagram of Read access with RBURST bit set (CAS=2, RPIPE=0) . . . 843
Figure 118. Read access crossing row boundary . . . 845
Figure 119. Write access crossing row boundary . . . 845
Figure 120. Self-refresh mode . . . 848
Figure 121. Power-down mode . . . 849
Figure 122. QUADSPI block diagram when dual-flash mode is disabled . . . 860
Figure 123. QUADSPI block diagram when dual-flash mode is enabled . . . 860
Figure 124. An example of a read command in quad mode . . . 862
Figure 125. An example of a DDR command in quad mode . . . 865
Figure 126. nCS when CKMODE = 0 (T = CLK period) . . . 873
Figure 127. nCS when CKMODE = 1 in SDR mode (T = CLK period) . . . 874
Figure 128. nCS when CKMODE = 1 in DDR mode (T = CLK period) . . . 874
Figure 129. nCS when CKMODE = 1 with an abort (T = CLK period) . . . 874
Figure 130. DLYB block diagram . . . 889
Figure 131. ADC block diagram . . . 897
Figure 132. ADC clock scheme . . . 900
Figure 133. ADC1 connectivity . . . 901
Figure 134. ADC2 connectivity . . . 902
Figure 135. ADC3 connectivity . . . 903
Figure 136. ADC calibration . . . 907
Figure 137. Updating the ADC offset calibration factor . . . 907
Figure 138. Mixing single-ended and differential channels . . . 908
Figure 139. Enabling / Disabling the ADC . . . 911
Figure 140. Analog to digital conversion time . . . 917
Figure 141. Stopping ongoing regular conversions . . . 918
Figure 142. Stopping ongoing regular and injected conversions . . . 918
Figure 143. Triggers are shared between ADC master and ADC slave . . . 920
Figure 144. Injected conversion latency . . . 924
Figure 145. Example of JSQR queue of context (sequence change) . . . 927
Figure 146. Example of JSQR queue of context (trigger change) . . . 927
Figure 147. Example of JSQR queue of context with overflow before conversion . . . 928
Figure 148. Example of JSQR queue of context with overflow during conversion . . . 928

Figure 149. Example of JSQR queue of context with empty queue (case JQM=0) . . . 929
Figure 150. Example of JSQR queue of context with empty queue (case JQM=1) . . . 930
Figure 151. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion. . . . 930
Figure 152. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs. . . . 931
Figure 153. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion . . . 931
Figure 154. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1) . . . 932
Figure 155. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0) . . . 932
Figure 156. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1) . . . 933
Figure 157. Single conversions of a sequence, software trigger . . . 935
Figure 158. Continuous conversion of a sequence, software trigger . . . 935
Figure 159. Single conversions of a sequence, hardware trigger . . . 936
Figure 160. Continuous conversions of a sequence, hardware trigger . . . 936
Figure 161. Right alignment (offset disabled, unsigned value) . . . 938
Figure 162. Right alignment (offset enabled, signed value) . . . 938
Figure 163. Left alignment (offset disabled, unsigned value) . . . 939
Figure 164. Left alignment (offset enabled, signed value) . . . 939
Figure 165. Example of overrun (OVSR) . . . 941
Figure 166. AUTODLY=1, regular conversion in continuous mode, software trigger . . . 945
Figure 167. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0) . . . 945
Figure 168. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1) . . . 946
Figure 169. AUTODLY=1, regular continuous conversions interrupted by injected conversions . . . 947
Figure 170. AUTODLY=1 in auto- injected mode (JAUTO=1) . . . 947
Figure 171. Analog watchdog guarded area . . . 948
Figure 172. ADCy_AWDx_OUT signal generation (on all regular channels) . . . 950
Figure 173. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW) . . . 950
Figure 174. ADCy_AWDx_OUT signal generation (on a single regular channel) . . . 951
Figure 175. ADCy_AWDx_OUT signal generation (on all injected channels) . . . 951
Figure 176. 16-bit result oversampling with 10-bits right shift and rouding . . . 952
Figure 177. Triggered regular oversampling mode (TROVS bit = 1) . . . 954
Figure 178. Regular oversampling modes (4x ratio) . . . 955
Figure 179. Regular and injected oversampling modes used simultaneously . . . 955
Figure 180. Triggered regular oversampling with injection . . . 956
Figure 181. Oversampling in auto-injected mode . . . 956
Figure 182. Dual ADC block diagram(1) . . . 959
Figure 183. Injected simultaneous mode on 4 channels: dual ADC mode . . . 960
Figure 184. Regular simultaneous mode on 16 channels: dual ADC mode . . . 962
Figure 185. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode . . . 963
Figure 186. Interleaved mode on 1 channel in single conversion mode: dual ADC mode . . . 964
Figure 187. Interleaved conversion with injection . . . 964
Figure 188. Alternate trigger: injected group of each ADC . . . 965
Figure 189. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 966
Figure 190. Alternate + regular simultaneous . . . 967
Figure 191. Case of trigger occurring during injected conversion . . . 967
Figure 192. Interleaved single channel CH0 with injected sequence CH11, CH12 . . . 968
Figure 193. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first . . . 968

Figure 194. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first . . . 968
Figure 195. DMA Requests in regular simultaneous mode when DAMDF=0b00 . . . 969
Figure 196. DMA requests in regular simultaneous mode when DAMDF=0b10 . . . 970
Figure 197. DMA requests in interleaved mode when DAMDF=0b10 . . . 970
Figure 198. Temperature sensor channel block diagram . . . 972
Figure 199. VBAT channel block diagram . . . 974
Figure 200. VREFINT channel block diagram . . . 974
Figure 201. Dual-channel DAC block diagram . . . 1022
Figure 202. Data registers in single DAC channel mode . . . 1024
Figure 203. Data registers in dual DAC channel mode . . . 1025
Figure 204. Timing diagram for conversion with trigger disabled TEN = 0 . . . 1025
Figure 205. DAC LFSR register calculation algorithm . . . 1028
Figure 206. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 1028
Figure 207. DAC triangle wave generation . . . 1029
Figure 208. DAC conversion (SW trigger enabled) with triangle wave generation . . . 1029
Figure 209. DAC Sample and Hold mode phase diagram . . . 1032
Figure 210. Comparator functional block diagram . . . 1062
Figure 211. Comparator hysteresis . . . 1065
Figure 212. Comparator output blanking . . . 1065
Figure 213. Output redirection . . . 1067
Figure 214. Scaler block diagram . . . 1069
Figure 215. Standalone mode: external gain setting mode . . . 1080
Figure 216. Follower configuration . . . 1081
Figure 217. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used . . . 1082
Figure 218. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering . . . 1083
Figure 219. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15) . . . 1084
Figure 220. Example configuration . . . 1084
Figure 221. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15) with filtering . . . 1085
Figure 222. Example configuration . . . 1085
Figure 223. Single DFSDM block diagram . . . 1099
Figure 224. Input channel pins redirection . . . 1103
Figure 225. Channel transceiver timing diagrams . . . 1105
Figure 226. Clock absence timing diagram for SPI . . . 1106
Figure 227. Clock absence timing diagram for Manchester coding . . . 1107
Figure 228. First conversion for Manchester coding (Manchester synchronization) . . . 1109
Figure 229. DFSDM_CHyDATINR registers operation modes and assignment . . . 1113
Figure 230. Example: Sinc3 filter response . . . 1115
Figure 231. DCMI block diagram . . . 1155
Figure 232. Top-level block diagram . . . 1155
Figure 233. DCMI signal waveforms . . . 1157
Figure 234. Timing diagram . . . 1159
Figure 235. Frame capture waveforms in snapshot mode . . . 1161
Figure 236. Frame capture waveforms in continuous grab mode . . . 1162
Figure 237. Coordinates and size of the window after cropping . . . 1162
Figure 238. Data capture waveforms . . . 1163
Figure 239. Pixel raster scan order . . . 1164
Figure 240. LTDC block diagram . . . 1181
Figure 241. LCD-TFT synchronous timings . . . 1184

Figure 242. Layer window programmable parameters . . . 1187
Figure 243. Blending two layers with background . . . 1190
Figure 244. Interrupt events . . . 1191
Figure 245. JPEG codec block diagram . . . 1216
Figure 246. RNG block diagram . . . 1231
Figure 247. Entropy source model . . . 1232
Figure 248. RNG initialization overview . . . 1235
Figure 249. CRYP block diagram . . . 1246
Figure 250. AES-ECB mode overview . . . 1249
Figure 251. AES-CBC mode overview . . . 1250
Figure 252. AES-CTR mode overview . . . 1251
Figure 253. AES-GCM mode overview . . . 1252
Figure 254. AES-GMAC mode overview . . . 1252
Figure 255. AES-CCM mode overview . . . 1253
Figure 256. STM32 cryptolib DES/TDES flowcharts . . . 1254
Figure 257. STM32 cryptolib AES flowchart examples . . . 1255
Figure 258. Example of suspend mode management . . . 1260
Figure 259. DES/TDES-ECB mode encryption . . . 1261
Figure 260. DES/TDES-ECB mode decryption . . . 1262
Figure 261. DES/TDES-CBC mode encryption . . . 1263
Figure 262. DES/TDES-CBC mode decryption . . . 1264
Figure 263. AES-ECB mode encryption . . . 1266
Figure 264. AES-ECB mode decryption . . . 1267
Figure 265. AES-CBC mode encryption . . . 1268
Figure 266. AES-CBC mode decryption . . . 1269
Figure 267. Message construction for the Counter mode . . . 1271
Figure 268. AES-CTR mode encryption . . . 1272
Figure 269. AES-CTR mode decryption . . . 1273
Figure 270. Message construction for the Galois/counter mode . . . 1275
Figure 271. Message construction for the Galois Message Authentication Code mode . . . 1280
Figure 272. Message construction for the Counter with CBC-MAC mode . . . 1281
Figure 273. 64-bit block construction according to the data type (IN FIFO) . . . 1288
Figure 274. 128-bit block construction according to the data type . . . 1290
Figure 275. CRYP interrupt mapping diagram . . . 1295
Figure 276. HASH block diagram . . . 1315
Figure 277. Message data swapping feature . . . 1317
Figure 278. HASH save/restore mechanism . . . 1323
Figure 279. HASH interrupt mapping diagram . . . 1325
Figure 280. High-resolution timer block diagram . . . 1342
Figure 281. Timer A..E overview . . . 1347
Figure 282. Continuous timer operation . . . 1348
Figure 283. Single-shot timer operation . . . 1349
Figure 284. Timer reset resynchronization (prescaling ratio above 32) . . . 1350
Figure 285. Repetition rate vs HRTIM_REPxR content in continuous mode . . . 1351
Figure 286. Repetition counter behavior in single-shot mode . . . 1352
Figure 287. Compare events action on outputs: set on compare 1, reset on compare 2 . . . 1353
Figure 288. Timing unit capture circuitry . . . 1355
Figure 289. Auto-delayed overview (Compare 2 only) . . . 1356
Figure 290. Auto-delayed compare . . . 1357
Figure 291. Push-pull mode block diagram . . . 1359
Figure 292. Push-pull mode example . . . 1360
Figure 293. Complementary outputs with deadtime insertion . . . 1360

Figure 294. Deadtime insertion vs deadtime sign (1 indicates negative deadtime) . . . 1361
Figure 295. Complementary outputs for low pulse width (SDTRx = SDTFx = 0) . . . 1362
Figure 296. Complementary outputs for low pulse width (SDTRx = SDTFx = 1) . . . 1362
Figure 297. Complementary outputs for low pulse width (SDTRx = 0, SDTFx = 1) . . . 1362
Figure 298. Complementary outputs for low pulse width (SDTRx = 1, SDTFx=0) . . . 1363
Figure 299. Master timer overview . . . 1364
Figure 300. External event conditioning overview (1 channel represented) . . . 1367
Figure 301. Latency to external events falling edge (counter reset and output set) . . . 1370
Figure 302. Latency to external events (output reset on external event) . . . 1370
Figure 303. Event blanking mode . . . 1371
Figure 304. Event postpone mode . . . 1371
Figure 305. External trigger blanking with edge-sensitive trigger . . . 1373
Figure 306. External trigger blanking, level sensitive triggering . . . 1373
Figure 307. Event windowing mode . . . 1374
Figure 308. External trigger windowing with edge-sensitive trigger . . . 1375
Figure 309. External trigger windowing, level sensitive triggering . . . 1375
Figure 310. Delayed Idle mode entry . . . 1377
Figure 311. Burst mode and delayed protection priorities (DIDL = 0) . . . 1378
Figure 312. Burst mode and delayed protection priorities (DIDL = 1) . . . 1379
Figure 313. Balanced Idle protection example . . . 1380
Figure 314. Output management overview . . . 1390
Figure 315. HRTIM output states and transitions . . . 1390
Figure 316. Burst mode operation example . . . 1392
Figure 317. Burst mode trigger on external event . . . 1394
Figure 318. Delayed burst mode entry with deadtime enabled and IDLESx = 1 . . . 1396
Figure 319. Delayed Burst mode entry during deadtime . . . 1397
Figure 320. Burst mode exit when the deadtime generator is enabled . . . 1398
Figure 321. Burst mode emulation example . . . 1400
Figure 322. Carrier frequency signal insertion . . . 1400
Figure 323. HRTIM outputs with Chopper mode enabled . . . 1401
Figure 324. Fault protection circuitry (FAULT1 fully represented, FAULT2..5 partially) . . . 1402
Figure 325. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING}$ = fHRTIM, N = 4) . . . 1403
Figure 326. Auxiliary outputs . . . 1405
Figure 327. Auxiliary and main outputs during burst mode (DIDLx = 0) . . . 1406
Figure 328. Deadtime distortion on auxiliary output when exiting burst mode . . . 1406
Figure 329. Counter behavior in synchronized start mode . . . 1410
Figure 330. ADC trigger selection overview . . . 1411
Figure 331. Combining several updates on a single hrtim_dac_trgx output . . . 1412
Figure 332. DMA burst overview . . . 1416
Figure 333. Burst DMA operation flowchart . . . 1417
Figure 334. Registers update following DMA burst transfer . . . 1418
Figure 335. Buck converter topology . . . 1420
Figure 336. Dual Buck converter management . . . 1421
Figure 337. Synchronous rectification depending on output current . . . 1421
Figure 338. Buck with synchronous rectification . . . 1422
Figure 339. 3-phase interleaved buck converter . . . 1423
Figure 340. 3-phase interleaved buck converter control . . . 1424
Figure 341. Transition mode PFC . . . 1424
Figure 342. Transition mode PFC waveforms . . . 1425
Figure 343. Advanced-control timer block diagram . . . 1515
Figure 344. Counter timing diagram with prescaler division change from 1 to 2 . . . 1517
Figure 345. Counter timing diagram with prescaler division change from 1 to 4 . . . 1517

Figure 346. Counter timing diagram, internal clock divided by 1 . . . 1519
Figure 347. Counter timing diagram, internal clock divided by 2 . . . 1519
Figure 348. Counter timing diagram, internal clock divided by 4 . . . 1520
Figure 349. Counter timing diagram, internal clock divided by N . . . 1520
Figure 350. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . 1521
Figure 351. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 1521
Figure 352. Counter timing diagram, internal clock divided by 1 . . . 1523
Figure 353. Counter timing diagram, internal clock divided by 2 . . . 1523
Figure 354. Counter timing diagram, internal clock divided by 4 . . . 1524
Figure 355. Counter timing diagram, internal clock divided by N . . . 1524
Figure 356. Counter timing diagram, update event when repetition counter is not used . . . 1525
Figure 357. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 1526
Figure 358. Counter timing diagram, internal clock divided by 2 . . . 1527
Figure 359. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . 1527
Figure 360. Counter timing diagram, internal clock divided by N . . . 1528
Figure 361. Counter timing diagram, update event with ARPE=1 (counter underflow) . . . 1528
Figure 362. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . 1529
Figure 363. Update rate examples depending on mode and TIMx_RCR register settings . . . 1530
Figure 364. External trigger input block . . . 1531
Figure 365. TIM1/TIM8 ETR input circuitry . . . 1531
Figure 366. Control circuit in normal mode, internal clock divided by 1 . . . 1532
Figure 367. TI2 external clock connection example . . . 1533
Figure 368. Control circuit in external clock mode 1 . . . 1534
Figure 369. External trigger input block . . . 1534
Figure 370. Control circuit in external clock mode 2 . . . 1535
Figure 371. Capture/compare channel (example: channel 1 input stage) . . . 1536
Figure 372. Capture/compare channel 1 main circuit . . . 1537
Figure 373. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) . . . 1537
Figure 374. Output stage of capture/compare channel (channel 4) . . . 1538
Figure 375. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . 1538
Figure 376. PWM input mode timing . . . 1540
Figure 377. Output compare mode, toggle on OC1 . . . 1542
Figure 378. Edge-aligned PWM waveforms (ARR=8) . . . 1543
Figure 379. Center-aligned PWM waveforms (ARR=8) . . . 1544
Figure 380. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 1546
Figure 381. Combined PWM mode on channel 1 and 3 . . . 1547
Figure 382. 3-phase combined PWM signals with multiple trigger pulses per period . . . 1548
Figure 383. Complementary output with dead-time insertion . . . 1549
Figure 384. Dead-time waveforms with delay greater than the negative pulse . . . 1549
Figure 385. Dead-time waveforms with delay greater than the positive pulse . . . 1550
Figure 386. Break and Break2 circuitry overview . . . 1552
Figure 387. Various output behavior in response to a break event on BRK (OSSI = 1) . . . 1554
Figure 388. PWM output state following BRK and BRK2 pins assertion (OSSI=1) . . . 1555
Figure 389. PWM output state following BRK assertion (OSSI=0) . . . 1556
Figure 390. Output redirection . . . 1556
Figure 391. Clearing TIMx OCxREF . . . 1557
Figure 392. 6-step generation, COM example (OSSR=1) . . . 1558
Figure 393. Example of one pulse mode. . . . 1559
Figure 394. Retriggerable one pulse mode . . . 1561
Figure 395. Example of counter operation in encoder interface mode. . . . 1562
Figure 396. Example of encoder interface mode with TI1FP1 polarity inverted. . . . 1563
Figure 397. Measuring time interval between edges on 3 signals . . . 1564

Figure 398. Example of Hall sensor interface . . . 1566
Figure 399. Control circuit in reset mode . . . 1567
Figure 400. Control circuit in Gated mode . . . 1568
Figure 401. Control circuit in trigger mode . . . 1569
Figure 402. Control circuit in external clock mode 2 + trigger mode . . . 1570
Figure 403. General-purpose timer block diagram . . . 1620
Figure 404. Counter timing diagram with prescaler division change from 1 to 2 . . . 1622
Figure 405. Counter timing diagram with prescaler division change from 1 to 4 . . . 1622
Figure 406. Counter timing diagram, internal clock divided by 1 . . . 1623
Figure 407. Counter timing diagram, internal clock divided by 2 . . . 1624
Figure 408. Counter timing diagram, internal clock divided by 4 . . . 1624
Figure 409. Counter timing diagram, internal clock divided by N . . . 1625
Figure 410. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) . . . 1625
Figure 411. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) . . . 1626
Figure 412. Counter timing diagram, internal clock divided by 1 . . . 1627
Figure 413. Counter timing diagram, internal clock divided by 2 . . . 1627
Figure 414. Counter timing diagram, internal clock divided by 4 . . . 1628
Figure 415. Counter timing diagram, internal clock divided by N . . . 1628
Figure 416. Counter timing diagram, Update event when repetition counter is not used . . . 1629
Figure 417. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . 1630
Figure 418. Counter timing diagram, internal clock divided by 2 . . . 1631
Figure 419. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . 1631
Figure 420. Counter timing diagram, internal clock divided by N . . . 1632
Figure 421. Counter timing diagram, Update event with ARPE=1 (counter underflow) . . . 1632
Figure 422. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . 1633
Figure 423. Control circuit in normal mode, internal clock divided by 1 . . . 1634
Figure 424. TI2 external clock connection example . . . 1634
Figure 425. Control circuit in external clock mode 1 . . . 1635
Figure 426. External trigger input block . . . 1636
Figure 427. Control circuit in external clock mode 2 . . . 1636
Figure 428. Capture/Compare channel (example: channel 1 input stage) . . . 1637
Figure 429. Capture/Compare channel 1 main circuit . . . 1638
Figure 430. Output stage of Capture/Compare channel (channel 1) . . . 1638
Figure 431. PWM input mode timing . . . 1640
Figure 432. Output compare mode, toggle on OC1 . . . 1642
Figure 433. Edge-aligned PWM waveforms (ARR=8) . . . 1643
Figure 434. Center-aligned PWM waveforms (ARR=8) . . . 1645
Figure 435. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 1646
Figure 436. Combined PWM mode on channels 1 and 3 . . . 1647
Figure 437. Clearing TIMx OCxREF . . . 1648
Figure 438. Example of one-pulse mode . . . 1649
Figure 439. Retriggerable one pulse mode . . . 1651
Figure 440. Example of counter operation in encoder interface mode . . . 1652
Figure 441. Example of encoder interface mode with TI1FP1 polarity inverted . . . 1653
Figure 442. Control circuit in reset mode . . . 1654
Figure 443. Control circuit in gated mode . . . 1655
Figure 444. Control circuit in trigger mode . . . 1656
Figure 445. Control circuit in external clock mode 2 + trigger mode . . . 1657
Figure 446. Master/Slave timer example . . . 1657
Figure 447. Gating TIM2 with OC1REF of TIM3 . . . 1658
Figure 448. Gating TIM2 with Enable of TIM3 . . . 1659

Figure 449. Triggering TIM2 with update of TIM3 1660
Figure 450. Triggering TIM2 with Enable of TIM3 1660
Figure 451. Triggering TIM3 and TIM2 with TIM3 TI1 input. 1661
Figure 452. General-purpose timer block diagram (TIM12). 1695
Figure 453. General-purpose timer block diagram (TIM13/TIM14) 1696
Figure 454. Counter timing diagram with prescaler division change from 1 to 2 1698
Figure 455. Counter timing diagram with prescaler division change from 1 to 4 1698
Figure 456. Counter timing diagram, internal clock divided by 1 1699
Figure 457. Counter timing diagram, internal clock divided by 2 1700
Figure 458. Counter timing diagram, internal clock divided by 4 1700
Figure 459. Counter timing diagram, internal clock divided by N. 1701
Figure 460. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded). 1701
Figure 461. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded). 1702
Figure 462. Control circuit in normal mode, internal clock divided by 1. 1703
Figure 463. TI2 external clock connection example. 1703
Figure 464. Control circuit in external clock mode 1 1704
Figure 465. Capture/compare channel (example: channel 1 input stage 1705
Figure 466. Capture/compare channel 1 main circuit 1705
Figure 467. Output stage of capture/compare channel (channel 1). 1706
Figure 468. PWM input mode timing 1708
Figure 469. Output compare mode, toggle on OC1. 1710
Figure 470. Edge-aligned PWM waveforms (ARR=8) 1711
Figure 471. Combined PWM mode on channel 1 and 2 1712
Figure 472. Example of one pulse mode. 1713
Figure 473. Retriggerable one pulse mode 1714
Figure 474. Measuring time interval between edges on 2 signals 1715
Figure 475. Control circuit in reset mode 1716
Figure 476. Control circuit in gated mode 1717
Figure 477. Control circuit in trigger mode. 1718
Figure 478. TIM15 block diagram 1748
Figure 479. TIM16/TIM17 block diagram 1749
Figure 480. Counter timing diagram with prescaler division change from 1 to 2 1751
Figure 481. Counter timing diagram with prescaler division change from 1 to 4 1751
Figure 482. Counter timing diagram, internal clock divided by 1 1753
Figure 483. Counter timing diagram, internal clock divided by 2 1753
Figure 484. Counter timing diagram, internal clock divided by 4 1754
Figure 485. Counter timing diagram, internal clock divided by N. 1754
Figure 486. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded). 1755
Figure 487. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded). 1755
Figure 488. Update rate examples depending on mode and TIMx_RCR register settings 1757
Figure 489. Control circuit in normal mode, internal clock divided by 1. 1758
Figure 490. TI2 external clock connection example. 1758
Figure 491. Control circuit in external clock mode 1 1759
Figure 492. Capture/compare channel (example: channel 1 input stage) 1760
Figure 493. Capture/compare channel 1 main circuit 1760
Figure 494. Output stage of capture/compare channel (channel 1). 1761
Figure 495. Output stage of capture/compare channel (channel 2 for TIM15) 1761
Figure 496. PWM input mode timing 1763

Figure 497. Output compare mode, toggle on OC1 . . . 1765
Figure 498. Edge-aligned PWM waveforms (ARR=8) . . . 1766
Figure 499. Combined PWM mode on channel 1 and 2 . . . 1767
Figure 500. Complementary output with dead-time insertion. . . . 1768
Figure 501. Dead-time waveforms with delay greater than the negative pulse. . . . 1769
Figure 502. Dead-time waveforms with delay greater than the positive pulse. . . . 1769
Figure 503. Break circuitry overview . . . 1771
Figure 504. Output behavior in response to a break . . . 1773
Figure 505. Example of one pulse mode. . . . 1774
Figure 506. Retriggerable one pulse mode . . . 1776
Figure 507. Measuring time interval between edges on 2 signals . . . 1777
Figure 508. Control circuit in reset mode . . . 1778
Figure 509. Control circuit in gated mode . . . 1779
Figure 510. Control circuit in trigger mode . . . 1780
Figure 511. Basic timer block diagram . . . 1830
Figure 512. Counter timing diagram with prescaler division change from 1 to 2 . . . 1832
Figure 513. Counter timing diagram with prescaler division change from 1 to 4 . . . 1832
Figure 514. Counter timing diagram, internal clock divided by 1 . . . 1833
Figure 515. Counter timing diagram, internal clock divided by 2 . . . 1834
Figure 516. Counter timing diagram, internal clock divided by 4 . . . 1834
Figure 517. Counter timing diagram, internal clock divided by N . . . 1835
Figure 518. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 1835
Figure 519. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 1836
Figure 520. Control circuit in normal mode, internal clock divided by 1 . . . 1837
Figure 521. Low-power timer block diagram (LPTIM1 and LPTIM2) . . . 1844
Figure 522. Low-power timer block diagram (LPTIM3) . . . 1845
Figure 523. Low-power timer block diagram (LPTIM4 and LPTIM5) . . . 1845
Figure 524. Glitch filter timing diagram . . . 1850
Figure 525. LPTIM output waveform, single counting mode configuration . . . 1852
Figure 526. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 1852
Figure 527. LPTIM output waveform, Continuous counting mode configuration . . . 1853
Figure 528. Waveform generation . . . 1854
Figure 529. Encoder mode counting sequence . . . 1858
Figure 530. Watchdog block diagram . . . 1874
Figure 531. Window watchdog timing diagram . . . 1876
Figure 532. Independent watchdog block diagram . . . 1880
Figure 533. RTC block overview . . . 1890
Figure 534. Detailed RTC block diagram . . . 1891
Figure 535. Tamper detection . . . 1892
Figure 536. I2C block diagram . . . 1937
Figure 537. I2C bus protocol . . . 1939
Figure 538. Setup and hold timings . . . 1940
Figure 539. I2C initialization flowchart . . . 1943
Figure 540. Data reception . . . 1944
Figure 541. Data transmission . . . 1945
Figure 542. Slave initialization flowchart . . . 1948
Figure 543. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0 . . . 1950
Figure 544. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1 . . . 1951
Figure 545. Transfer bus diagrams for I2C slave transmitter . . . 1952

Figure 546. Transfer sequence flowchart for slave receiver with NOSTRETCH=0 1953
Figure 547. Transfer sequence flowchart for slave receiver with NOSTRETCH=1 1954
Figure 548. Transfer bus diagrams for I2C slave receiver 1954
Figure 549. Master clock generation 1956
Figure 550. Master initialization flowchart 1958
Figure 551. 10-bit address read access with HEAD10R=0 1958
Figure 552. 10-bit address read access with HEAD10R=1 1959
Figure 553. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes 1960
Figure 554. Transfer sequence flowchart for I2C master transmitter for N>255 bytes 1961
Figure 555. Transfer bus diagrams for I2C master transmitter 1962
Figure 556. Transfer sequence flowchart for I2C master receiver for N≤255 bytes. 1964
Figure 557. Transfer sequence flowchart for I2C master receiver for N >255 bytes 1965
Figure 558. Transfer bus diagrams for I2C master receiver 1966
Figure 559. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ 1970
Figure 560. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC. 1974
Figure 561. Transfer bus diagrams for SMBus slave transmitter (SBC=1) 1975
Figure 562. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC 1976
Figure 563. Bus transfer diagrams for SMBus slave receiver (SBC=1). 1977
Figure 564. Bus transfer diagrams for SMBus master transmitter. 1978
Figure 565. Bus transfer diagrams for SMBus master receiver 1980
Figure 566. I2C interrupt mapping diagram 1986
Figure 567. USART block diagram 2006
Figure 568. Word length programming 2009
Figure 569. Configurable stop bits 2011
Figure 570. TC/TXE behavior when transmitting 2014
Figure 571. Start bit detection when oversampling by 16 or 8 2015
Figure 572. usart_ker_ck clock divider block diagram 2018
Figure 573. Data sampling when oversampling by 16 2020
Figure 574. Data sampling when oversampling by 8 2020
Figure 575. Mute mode using Idle line detection 2027
Figure 576. Mute mode using address mark detection 2028
Figure 577. Break detection in LIN mode (11-bit break length - LBDL bit is set). 2031
Figure 578. Break detection in LIN mode vs. Framing error detection. 2032
Figure 579. USART example of synchronous master transmission. 2033
Figure 580. USART data clock timing diagram in synchronous master mode (M bits =00). 2033
Figure 581. USART data clock timing diagram in synchronous master mode (M bits = ‘01’) 2034
Figure 582. USART data clock timing diagram in synchronous slave mode (M bits =00). 2035
Figure 583. ISO 7816-3 asynchronous protocol 2037
Figure 584. Parity error detection using the 1.5 stop bits 2039
Figure 585. IrDA SIR ENDEC block diagram. 2043
Figure 586. IrDA data modulation (3/16) - Normal mode. 2043
Figure 587. Transmission using DMA 2045
Figure 588. Reception using DMA 2046
Figure 589. Hardware flow control between 2 USARTs 2046
Figure 590. RS232 RTS flow control 2047
Figure 591. RS232 CTS flow control 2048
Figure 592. Wakeup event verified (wakeup event = address match, FIFO disabled) 2051
Figure 593. Wakeup event not verified (wakeup event = address match, FIFO disabled) 2051

Figure 594. LPUART block diagram . . . 2084
Figure 595. LPUART word length programming . . . 2086
Figure 596. Configurable stop bits . . . 2088
Figure 597. TC/TXE behavior when transmitting . . . 2090
Figure 598. lpuart_ker_ck clock divider block diagram . . . 2093
Figure 599. Mute mode using Idle line detection . . . 2097
Figure 600. Mute mode using address mark detection . . . 2098
Figure 601. Transmission using DMA . . . 2101
Figure 602. Reception using DMA . . . 2102
Figure 603. Hardware flow control between 2 LPUARTs . . . 2102
Figure 604. RS232 RTS flow control . . . 2103
Figure 605. RS232 CTS flow control . . . 2104
Figure 606. Wakeup event verified (wakeup event = address match, FIFO disabled) . . . 2107
Figure 607. Wakeup event not verified (wakeup event = address match, FIFO disabled) . . . 2107
Figure 608. SPI2S block diagram . . . 2131
Figure 609. Full-duplex single master/ single slave application . . . 2133
Figure 610. Half-duplex single master/ single slave application . . . 2133
Figure 611. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) . . . 2135
Figure 612. Master and three independent slaves at star topology . . . 2136
Figure 613. Master and three slaves at circular (daisy chain) topology . . . 2137
Figure 614. Multi-master application . . . 2138
Figure 615. Scheme of SS control logic . . . 2140
Figure 616. Data flow timing control (SSOE=1, SSOM=0, SSM=0) . . . 2140
Figure 617. SS interleaving pulses between data (SSOE=1, SSOM=1,SSM=0) . . . 2141
Figure 618. Data clock timing diagram . . . 2143
Figure 619. Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit . . . 2144
Figure 620. Packing data in FIFO for transmission and reception . . . 2151
Figure 621. TI mode transfer . . . 2153
Figure 622. Low-power mode application example . . . 2158
Figure 623. Waveform examples . . . 2164
Figure 624. Master I2S Philips protocol waveforms (16/32-bit full accuracy) . . . 2165
Figure 625. I2S Philips standard waveforms . . . 2165
Figure 626. Master MSB Justified 16-bit or 32-bit full-accuracy length . . . 2166
Figure 627. Master MSB justified 16 or 24-bit data length . . . 2166
Figure 628. Slave MSB justified . . . 2167
Figure 629. LSB justified 16 or 24-bit data length . . . 2167
Figure 630. Master PCM when the frame length is equal the data length . . . 2168
Figure 631. Master PCM standard waveforms (16 or 24-bit data length) . . . 2168
Figure 632. Slave PCM waveforms . . . 2169
Figure 633. Start-up sequence, I2S Philips standard, master . . . 2171
Figure 634. Start-up sequence, I2S Philips standard, slave . . . 2172
Figure 635. Stop sequence, I2S Philips standard, master . . . 2173
Figure 636. I^2S clock generator architecture . . . 2173
Figure 637. Data Format . . . 2176
Figure 638. Handling of underrun situation . . . 2177
Figure 639. Handling of overrun situation . . . 2178
Figure 640. Frame error detection, with FIXCH=0 . . . 2179
Figure 641. Frame error detection, with FIXCH=1 . . . 2179
Figure 642. SAI functional block diagram . . . 2206

Figure 643. Audio frame . . . 2210
Figure 644. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . 2212
Figure 645. FS role is start of frame (FSDEF = 0) . . . 2213
Figure 646. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . 2214
Figure 647. First bit offset . . . 2214
Figure 648. Audio block clock generator overview . . . 2215
Figure 649. PDM typical connection and timing . . . 2219
Figure 650. Detailed PDM interface block diagram . . . 2220
Figure 651. Start-up sequence . . . 2221
Figure 652. SAI_ADR format in TDM, 32-bit slot width . . . 2222
Figure 653. SAI_ADR format in TDM, 16-bit slot width . . . 2223
Figure 654. SAI_ADR format in TDM, 8-bit slot width . . . 2224
Figure 655. AC'97 audio frame . . . 2227
Figure 656. Example of typical AC'97 configuration on devices featuring at least 2 embedded SAIs (three external AC'97 decoders) . . . 2228
Figure 657. SPDIF format . . . 2229
Figure 658. SAI_xDR register ordering . . . 2230
Figure 659. Data companding hardware in an audio block in the SAI . . . 2233
Figure 660. Tristate strategy on SD output line on an inactive slot . . . 2235
Figure 661. Tristate on output data line in a protocol like I2S . . . 2236
Figure 662. Overrun detection error . . . 2237
Figure 663. FIFO underrun event . . . 2237
Figure 664. SPDIFRX block diagram . . . 2272
Figure 665. S/PDIF Sub-Frame Format . . . 2273
Figure 666. S/PDIF block format . . . 2273
Figure 667. S/PDIF Preambles . . . 2274
Figure 668. Channel coding example . . . 2275
Figure 669. SPDIFRX decoder . . . 2276
Figure 670. Noise filtering and edge detection . . . 2276
Figure 671. Thresholds . . . 2278
Figure 672. Synchronization flowchart . . . 2280
Figure 673. Synchronization process scheduling . . . 2281
Figure 674. SPDIFRX States . . . 2282
Figure 675. SPDIFRX_FMTx_DR register format . . . 2284
Figure 676. Channel/user data format . . . 2285
Figure 677. S/PDIF overrun error when RXSTEO = 0 . . . 2287
Figure 678. S/PDIF overrun error when RXSTEO = 1 . . . 2288
Figure 679. SPDIFRX interface interrupt mapping diagram . . . 2289
Figure 680. S1 signal coding . . . 2306
Figure 681. S2 signal coding . . . 2306
Figure 682. SWPMI block diagram . . . 2308
Figure 683. SWP bus states . . . 2311
Figure 684. SWP frame structure . . . 2312
Figure 685. SWPMI No software buffer mode transmission . . . 2313
Figure 686. SWPMI No software buffer mode transmission, consecutive frames . . . 2314
Figure 687. SWPMI Multi software buffer mode transmission . . . 2316
Figure 688. SWPMI No software buffer mode reception . . . 2318
Figure 689. SWPMI single software buffer mode reception . . . 2319
Figure 690. SWPMI Multi software buffer mode reception . . . 2321
Figure 691. SWPMI single buffer mode reception with CRC error . . . 2322
Figure 692. MDIOS block diagram . . . 2335
Figure 693. MDIO protocol write frame waveform . . . 2336

Figure 694. MDIO protocol read frame waveform . . . 2336
Figure 695. SDMMC “no response” and “no data” operations . . . 2351
Figure 696. SDMMC (multiple) block read operation . . . 2351
Figure 697. SDMMC (multiple) block write operation . . . 2351
Figure 698. SDMMC (sequential) stream read operation . . . 2352
Figure 699. SDMMC (sequential) stream write operation . . . 2352
Figure 700. SDMMC block diagram . . . 2354
Figure 701. SDMMC Command and data phase relation . . . 2356
Figure 702. Control unit . . . 2358
Figure 703. Command/Response path . . . 2359
Figure 704. Command path state machine (CPSM) . . . 2360
Figure 705. Data path . . . 2366
Figure 706. DDR mode data packet clocking . . . 2367
Figure 707. DDR mode CRC status / boot acknowledgment clocking . . . 2367
Figure 708. Data path state machine (DPSM) . . . 2368
Figure 709. CLKMUX unit . . . 2378
Figure 710. Asynchronous interrupt generation . . . 2383
Figure 711. Synchronous interrupt period data read . . . 2383
Figure 712. Synchronous interrupt period data write . . . 2384
Figure 713. Asynchronous interrupt period data read . . . 2385
Figure 714. Asynchronous interrupt period data write . . . 2385
Figure 715. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25 . . . 2388
Figure 716. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104 . . . 2388
Figure 717. ReadWait with SDMMC_CK < 50 MHz . . . 2389
Figure 718. ReadWait with SDMMC_CK > 50 MHz . . . 2390
Figure 719. CMD12 stream timing . . . 2392
Figure 720. CMD5 Sleep Awake procedure . . . 2394
Figure 721. Normal boot mode operation . . . 2396
Figure 722. Alternative boot mode operation . . . 2397
Figure 723. Command response R1b busy signaling . . . 2398
Figure 724. SDMMC state control . . . 2399
Figure 725. Card cycle power / power up diagram . . . 2400
Figure 726. Hardware flow timing . . . 2401
Figure 727. CMD11 signal voltage switch sequence . . . 2401
Figure 728. Voltage switch transceiver typical application . . . 2403
Figure 729. CAN subsystem . . . 2430
Figure 730. FDCAN block diagram . . . 2432
Figure 731. Transceiver delay measurement . . . 2437
Figure 732. Pin control in bus monitoring mode . . . 2439
Figure 733. Pin control in loop back mode . . . 2441
Figure 734. Message RAM configuration . . . 2442
Figure 735. Standard message ID filter path . . . 2445
Figure 736. Extended message ID filter path . . . 2446
Figure 737. Example of mixed configuration dedicated Tx buffers / Tx FIFO . . . 2452
Figure 738. Example of mixed configuration dedicated Tx buffers / Tx queue . . . 2452
Figure 739. Bit timing . . . 2454
Figure 740. Bypass operation . . . 2456
Figure 741. FSM calibration . . . 2457
Figure 742. Cycle time and global time synchronization . . . 2472
Figure 743. TTCAN level 0 and level 2 drift compensation . . . 2473
Figure 744. Level 0 schedule synchronization state machine . . . 2480
Figure 745. Level 0 master to slave relation . . . 2481

Figure 746. USB1 OTG high-speed block diagram (OTG_HS1) . . . 2567
Figure 747. USB2 OTG high-speed block diagram (OTG_HS2) . . . 2567
Figure 748. SOF connectivity (SOF trigger output to TIM and ITR1 connection) . . . 2578
Figure 749. Updating OTG_HFIR dynamically (RLDCTRL = 0) . . . 2581
Figure 750. Device-mode FIFO address mapping and AHB FIFO access mapping . . . 2582
Figure 751. Host-mode FIFO address mapping and AHB FIFO access mapping . . . 2583
Figure 752. Interrupt hierarchy . . . 2587
Figure 753. Transmit FIFO write task . . . 2681
Figure 754. Receive FIFO read task . . . 2682
Figure 755. Normal bulk/control OUT/SETUP . . . 2683
Figure 756. Bulk/control IN transactions . . . 2687
Figure 757. Normal interrupt OUT . . . 2690
Figure 758. Normal interrupt IN . . . 2695
Figure 759. Isochronous OUT transactions . . . 2697
Figure 760. Isochronous IN transactions . . . 2700
Figure 761. Normal bulk/control OUT/SETUP transactions - DMA . . . 2702
Figure 762. Normal bulk/control IN transaction - DMA . . . 2704
Figure 763. Normal interrupt OUT transactions - DMA mode . . . 2705
Figure 764. Normal interrupt IN transactions - DMA mode . . . 2706
Figure 765. Normal isochronous OUT transaction - DMA mode . . . 2707
Figure 766. Normal isochronous IN transactions - DMA mode . . . 2708
Figure 767. Receive FIFO packet read . . . 2714
Figure 768. Processing a SETUP packet . . . 2716
Figure 769. Bulk OUT transaction . . . 2722
Figure 770. TRDT max timing case . . . 2730
Figure 771. A-device SRP . . . 2731
Figure 772. B-device SRP . . . 2732
Figure 773. A-device HNP . . . 2733
Figure 774. B-device HNP . . . 2735
Figure 775. Ethernet high-level block diagram . . . 2743
Figure 776. DMA transmission flow (standard mode) . . . 2746
Figure 777. DMA transmission flow (OSP mode) . . . 2748
Figure 778. Receive DMA flow . . . 2750
Figure 779. Overview of MAC transmission flow . . . 2753
Figure 780. MAC reception flow . . . 2755
Figure 781. Packet filtering sequence . . . 2760
Figure 782. Networked time synchronization . . . 2767
Figure 783. Propagation delay calculation in clocks supporting peer-to-peer path correction . . . 2768
Figure 784. System time update using Fine method . . . 2773
Figure 785. TCP segmentation offload overview . . . 2780
Figure 786. Header and payload fields of segmented packets . . . 2782
Figure 787. Supported PHY interfaces . . . 2791
Figure 788. SMA Interface block . . . 2791
Figure 789. MDIO packet structure . . . 2792
Figure 790. Write data packet . . . 2793
Figure 791. Read data packet . . . 2794
Figure 792. Media independent interface (MII) signals . . . 2794
Figure 793. RMII block diagram . . . 2796
Figure 794. Transmission bit order . . . 2797
Figure 795. Receive bit order . . . 2798
Figure 796. Descriptor ring structure . . . 2815

Figure 797. DMA descriptor ring . . . 2816
Figure 798. Transmit descriptor (read format) . . . 2817
Figure 799. Transmit descriptor write-back format . . . 2821
Figure 800. Transmit context descriptor format . . . 2825
Figure 801. Receive normal descriptor (read format) . . . 2829
Figure 802. Receive normal descriptor (write-back format) . . . 2832
Figure 803. Receive context descriptor . . . 2839
Figure 804. Generation of ETH_DMAISR flags . . . 2857
Figure 805. HDMI-CEC block diagram . . . 2980
Figure 806. Message structure . . . 2980
Figure 807. Blocks . . . 2981
Figure 808. Bit timings . . . 2981
Figure 809. Signal free time . . . 2982
Figure 810. Arbitration phase . . . 2982
Figure 811. SFT of three nominal bit periods . . . 2982
Figure 812. Error bit timing . . . 2984
Figure 813. Error handling . . . 2985
Figure 814. TXERR detection . . . 2987
Figure 815. Block diagram of debug infrastructure . . . 2999
Figure 816. Power domains of debug infrastructure . . . 3001
Figure 817. Clock domains of debug infrastructure . . . 3002
Figure 818. SWD successful data transfer . . . 3005
Figure 819. JTAG TAP state machine . . . 3006
Figure 820. Debug and access port connections . . . 3017
Figure 821. APB-D CoreSight component topology . . . 3025
Figure 822. Global timestamp distribution . . . 3031
Figure 823. Embedded cross trigger . . . 3039
Figure 824. Mapping of trigger inputs to outputs . . . 3041
Figure 825. ETF state transition diagram . . . 3070
Figure 826. Cortex-M7 CoreSight Topology . . . 3140

1 Documentation conventions

1.1 General information

The STM32H743/753 and STM32H750 devices have an Arm®(a) Cortex®-M7 with FPU core

arm

1.2 List of abbreviations for registers

The following abbreviations(b) are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

b. This is an exhaustive list of all abbreviations applicable to STM microcontrollers, some of them may not be used in the current document.

1.3 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **Double word**: data of 64-bit length.
- Flash word: data of 256-bit length
- **IAP (in-application programming)**: IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming)**: ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes**: product configuration bits stored in the Flash memory.
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.
- **AXI**: Advanced extensible Interface protocol
- **PCROP**: proprietary code readout protection.
- **RDP**: readout protection

1.4 Availability of peripherals

For availability of peripherals and their number across all sales types, refer to the particular device datasheet.

1.5 Availability of security features

For security feature availability please refer to the table below:

Table 1. Availability of security features

Security feature	STM32H753xx	STM32H743xx
Embedded Flash memory (FLASH): – Flash Secure-only area	Available	Not available
Security memory management: – Secure access mode – Root secure services (RSS)		
Cryptographic processor (CRYP)		
Hash processor (HASH)		

2 Memory and bus architecture

2.1 System architecture

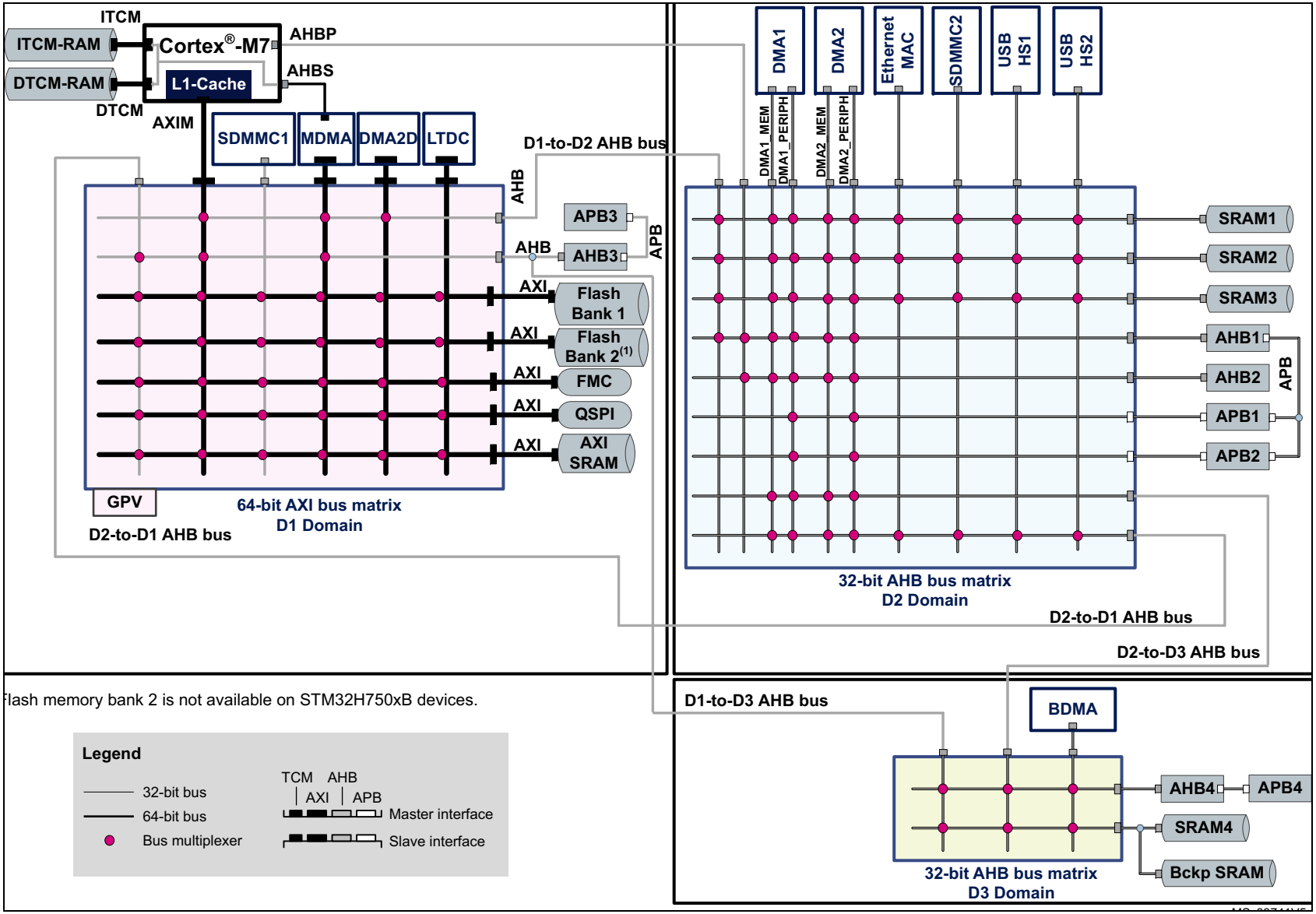
An AXI bus matrix, two AHB bus matrices and bus bridges allow interconnecting bus masters with bus slaves, as illustrated in *Table 2* and *Figure 1*.

Table 2. Bus-master-to-bus-slave interconnect

	Bus master / type[1]																	
	Cortex-M7 - AXIM	Cortex-M7 - AHBP	**Cortex-M7 - ITCM**	**Cortex-M7 -DTCM**	SDMMC1	**MDMA - AXI**	MDMA - AHBS	**DMA2D**	**LTDC**	DMA1 - MEM	DMA1 - PERIPH	DMA2 - MEM	DMA2 - PERIPH	Eth. MAC - AHB	SDMMC2 - AHB	USBHS1 - AHB	USBHS2 - AHB	BDMA - AHB
Bus slave / type[1]	**Interconnect path and type[2]**																	
ITCM	-	-	**X**	-	-	-	**X**	-	-	-	-	-	-	-	-	-	-	-
DTCM	-	-	-	**X**	-	-	X	-	-	-	-	-	-	-	-	-	-	-
AHB3 peripherals	X	-	-	-	-	X	-	-	-	X	X	X	X	X	X	X	X	-
APB3 peripherals	X	-	-	-	-	X	-	-	-	X	X	X	X	X	X	X	X	-
Flash bank 1	**X**	-	-	-	*X*	**X**	-	**X**	**X**	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
Flash bank 2[3]	**X**	-	-	-	*X*	**X**	-	**X**	**X**	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
AXI SRAM	**X**	-	-	-	*X*	**X**	-	**X**	**X**	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
QUADSPI	**X**	-	-	-	*X*	**X**	-	**X**	**X**	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
FMC	**X**	-	-	-	*X*	**X**	-	**X**	**X**	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
SRAM 1	X	-	-	-	-	X	-	X	-	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
SRAM 2	X	-	-	-	-	X	-	X	-	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
SRAM 3	X	-	-	-	-	X	-	X	-	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	-
AHB1 peripherals	-	X	-	-	-	X	-	X	-	X	X	X	X	-	-	-	-	-
APB1 peripherals	-	X	-	-	-	X	-	X	-	X	X	X	X	-	-	-	-	-
AHB2 peripherals	-	X	-	-	-	-	-	-	-	X	X	X	X	-	-	-	-	-
APB2 peripherals	-	X	-	-	-	X	-	X	-	X	X	X	X	-	-	-	-	-
AHB4 peripherals	X	-	-	-	-	X	-	-	-	X	X	X	X	X	X	X	X	*X*
APB4 peripherals	X	-	-	-	-	X	-	-	-	X	X	X	X	X	X	X	X	*X*
SRAM4	X	-	-	-	-	X	-	-	-	X	X	*X*	X	*X*	*X*	*X*	*X*	*X*
Backup RAM	X	-	-	-	-	X	-	-	-	X	X	*X*	X	*X*	*X*	*X*	*X*	*X*

1. **Bold** font type denotes 64-bit bus, plain type denotes 32-bit bus.
2. “X” = access possible, “-” = access not possible, shading = access useful/usable.
3. Bank 2 is not available on STM32H750xB devices.

Figure 1. System architecture for STM32H743/53xx and STM32H750xB devices



Flash memory bank 2 is not available on STM32H750xB devices.

2.1.1 Bus matrices

AXI bus matrix in D1 domain

The D1 domain multi AXI bus matrix ensures and arbitrates concurrent accesses from multiple masters to multiple slaves. This allows efficient simultaneous operation of high-speed peripherals.

The arbitration uses a round-robin algorithm with QoS capability.

Refer to *Section 2.2: AXI interconnect matrix (AXIM)* for more information on AXI interconnect.

AHB bus matrices in D2 and D3 domains

The AHB bus matrices in D2 and D3 domains ensure and arbitrate concurrent accesses from multiple masters to multiple slaves. This allows efficient simultaneous operation of high-speed peripherals.

The arbitration uses a round-robin algorithm.

2.1.2 TCM buses

The DTCM and ITCM (data and instruction tightly coupled RAMs) are connected through dedicated TCM buses directly to the Cortex-M7 core. The MDMA controller can access the DTCM and ITCM through AHBS, a specific CPU slave AHB. The ITCM is accessed by Cortex-M7 at CPU clock speed, with zero wait states.

2.1.3 Bus-to-bus bridges

To allow peripherals with different types of buses to communicate together, there is a number of bus-to-bus bridges in the system.

The AHB/APB bridges in D1 and D3 domains allow connecting peripherals on APB3 and APB4 to AHB3 and AHB4, respectively. The AHB/APB bridges in D2 domain allow peripherals on APB1 and APB2 to connect to AHB1. These AHB/APB bridges provide full synchronous interfacing, which allows the APB peripherals to operate with clocks independent of AHB that they connect to.

The AHB/APB bridges also allow APB1 and APB2 peripherals to connect to DMA1 and DMA2 peripheral buses, respectively, without transiting through AHB1.

The AHB/APB bridges convert 8-bit / 16-bit APB data to 32-bit AHB data, by replicating it to the three upper bytes / the upper half-word of the 32-bit word.

The AXI bus matrix incorporates AHB/AXI bus bridge functionality on its slave bus interfaces. The AXI/AHB bus bridges on its master interfaces marked as 32-bit in *Figure 1* are outside the matrix.

The Cortex-M7 CPU provides AHB/TCM-bus (ITCM and DTCM buses) translation from its AHBS slave AHB, allowing the MDMA controller to access the ITCM and DTCM.

2.1.4 Inter-domain buses

D2-to-D1 AHB

This 32-bit bus connects the D2 domain to the AXI bus matrix in the D1 domain. It allows bus masters in the D2 domain to access resources (bus slaves) in the D1 domain and indirectly, via the D1-to-D3 AHB, in the D3 domain.

D1-to-D2 AHB

This 32-bit bus connects the D1 domain to the D2 domain AHB bus matrix. It allows bus masters in the D1 domain to access resources (bus slaves) in the D2 domain.

D1-to-D3 AHB

This 32-bit bus connects the D1 domain to the D3 domain AHB bus matrix. It allows bus masters in the D1 domain to access resources (bus slaves) in the D3 domain.

D2-to-D3 AHB

This 32-bit bus connects the D2 domain to the D3 domain AHB bus matrix. It allows bus masters in the D2 domain to access resources (bus slaves) in the D3 domain.

2.1.5 CPU buses

Cortex®-M7 AXIM bus

The Cortex®-M7 CPU uses the 64-bit AXIM bus to access all memories (excluding ITCM, and DTCM) and AHB3, AHB4, APB3 and APB4 peripherals (excluding AHB1, APB1 and APB2 peripherals).

The AXIM bus connects the CPU to the AXI bus matrix in the D1 domain.

Cortex®-M7 ITCM bus

The Cortex®-M7 CPU uses the 64-bit ITCM bus for fetching instructions from and accessing data in the ITCM.

Cortex®-M7 DTCM bus

The Cortex®-M7 CPU uses the 2x32-bit DTCM bus for accessing data in the DTCM. The 2x32-bit DTCM bus allows load/load and load/store instruction pairs to be dual-issued on the DTCM memory. It can also fetch instructions.

Cortex®-M7 AHBS bus

The Cortex®-M7 CPU uses the 32-bit AHBS slave bus to allow the MDMA controller to access the ITCM and the DTCM.

Cortex®-M7 AHBP bus

The Cortex®-M7 CPU uses the 32-bit AHBP bus for accessing AHB1, AHB2, APB1 and APB2 peripherals via the AHB bus matrix in the D2 domain.

2.1.6 Bus master peripherals

SDMMC1

The SDMMC1 uses a 32-bit bus, connected to the AXI bus matrix, through which it can access internal AXI SRAM and Flash memories, and external memories through the Quad-SPI controller and the FMC.

SDMMC2

The SDMMC2 uses a 32-bit bus, connected to the AHB bus matrix in D2 domain. Through the system bus matrices, it can access the internal AXI SRAM, SRAM1, SRAM2, SRAM3 and Flash memories, and external memories through the Quad-SPI controller and the FMC.

MDMA controller

The MDMA controller has two bus masters: an AXI 64-bit bus, connected to the AXI bus matrix and an AHB 32-bit bus connected to the Cortex-M7 AHBS slave bus.

The MDMA is optimized for DMA data transfers between memories since it supports linked list transfers that allow performing a chained list of transfers without the need for CPU intervention. Through the system bus matrices and the Cortex-M7 AHBS slave bus, the MDMA can access all internal and external memories through the Quad-SPI controller and the FMC.

DMA1 and DMA2 controllers

The DMA1 and DMA2 controllers have two 32-bit buses - memory bus and peripheral bus, connected to the AHB bus matrix in D2 domain.

The memory bus allows DMA data transfers between memories. Through the system bus matrices, the memory bus can access all internal memories except ITCM and DTCM, and external memories through the Quad-SPI controller and the FMC.

The peripheral bus allows DMA data transfers between two peripherals, between two memories or between a peripheral and a memory. Through the system bus matrices, the peripheral bus can access all internal memories except ITCM and DTCM, external memories through the Quad-SPI controller and the FMC, and all AHB and APB peripherals. A direct access to APB1 and APB2 is available, without passing through AHB1. Direct path to APB1 and APB2 bridges allows reducing the bandwidth usage on AHB1 bus by improving data treatment efficiency for APB and AHB peripherals.

BDMA controller

The BDMA controller uses a 32-bit bus, connected to the AHB bus matrix in D3 domain, for DMA data transfers between two peripherals, between two memories or between a peripheral and a memory. BDMA transfers are limited to the D3 domain resources. It can access the internal SRAM4, backup RAM, and AHB4 and APB4 peripherals through the AHB bus matrix in the D3 domain.

Chrom-Art Accelerator™ (DMA2D)

The DMA2D graphics accelerator uses a 64-bit bus, connected to the AXI bus matrix. Through the system bus matrices, internal AXI SRAM, SRAM1, SRAM2, SRAM3 and Flash memories, and external memories through the Quad-SPI controller and the FMC.

LCD-TFT controller (LTDC)

The LCD-TFT display controller, LTDC, uses a 64-bit bus, connected to the AXI bus matrix, through which it can access internal AXI SRAM and Flash memories, and external memories through the Quad-SPI controller and the FMC.

Ethernet MAC

The Ethernet MAC uses a 32-bit bus, connected to the AHB bus matrix in the D2 domain. Through the system bus matrices, it can access all internal memories except ITCM and DTCM, and external memories through the Quad-SPI controller and the FMC.

USBHS1 and USBHS2 peripherals

The USBHS1 and USBHS2 peripherals use 32-bit buses, connected to the AHB bus matrix in the D2 domain. Through the system bus matrices, they can access all internal memories except ITCM and DTCM, and external memories through the Quad-SPI controller and the FMC.

2.1.7 Clocks to functional blocks

Upon reset, clocks to blocks such as peripherals and some memories are disabled (except for the SRAM, DTCM, ITCM and Flash memory). To operate a block with no clock upon reset, the software must first enable its clock through RCC_AHBxENR or RCC_APBxENR register, respectively.

2.2 AXI interconnect matrix (AXIM)

2.2.1 AXI introduction

The AXI (advanced extensible interface) interconnect is based on the Arm® CoreLink™ NIC-400 Network Interconnect. The interconnect has six initiator ports, or ASIBs (AMBA slave interface blocks), and seven target ports, or AMIBs (AMBA master interface blocks). The ASIBs are connected to the AMIBs via an AXI switch matrix.

Each ASIB is a slave on an AXI bus or AHB (advanced high-performance bus). Similarly, each AMIB is a master on an AXI or AHB bus. Where an ASIB or AMIB is connected to an AHB, it converts between the AHB and the AXI protocol.

The AXI interconnect includes a GPV (global programmer view) which contains registers for configuring certain parameters, such as the QoS (quality of service) level at each ASIB.

Any accesses to unallocated address space are handled by the default slave, which generates the return signals. This ensures that such transactions complete and do not block the issuing master and ASIB.

2.2.2 AXI interconnect main features

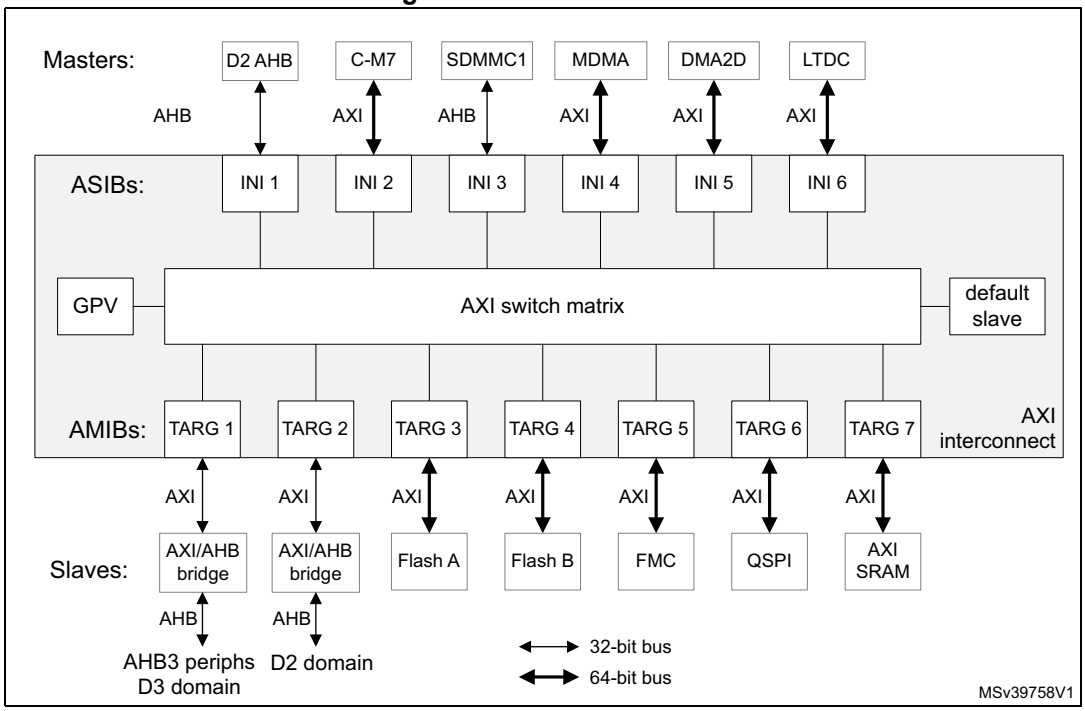
- 64-bit AXI bus switch matrix with six ASIBs and seven AMIBs, in D1 domain
- AHB/AXI bridge function built into the ASIBs
- concurrent connectivity of multiple ASIBs to multiple AMIBs
- programmable traffic priority management (QoS - quality of service)
- software-configurable via GPV

2.2.3 AXI interconnect functional description

Block diagram

The AXI interconnect is shown in *Figure 2*.

Figure 2. AXI interconnect



ASIB configuration

Table 3 summarizes the characteristics of the ASIBs.

Table 3. ASIB configuration

ASIB	Connected master	Protocol	Bus width	R/W issuing
INI 1	AHB from D2 domain	AHB-lite	32	1/4
INI 2	Cortex-M7	AXI4	64	7/32
INI 3	SDMMC1	AHB-lite	32	1/4
INI 4	MDMA	AXI4	64	4/1
INI 5	DMA2D	AXI4	64	2/1
INI 6	LTDC	AXI4	64	1/1

AMIB configuration

Table 4 summarizes the characteristics of the AMIBs.

Table 4. AMIB configuration

AMIB	Connected slave	Protocol	Bus width	R/W/Total acceptance
TARG 1	Peripheral 3 and D3 AHB	AXI4(1)	32	1/1/1
TARG 2	D2 AHB	AXI4(1)	32	1/1/1
TARG 3	Flash A	AXI4	64	3/2/5
TARG 4	Flash B	AXI4	64	3/2/5
TARG 5	FMC	AXI4	64	3/3/6
TARG 6	QUADSPI	AXI4	64	2/1/3
TARG 7	AXI SRAM	AXI3	64	2/2/2

1. Conversion to AHB protocol is done via an AXI/AHB bridge sitting between AXI interconnect and the connected slave.

Quality of service (QoS)

The AXI switch matrix uses a priority-based arbitration when two ASIB simultaneously attempt to access the same AMIB. Each ASIB has programmable read channel and write channel priorities, known as QoS, from 0 to 15, such that the higher the value, the higher the priority. The read channel QoS value is programmed in the *AXI interconnect - INI x read QoS register (AXI_INIx_READ_QOS)*, and the write channel in the *AXI interconnect - INI x write QoS register (AXI_INIx_WRITE_QOS)*. The default QoS value for all channels is 0 (lowest priority).

If two coincident transactions arrive at the same AMIB, the higher priority transaction passes before the lower priority. If the two transactions have the same QoS value, then a least-recently-used (LRU) priority scheme is adopted.

The QoS values should be programmed according to the latency requirements for the application. Setting a higher priority for an ASIB ensures a lower latency for transactions initiated by the associated bus master. This can be useful for real-time-constrained tasks, such as graphics processing (LTDC, DMA2D). Assigning a high priority to masters that can make many and frequent accesses to the same slave (such as the Cortex-M7 CPU) can block access to that slave by other lower-priority masters.

Global programmer view (GPV)

The GPV contains configuration registers for the AXI interconnect (see *Section 2.2.4*). These registers are only accessible by the Cortex-M7 CPU.

2.2.4 AXI interconnect registers

AXI interconnect - peripheral ID4 register (AXI_PERIPH_ID_4)

Address offset: 0x1FD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r				r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: N/A

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm®

AXI interconnect - peripheral ID0 register (AXI_PERIPH_ID_0)

Address offset: 0x1FE0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Peripheral part number bits 0 to 7
0x00: Part number = 0x400

AXI interconnect - peripheral ID1 register (AXI_PERIPH_ID_1)

Address offset: 0x1FE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r				r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity bits 0 to 3
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Peripheral part number bits 8 to 11
0x4: Part number = 0x400

AXI interconnect - peripheral ID2 register (AXI_PERIPH_ID_2)

Address offset: 0x1FE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r				r	r		

Bits 7:4 **REVISION[3:0]**: Peripheral revision number
0x2: r0p2

Bit 3 **JEDEC:** JEP106 code flag
0x1: JEDEC allocated code

Bits 2:0 **JEP106ID[6:4]**: JEP106 Identity bits 4 to 6
0x3: Arm® JEDEC code

AXI interconnect - peripheral ID3 register (AXI_PERIPH_ID_3)

Address offset: 0x1FEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_AND[3:0]				CUST_MOD_NUM[3:0]			
								r				r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REV_AND[3:0]**: Customer version
0: None

Bits 3:0 **CUST_MOD_NUM[3:0]**: Customer modification
0: None

AXI interconnect - component ID0 register (AXI_COMP_ID_0)

Address offset: 0x1FF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE7:0]**: Preamble bits 0 to 7
0xD: Common ID value

AXI interconnect - component ID1 register (AXI_COMP_ID_1)

Address offset: 0x1FF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r				r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component class
0xF: Generic IP component class

Bits 3:0 **PREAMBLE[11:8]**: Preamble bits 8 to 11
0x0: Common ID value

AXI interconnect - component ID2 register (AXI_COMP_ID_2)

Address offset: 0x1FF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Preamble bits 12 to 19
0x05: Common ID value

AXI interconnect - component ID3 register (AXI_COMP_ID_3)

Address offset: 0x1FFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r							

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Preamble bits 20 to 27
0xB1: Common ID value

AXI interconnect - TARG x bus matrix issuing functionality register (AXI_TARGx_FN_MOD_ISS_BM)

Address offset: 0x1008 + 0x1000 * x, where x = 1 to 7

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE**: Switch matrix write issuing override for target
0: Normal issuing capability
1: Set switch matrix write issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE**: Switch matrix read issuing override for target
0: Normal issuing capability
1: Set switch matrix read issuing capability to 1

AXI interconnect - TARG x bus matrix functionality 2 register (AXI_TARGx_FN_MOD2)

Address offset: 0x1024 + 0x1000 * x, where x = 1, 2 and 7

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **BYPASS_MERGE**: Disable packing of beats to match the output data width. Unaligned transactions are not realigned to the input data word boundary.

0: Normal operation

1: Disable packing

AXI interconnect - TARG x long burst functionality modification register (AXI_TARGx_FN_MOD_LB)

Address offset: 0x102C + 0x1000 * x, where x = 1 and 2

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD_LB
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **FN_MOD_LB**: Controls burst breaking of long bursts

0: Long bursts can not be generated at the output of the ASIB

1: Long bursts can be generated at the output of the ASIB

AXI interconnect - TARG x issuing functionality modification register (AXI_TARGx_FN_MOD)

Address offset: 0x1108 + 0x1000 * x, where x = 1, 2 and 7

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE**: Override AMIB write issuing capability
- 0: Normal issuing capability
- 1: Force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE**: Override AMIB read issuing capability
- 0: Normal issuing capability
- 1: Force issuing capability to 1

AXI interconnect - INI x functionality modification 2 register (AXI_INIx_FN_MOD2)

Address offset: 0x41024 + 0x1000 * x, where x = 1 and 3

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **BYPASS_MERGE**: Disables alteration of transactions by the up-sizer unless required by the protocol
- 0: Normal operation
- 1: Transactions pass through unaltered where allowed

AXI interconnect - INI x AHB functionality modification register (AXI_INIx_FN_MOD_AHB)

Address offset: 0x41028 + 0x1000 * x, where x = 1 and 3

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WR_INC_OVERRIDE**: Converts all AHB-Lite read transactions to a series of single beat AXI transactions.

0: Override disabled

1: Override enabled

Bit 0 **RD_INC_OVERRIDE**: Converts all AHB-Lite write transactions to a series of single beat AXI transactions, and each AHB-Lite write beat is acknowledged with the AXI buffered write response.

0: Override disabled

1: Override enabled

AXI interconnect - INI x read QoS register (AXI_INIx_READ_QOS)

Address offset: 0x41100 + 0x1000 * x, where x = 1 to 76

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AR_QOS[3:0]**: Read channel QoS setting

0x0: Lowest priority

0xF: Highest priority

AXI interconnect - INI x write QoS register (AXI_INIx_WRITE_QOS)

Address offset: 0x41104 + 0x1000 * x, where x = 1 to 76

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AW_QOS[3:0]**: Write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

AXI interconnect - INI x issuing functionality modification register (AXI_INIx_FN_MOD)

Address offset: 0x41108 + 0x1000 * x, where x = 1 to 76

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE**: Override ASIB write issuing capability
0: Normal issuing capability
1: Force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE**: Override ASIB read issuing capability
0: Normal issuing capability
1: Force issuing capability to 1

2.2.5 AXI interconnect register map

Table 5. AXI interconnect register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1FD0	AXI_PERIPH_ID_4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0x1FD4	AXI_PERIPH_ID_5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Reserved							
	Reset value																									0	0	0	0	0	0	0	0
0x1FD8	AXI_PERIPH_ID_6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Reserved							
	Reset value																									0	0	0	0	0	0	0	0
0x1FDC	AXI_PERIPH_ID_7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Reserved							
	Reset value																									0	0	0	0	0	0	0	0
0x1FE0	AXI_PERIPH_ID_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1FE4	AXI_PERIPH_ID_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0x1FE8	AXI_PERIPH_ID_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0x1FEC	AXI_PERIPH_ID_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_AND[3:0]				CUST_MOD_NUM [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x1FF0	AXI_COMP_ID_0																									PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0x1FF4	AXI_COMP_ID_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	1	0	0	0	0

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1FF8	AXI_COMP_ID_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0x1FFC	AXI_COMP_ID_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1
0x2000 - 0x2004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x2008	AXI_TARG1_FN_MOD_ISS_BM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x200C - 0x2020	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x2024	AXI_TARG1_FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x2028	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x202C	AXI_TARG1_FN_MOD_LB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD_LB
	Reset value																																0
0x2030 - 0x2104	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x2108	AXI_TARG1_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x210C - 0x3004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3008	AXI_TARG2_ FN_MOD_ ISS_BM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x300C - 0x3020	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x3024	AXI_TARG2_ FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x3028	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x302C	AXI_TARG2_ FN_MOD_LB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD_LB
	Reset value																																0
0x3030 - 0x3104	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x3108	AXI_TARG2_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x310C - 0x4004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x4008	AXI_TARG3_ FN_MOD_ ISS_BM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x400C - 0x5004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5008	AXI_TARG4_ FN_MOD_ ISS_BM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x500C - 0x6004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x6008	AXI_TARG5_ FN_MOD_ ISS_BM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x600C - 0x7004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x7008	AXI_TARG6_ FN_MOD_ ISS_BM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x700C - 0x8004	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x8008	AXI_TARG7_ FN_MOD_ ISS_BM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x800C - 0x8020	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x8024	AXI_TARG7_ FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8028 - 0x8104	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x8108	AXI_TARG7_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x810C- 0x42020	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x42024	AXI_INI1_ FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x42028	AXI_INI1_FN_ MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4202C- 0x420FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x42100	AXI_INI1_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	0	0
0x42104	AXI_INI1_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	0	0
0x42108	AXI_INI1_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4210C- 0x430FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43100	AXI_INI2_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	0	0
0x43104	AXI_INI2_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	0	0
0x43108	AXI_INI2_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4310C - 0x44020	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x44024	AXI_INI3_ FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x44028	AXI_INI3_ FN_MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4402C- 0x440FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x44100	AXI_INI3_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	0	0
0x44104	AXI_INI3_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	0	0

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44108	AXI_INI3_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4410C-0x450FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x45100	AXI_INI4_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	0	0
0x45104	AXI_INI4_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	0	0
0x45108	AXI_INI4_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4510C-0x460FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x46100	AXI_INI5_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	0	0
0x46104	AXI_INI5_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	0	0
0x46108	AXI_INI5_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4610C-0x470FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x47100	AXI_INI6_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	0	0

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x47104	AXI_INI6_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	0	0
0x47108	AXI_INI6_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

2.3 Memory organization

2.3.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

The addressable memory space is divided into eight main blocks, of 512 Mbytes each.

2.3.2 Memory map and register boundary addresses

Table 6. Memory map and default device memory area attributes

Region	Boundary address	Arm® Cortex®-M7	Type	Attributes	Execute never
External Devices	0xD0000000 - 0xDFFFFFFF	FMC SDRAM Bank2 (or Reserved in case of FMC remap)	Device	-	Yes
	0xCC000000 - 0xCFFFFFFF	FMC SDRAM Bank1 (or remap of FMC NOR/PSRAM/SRAM 4 Bank1)			
	0xC8000000 - 0xCBFFFFFF	FMC SDRAM Bank1 (or remap of FMC NOR/PSRAM/SRAM 3 Bank1)			
	0xC4000000 - 0xC7FFFFFF	FMC SDRAM Bank1 (or remap of FMC NOR/PSRAM/SRAM 2 Bank1)			
	0xC0000000 - 0xC3FFFFFF	FMC SDRAM Bank1 (or remap of FMC NOR/PSRAM/SRAM 1 Bank1)			
	0xA0000000 - 0xBFFFFFFF	Reserved			
External Memories	0x90000000 - 0x9FFFFFFF	QUADSPI	Normal	Write-through cache attribute	No
	0x80000000 - 0x8FFFFFFF	FMC NAND Flash memory			
	0x70000000 - 0x7FFFFFFF	Reserved (or remap of FMC SDRAM Bank2)		Write-back, write allocate cache attribute	
	0x6C000000 - 0x6FFFFFFF	FMC NOR/PSRAM/SRAM 4 Bank1 (or remap of FMC SDRAM Bank1)			
	0x68000000 - 0x6BFFFFFF	FMC NOR/PSRAM/SRAM 3 Bank1 (or remap of FMC SDRAM Bank1)			
	0x64000000 - 0x67FFFFFF	FMC NOR/PSRAM/SRAM 2 Bank1 (or remap of FMC SDRAM Bank1)			
	0x60000000 - 0x63FFFFFF	FMC NOR/PSRAM/SRAM 1 Bank1 (or remap of FMC SDRAM Bank1)			
Peripherals	0x40000000 - 0x5FFFFFFF	Peripherals (refer to *Table 7: Register boundary addresses*)	Device	-	Yes

Table 6. Memory map and default device memory area attributes

Region	Boundary address	Arm® Cortex®-M7	Type	Attributes	Execute never
RAM	0x38801000 - 0x3FFFFFFF	Reserved	Normal	Write-back, write allocate cache attribute	No
	0x38800000 - 0x38800FFF	Backup SRAM			
	0x38010000 - 0x387FFFFF	Reserved			
	0x38000000 - 0x3800FFFF	SRAM4			
	0x30048000 - 0x37FFFFFF	Reserved			
	0x30040000 - 0x30047FFF	SRAM3			
	0x30020000 - 0x3003FFFF	SRAM2			
	0x30000000 - 0x3001FFFF	SRAM1			
	0x24080000 - 0x2FFFFFFF	Reserved			
	0x24000000 - 0x2407FFFF	AXI SRAM			
	0x20020000 - 0x23FFFFFF	Reserved			
	0x20000000 - 0x2001FFFF	DTCM			
Code	0x1FF20000 - 0x1FFFFFFF	Reserved	Normal	Write-through cache attribute	No
	0x1FF00000 - 0x1FF1FFFF	System Memory			
	0x08200000 - 0x1FEFFFFF	Reserved			
	0x08100000 - 0x081FFFFF	Flash memory bank 2			
	0x08000000 - 0x080FFFFF	Flash memory bank 1			
	0x00010000 - 0x07FFFFFF	Reserved			
	0x00000000 - 0x0000FFFF	ITCM			

All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and register areas, refer to the following table.

The following table gives the boundary addresses of the peripherals available in the devices.

Table 7. Register boundary addresses(1)

Boundary address	Peripheral	Bus	Register map
0x58026400 - 0x580267FF	HSEM	AHB4 (D3)	*Section 9.4: HSEM registers*
0x58026000 - 0x580263FF	ADC3		*Section 24.6: ADC common registers*
0x58025800 - 0x58025BFF	DMAMUX2		*Section 16.6: DMAMUX registers*
0x58025400 - 0x580257FF	BDMA		*Section 15.6: BDMA registers*
0x58024C00 - 0x58024FFF	CRC		*Section 20.4: CRC registers*
0x58024800 - 0x58024BFF	PWR		*Section 5.8: PWR register description*
0x58024400 - 0x580247FF	RCC		*Section 7.7: RCC register description*
0x58022800 - 0x58022BFF	GPIOK		*Section 10.4: GPIO registers*
0x58022400 - 0x580227FF	GPIOJ		*Section 10.4: GPIO registers*
0x58022000 - 0x580223FF	GPIOI		*Section 10.4: GPIO registers*
0x58021C00 - 0x58021FFF	GPIOH		*Section 10.4: GPIO registers*
0x58021800 - 0x58021BFF	GPIOG		*Section 10.4: GPIO registers*
0x58021400 - 0x580217FF	GPIOF		*Section 10.4: GPIO registers*
0x58021000 - 0x580213FF	GPIOE		*Section 10.4: GPIO registers*
0x58020C00 - 0x58020FFF	GPIOD		*Section 10.4: GPIO registers*
0x58020800 - 0x58020BFF	GPIOC		*Section 10.4: GPIO registers*
0x58020400 - 0x580207FF	GPIOB		*Section 10.4: GPIO registers*
0x58020000 - 0x580203FF	GPIOA		*Section 10.4: GPIO registers*

Table 7. Register boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Register map
0x58005800 - 0x58006BFF	Reserved	APB4 (D3)	Reserved
0x58005400 - 0x580057FF	SAI4		*Section 50.5: SAI registers*
0x58004C00 - 0x58004FFF	Reserved		Reserved
0x58004800 - 0x58004BFF	IWDG1		*Section 44.4: IWDG registers*
0x58004000 - 0x580043FF	RTC & BKP registers		*Section 45.6: RTC registers*
0x58003C00 - 0x58003FFF	VREF		*Section 26.3: VREFBUF registers*
0x58003800 - 0x58003BFF	COMP1 - COMP2		*Section 27.7: COMP registers*
0x58003000 - 0x580033FF	LPTIM5		*Section 42.7: LPTIM registers*
0x58002C00 - 0x58002FFF	LPTIM4		*Section 42.7: LPTIM registers*
0x58002800 - 0x58002BFF	LPTIM3		*Section 42.7: LPTIM registers*
0x58002400 - 0x580027FF	LPTIM2		*Section 42.7: LPTIM registers*
0x58001C00 - 0x58001FFF	I2C4		*Section 46.7: I2C registers*
0x58001400 - 0x580017FF	SPI6		*Section 49.11: SPI/I2S registers*
0x58000C00 - 0x58000FFF	LPUART1		*Section 48.5: LPUART registers*
0x58000400 - 0x580007FF	SYSCFG		*Section 11.3: SYSCFG registers*
0x58000000 - 0x580003FF	EXTI		*Section 19.6: EXTI registers*
0x52008000 - 0x52008FFF	Delay Block SDMMC1	AHB3 (D1)	*Section 23.4: DLYB registers*
0x52007000 - 0x52007FFF	SDMMC1		*Section 54.8: SDMMC registers*
0x52006000 - 0x52006FFF	Delay Block QUADSPI		*Section 23.4: DLYB registers*
0x52005000 - 0x52005FFF	QUADSPI control registers		*Section 22.5: QUADSPI registers*
0x52004000 - 0x52004FFF	FMC control registers		*Section 21.7.6: NOR/PSRAM controller registers*, *Section 21.8.7: NAND Flash controller registers*, *Section 21.9.5: SDRAM controller registers*
0x52003000 - 0x52003FFF	JPEG		*Section 32.5: JPEG codec registers*
0x52002000 - 0x52002FFF	Flash interface registers		*Section 3.9: FLASH registers*
0x52001000 - 0x52001FFF	Chrom-Art (DMA2D)		*Section 17.6: DMA2D registers*
0x52000000 - 0x52000FFF	MDMA		*Section 13.5: MDMA registers*
0x51000000 - 0x510FFFFF	GPV		*Section 2.2: AXI interconnect matrix (AXIM)*
0x50003000 - 0x50003FFF	WWDG1	APB3 (D1)	*Section 43.4: WWDG registers*
0x50001000 - 0x50001FFF	LTDC		*Section 31.7: LTDC registers*
0x50000000 - 0x50000FFF	Reserved		-
0x48022800 - 0x48022BFF	Delay Block SDMMC2	AHB2 (D2)	*Section 23.4: DLYB registers*

Table 7. Register boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Register map
0x48022400 - 0x480227FF	SDMMC2		*Section 54.8: SDMMC registers*
0x48021800 - 0x48021BFF	RNG		*Section 33.8: RNG registers*
0x48021400 - 0x480217FF	HASH		*Section 35.6: HASH registers*
0x48021000 - 0x480213FF	CRYPTO		*Section 34.6: CRYP registers*
0x48020000 - 0x480203FF	DCMI		*Section 30.7: DCMI register description*
0x40080000 - 0x400BFFFF	USB2 OTG_FS	AHB1 (D2)	*Section 56.14: OTG_HS registers*
0x40040000 - 0x4007FFFF	USB1 OTG_HS		*Section 56.14: OTG_HS registers*
0x40028000 - 0x400293FF	ETHERNET MAC		*Section 57.11: Ethernet registers*
0x40024400 - 0x400247FF	Reserved		Reserved
0x40022000 - 0x400223FF	ADC1 - ADC2		*Section 24.6: ADC common registers*
0x40020800 - 0x40020BFF	DMAMUX1		*Section 16.6: DMAMUX registers*
0x40020400 - 0x400207FF	DMA2		*Section 14.5: DMA registers*
0x40020000 - 0x400203FF	DMA1		*Section 14.5: DMA registers*
0x40017400 - 0x400177FF	HRTIM	APB2 (D2)	*Section 36.5: HRTIM registers*
0x40017000 - 0x400173FF	DFSDM1		*Section 29.7: DFSDM channel y registers (y=0..7)*, *Section 29.8: DFSDM filter x module registers (x=0..3)*
0x40016000 - 0x400163FF	SAI3		*Section 50.5: SAI registers*
0x40015C00 - 0x40015FFF	SAI2		*Section 50.5: SAI registers*
0x40015800 - 0x40015BFF	SAI1		*Section 50.5: SAI registers*
0x40015000 - 0x400153FF	SPI5		*Section 49.11: SPI/I2S registers*
0x40014800 - 0x40014BFF	TIM17		*Section 40.6: TIM16/TIM17 registers*
0x40014400 - 0x400147FF	TIM16		*Section 40.6: TIM16/TIM17 registers*
0x40014000 - 0x400143FF	TIM15		*Section 40.5: TIM15 registers*
0x40013400 - 0x400137FF	SPI4		*Section 49.11: SPI/I2S registers*
0x40013000 - 0x400133FF	SPI1 / I2S1		*Section 49.11: SPI/I2S registers*
0x40011400 - 0x400117FF	USART6		*Section 47.7: USART registers*
0x40011000 - 0x400113FF	USART1		*Section 47.7: USART registers*
0x40010400 - 0x400107FF	TIM8		*Section 37.4: TIM1/TIM8 registers*
0x40010000 - 0x400103FF	TIM1		*Section 37.4: TIM1/TIM8 registers*

Table 7. Register boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Register map
0x4000AC00 - 0x4000D3FF	CAN Message RAM	APB1 (D2)	*Section 55.4: FDCAN registers*
0x4000A800 - 0x4000ABFF	CAN CCU		*Section 55.4: FDCAN registers*
0x4000A400 - 0x4000A7FF	FDCAN2		*Section 55.4: FDCAN registers*
0x4000A000 - 0x4000A3FF	FDCAN1		*Section 55.4: FDCAN registers*
0x40009400 - 0x400097FF	MDIOS		*Section 53.4: MDIOS registers*
0x40009000 - 0x400093FF	OPAMP		*Section 28.6: OPAMP registers*
0x40008800 - 0x40008BFF	SWPMI		*Section 52.6: SWPMI registers*
0x40008400 - 0x400087FF	CRS		*Section 8.7: CRS registers*
0x40007C00 - 0x40007FFF	UART8		*Section 47.7: USART registers*
0x40007800 - 0x40007BFF	UART7		*Section 47.7: USART registers*
0x40007400 - 0x400077FF	DAC1		*Section 25.7: DAC registers*
0x40006C00 - 0x40006FFF	HDMI-CEC		*Section 58.7: HDMI-CEC registers*
0x40005C00 - 0x40005FFF	I2C3		*Section 46.7: I2C registers*
0x40005800 - 0x40005BFF	I2C2		*Section 46.7: I2C registers*
0x40005400 - 0x400057FF	I2C1		*Section 46.7: I2C registers*
0x40005000 - 0x400053FF	UART5		*Section 47.7: USART registers*
0x40004C00 - 0x40004FFF	UART4		*Section 47.7: USART registers*
0x40004800 - 0x40004BFF	USART3		*Section 47.7: USART registers*
0x40004400 - 0x400047FF	USART2		*Section 47.7: USART registers*
0x40004000 - 0x400043FF	SPDIFRX1		*Section 51.5: SPDIFRX interface registers*
0x40003C00 - 0x40003FFF	SPI3 / I2S3		*Section 49.11: SPI/I2S registers*
0x40003800 - 0x40003BFF	SPI2 / I2S2		*Section 49.11: SPI/I2S registers*
0x40002C00 - 0x40002FFF	Reserved		Reserved
0x40002400 - 0x400027FF	LPTIM1		*Section 42.7: LPTIM registers*
0x40002000 - 0x400023FF	TIM14		*Section 38.4: TIM2/TIM3/TIM4/TIM5 registers*
0x40001C00 - 0x40001FFF	TIM13		*Section 38.4: TIM2/TIM3/TIM4/TIM5 registers*
0x40001800 - 0x40001BFF	TIM12		*Section 38.4: TIM2/TIM3/TIM4/TIM5 registers*
0x40001400 - 0x400017FF	TIM7		*Section 41.4: TIM6/TIM7 registers*
0x40001000 - 0x400013FF	TIM6		*Section 41.4: TIM6/TIM7 registers*
0x40000C00 - 0x40000FFF	TIM5		*Section 38.4: TIM2/TIM3/TIM4/TIM5 registers*
0x40000800 - 0x40000BFF	TIM4		*Section 38.4: TIM2/TIM3/TIM4/TIM5 registers*
0x40000400 - 0x400007FF	TIM3		*Section 38.4: TIM2/TIM3/TIM4/TIM5 registers*
0x40000000 - 0x400003FF	TIM2		*Section 38.4: TIM2/TIM3/TIM4/TIM5 registers*

1. Accessing a reserved area results in a bus error. Accessing undefined memory space in a peripheral returns zeros.

2.4 Embedded SRAM

The STM32H743/53xx and STM32H750xB devices feature:

- Up to 864 Kbytes of System SRAM
- 128 Kbytes of data TCM RAM
- 64 Kbytes of instruction TCM RAM
- 4 Kbytes of backup SRAM

The embedded system SRAM is divided into up to five blocks over the three power domains:

- D1 domain, AXI SRAM:
  - AXI SRAM is mapped at address 0x2400 0000 and accessible by all system masters except BDMA through D1 domain AXI bus matrix. AXI SRAM can be used for application data which are not allocated in DTCM RAM or reserved for graphic objects (such as frame buffers)
- D2 domain, AHB SRAM:
  - AHB SRAM1 is mapped at address 0x3000 0000 and accessible by all system masters except BDMA through D2 domain AHB matrix. AHB SRAM1 can be used as DMA buffers to store peripheral input/output data in D2 domain, or as code location for Cortex®-M4 CPU (application code available when D1 is powered off.
  - AHB SRAM2 is mapped at address 0x3002 0000 and accessible by all system masters except BDMA through D2 domain AHB matrix. AHB SRAM2 can be used as DMA buffers to store peripheral input/output data in D2 domain, or as read-write segment for application running on Cortex®-M4 CPU.
  - AHB SRAM3 is mapped at address 0x3004 0000 and accessible by all system masters except BDMA through D2 domain AHB matrix. AHB SRAM3 can be used as buffers to store peripheral input/output data for Ethernet and USB,.
- D3 domain, AHB SRAM:
  - AHB SRAM4 is mapped at address 0x3800 0000 and accessible by most of system masters through D3 domain AHB matrix. AHB SRAM4 can be used as BDMA buffers to store peripheral input/output data in D3 domain. It can also be used to retain some application data when D1 and D2 domain enter DStandby mode.

The system AHB SRAM can be accessed as bytes, half-words (16-bit units) or words (32-bit units), while the system AXI SRAM can be accessed as bytes, half-words, words or double-words (64-bit units). These memories can be addressed at maximum system clock frequency without wait state.

The AHB masters can read/write-access an SRAM section concurrently with the Ethernet MAC or the USB OTG HS peripheral accessing another SRAM section. For example, the Ethernet MAC accesses the SRAM2 while the CPU accesses the SRAM1, concurrently.

The TCM SRAMs are dedicated to the Cortex®-M7:

- DTCM-RAM on TCM interface is mapped at the address 0x2000 0000 and accessible by Cortex®-M7, and by MDMA through AHBS slave bus of the Cortex®-M7 CPU. The DTCM-RAM can be used as read-write segment to host critical real-time data (such as stack and heap) for application running on Cortex®-M7 CPU.
- ITCM-RAM on TCM interface mapped at the address 0x0000 0000 and accessible by Cortex®-M7 and by MDMA through AHBS slave bus of the Cortex®-M7 CPU. The

ITCM-RAM can be used to host code for time-critical routines (such as interrupt handlers) that requires deterministic execution.

The backup RAM is mapped at the address 0x3880 0000 and is accessible by most of the system masters through D3 domain's AHB matrix. With a battery connected to the V_{BAT} pin, the backup SRAM can be used to retain data during low-power mode (Standby and V_{BAT} mode).

Error code correction (ECC)

SRAM data are protected by ECC:

- 7 ECC bits are added per 32-bit word.
- 8 ECC bits are added per 64-bit word for AXI-SRAM and ITCM-RAM.

The ECC mechanism is based on the SECDED algorithm. It supports single-error correction and double-error detection.

2.5 Flash memory overview

The Flash memory interface manages CPU AXI accesses to the Flash memory. It implements the erase and program Flash memory operations and the read and write protection mechanisms.

The Flash memory is organized as follows:

- Two main memory block divided into sectors.
- An information block:
  - System memory from which the device boots in System memory boot mode
  - Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode.

Refer to *Section 3: Embedded Flash memory (FLASH)* for more details.

2.6 Boot configuration

In the STM32H743/753 and STM32H750, two different boot areas can be selected through the BOOT pin and the boot base address programmed in the BOOT_ADD0 and BOOT_ADD1 option bytes as shown in the *Table 8*.

Table 8. Boot modes

Boot mode selection		Boot area
BOOT	**Boot address option bytes**	
0	BOOT_ADD0[15:0]	Boot address defined by user option byte BOOT_ADD0[15:0] ST programmed value: Flash memory at 0x0800 0000
1	BOOT_ADD1[15:0]	Boot address defined by user option byte BOOT_ADD1[15:0] ST programmed value: System bootloader at 0x1FF0 0000

The values on the BOOT pin are latched on the 4th rising edge of SYSCLK after reset release. It is up to the user to set the BOOT pin after reset.

The BOOT pin is also re-sampled when the device exits the Standby mode. Consequently, they must be kept in the required Boot mode configuration when the device is in the Standby mode.

After startup delay, the selection of the boot area is done before releasing the processor reset.

The BOOT_ADD0 and BOOT_ADD1 address option bytes allows to program any boot memory address from 0x0000 0000 to 0x3FFF 0000 which includes:

- All Flash address space
- All RAM address space: ITCM, DTCM RAMs and SRAMs
- The TCM-RAM

The BOOT_ADD0 / BOOT_ADD1 option bytes can be modified after reset in order to boot from any other boot address after next reset.

If the programmed boot memory address is out of the memory mapped area or a reserved area, the default boot fetch address is programmed as follows:

- Boot address 0: FLASH at 0x0800 0000
- Boot address 1: ITCM-RAM at 0x0000 0000

When the Flash level 2 protection is enabled, only boot from Flash memory is available. If the boot address already programmed in the BOOT_ADD0 / BOOT_ADD1 option bytes is out of the memory range or belongs to the RAM address range, the default fetch will be forced from Flash memory at address 0x0800 0000 .

Embedded bootloader

The embedded bootloader code is located in system memory. It is programmed by ST during production. It is used to reprogram the Flash memory using one of the following serial interfaces:

- USART1 on PA9/PA10 pins, USART2 on PA3/2pins, and USART3 on PB10/PB11 pins.
- I2C1 on PB6/9 pins, I2C2 on PF0/PF1 pins, I2C3 on PA8/PC9 pins, and I2C4 on PD12/13 pins
- USB OTG FS in Device mode (DFU) on PA11/PA12 pins
- SPI1 on PA7/6/5/4 pins, SPI2 on PI3/2/1/0 pins, SPI3 on PC12/11/10/PA15pins, and SPI4 on PE14/13/12/11 pins.

For additional information, refer to the application note AN2606.

3 Embedded Flash memory (FLASH)

3.1 Introduction

The embedded Flash memory (FLASH) manages the accesses of any master to the 2 Mbytes of embedded non-volatile memory. It implements the read, program and erase operations, error corrections as well as various integrity and confidentiality protection mechanisms.

The embedded Flash memory manages the automatic loading of non-volatile user option bytes at power-on reset, and implements the dynamic update of these options.

3.2 FLASH main features

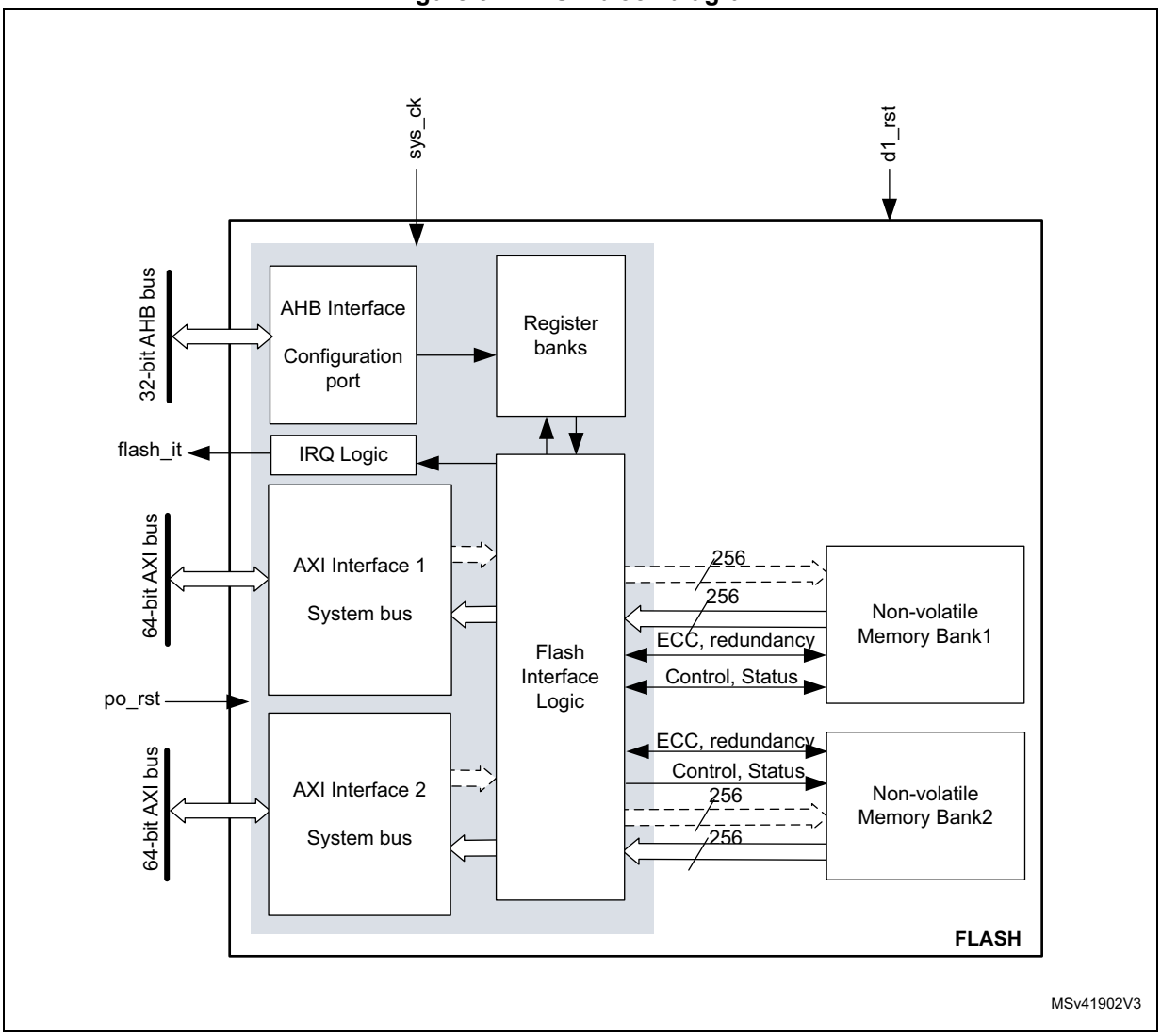
- 2 Mbytes of non-volatile memory divided into two banks of 1 Mbyte each
- Flash memory read operations supporting multiple length (64 bits, 32bits, 16bits or one byte)
- Flash memory programming by 256 bits
- 128-Kbyte sector erase, bank erase and dual-bank mass erase
- Dual-bank organization supporting:
  - simultaneous operations: two read/program/erase operations executed in parallel on both banks (only available on STM32H743/753)
  - Bank swapping: the address mapping of the user Flash memory of each bank can be swapped, along with the corresponding registers.
- Error Code Correction (ECC): one error detection/correction or two error detections per 256-bit Flash word using 10 ECC bits
- Cyclic redundancy check (CRC) hardware module
- User configurable non-volatile option bytes
- Flash memory enhanced protections, activated by option bytes
  - Read protection (RDP), preventing unauthorized Flash memory dump to safeguard sensitive application code
  - Write-protection of sectors (WRPS), available per bank (128 Kbyte sectors)
  - Two proprietary code readout protection (PCROP) areas (one per user Flash bank). When enabled, this area is execute-only.
  - Two secure-only areas (one per user Flash bank). When enabled this area is accessible only if the STM32microcontroller operates in Secure access mode.
- Read and write command queues to streamline Flash operations

3.3 FLASH functional description

3.3.1 FLASH block diagram

Figure 3 shows the embedded Flash memory block diagram.

Figure 3. FLASH block diagram



3.3.2 FLASH internal signals

Table 9 describes a list of the useful to know internal signals available at embedded Flash memory level. These signals are not available on the microcontroller pads.

Table 9. FLASH internal input/output signals

Internal signal name	Signal type	Description
sys_ck	Input	D1 domain bus clock (embedded Flash memory AXI interface clock)
po_rst	Input	Power on reset
d1_rst	Input	D1 domain system reset
flash_it	Output	Embedded Flash memory interface interrupt request

3.3.3 FLASH architecture and integration in the system

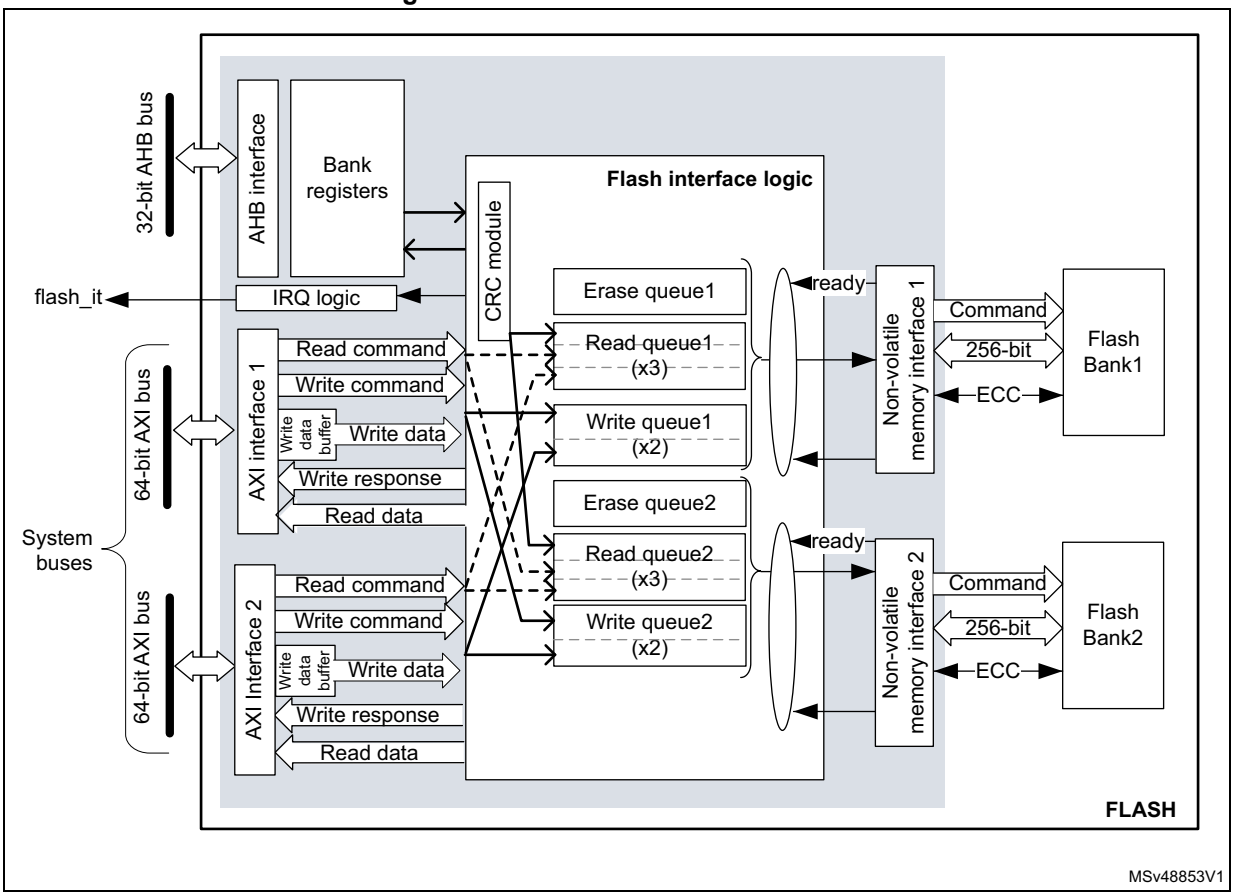
The embedded Flash memory is a central resource for the whole microcontroller. It serves as an interface to two non-volatile memory banks, and organizes the memory in a very specific way. The embedded Flash memory also proposes a set of security features to protect the assets stored in the non-volatile memory at boot time, at run-time and during firmware and configuration upgrades.

The embedded Flash memory offers two 64-bit AXI slave ports for code and data accesses, plus a 32-bit AHB configuration slave port used for register bank accesses.

Note: The application can simultaneously request a read and a write operation through each AXI interface.

The embedded Flash memory microarchitecture is shown in *Figure 4*.

Figure 4. Detailed FLASH architecture



Behind the system interfaces, the embedded Flash memory implements various command queues and buffers to perform Flash read, write and erase operations with maximum efficiency.

Thanks to the addition of a read and write data buffer, the AXI slave port handles the following access types:

- Multiple length: 64 bits, 32 bits, 16 bits and 8 bits
- Single or burst accesses
- Write wrap burst must not cross 32-byte aligned address boundaries to target exactly one Flash word

The AHB configuration slave port supports 8-bit, 16-bit and 32-bit word accesses.

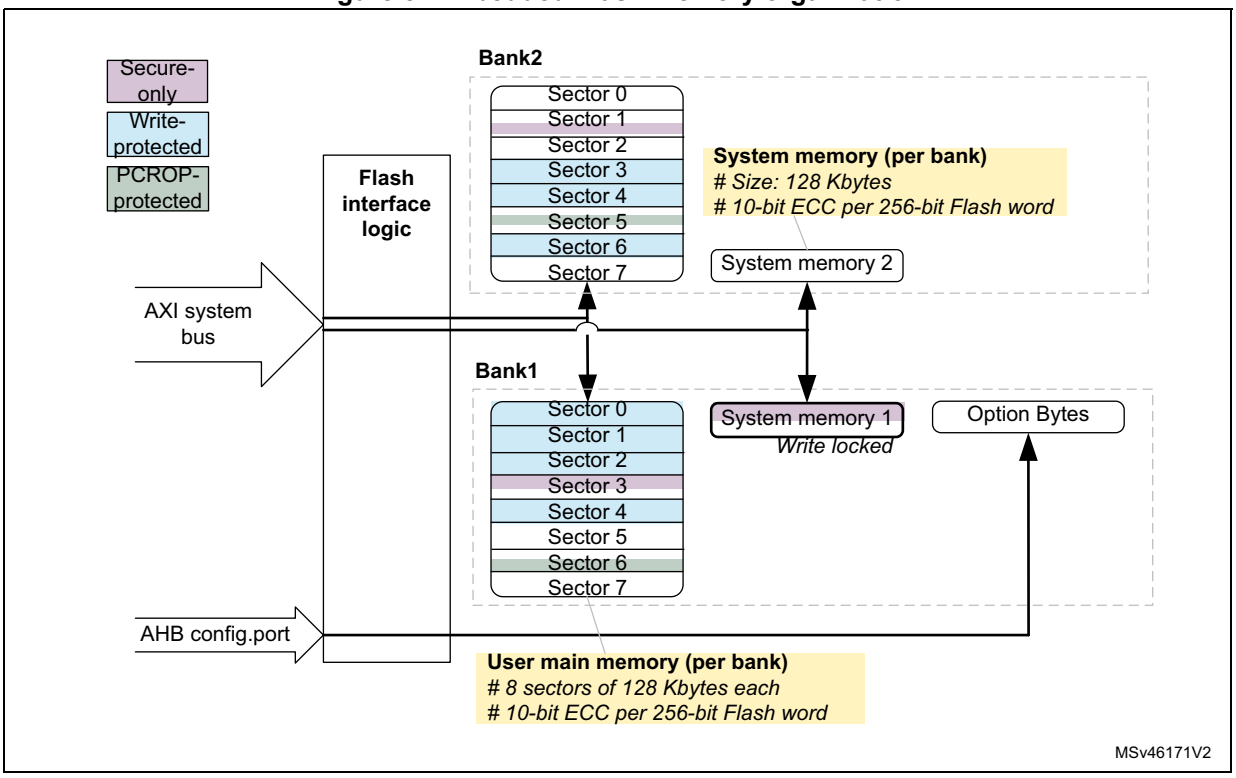
The embedded Flash memory is built in such a way that only one read or write operation can be executed at a time on a given bank.

3.3.4 Flash memory architecture and usage

Flash memory architecture

Figure 5 shows the non-volatile memory organization supported by the embedded Flash memory.

Figure 5. Embedded Flash memory organization



1. On STM32H750xB, bank 2 is not available and bank 1 contains only one 128 Kbyte sector.

The embedded Flash non-volatile memory is composed of:

- For STM32H743/753 devices: a 2-Mbyte main memory block, organized as two banks of 1 Mbyte each. Each bank is in turn divided in eight 128-Kbyte sectors and features Flash-word rows of 256 bits + 10 bits of ECC per word.
- For STM32H750xB devices: a 128-Kbyte user Flash memory block containing one user sector of 128 Kbytes (4 K Flash words).
- A system memory block of 256 Kbytes, divided into two 128 Kbyte banks. The system memory is ECC protected.
- A set of non-volatile option bytes loaded at reset by the embedded Flash memory and accessible by the application software only through the AHB configuration register interface.

The overall Flash memory architecture is summarized in *Table 10* and *Table 11*.

Table 10. Flash memory organization on STM32H750xB devices

Flash memory area		Address range	Size (bytes)	Region name	Access interface	SNB1/2 (1)
User main memory	Bank 1	0x0800 0000-0x0801 FFFF	128 K	Sector 0	AXI ports	0x0
System memory	Bank 1	0x1FF0 0000-0x1FF1 FFFF	128 K	System Flash (read-only)	AXI ports	N/A(2)
	Bank 2	0x1FF4 0000-0x1FF5 FFFF	128 K	System Flash		N/A(2)
Option bytes		N/A	-	User option bytes	Registers only(3)	N/A

1. SNB1/2 contains the target sector number for an erase operation. See *Section 3.3.10* for details.
2. Cannot be erased by application software.
3. On STM32H750xB devices, only the Bank 1 option byte registers are applicable. Bank2 option byte registers must be kept at reset value.

Table 11. Flash memory organization on STM32H743/753 devices

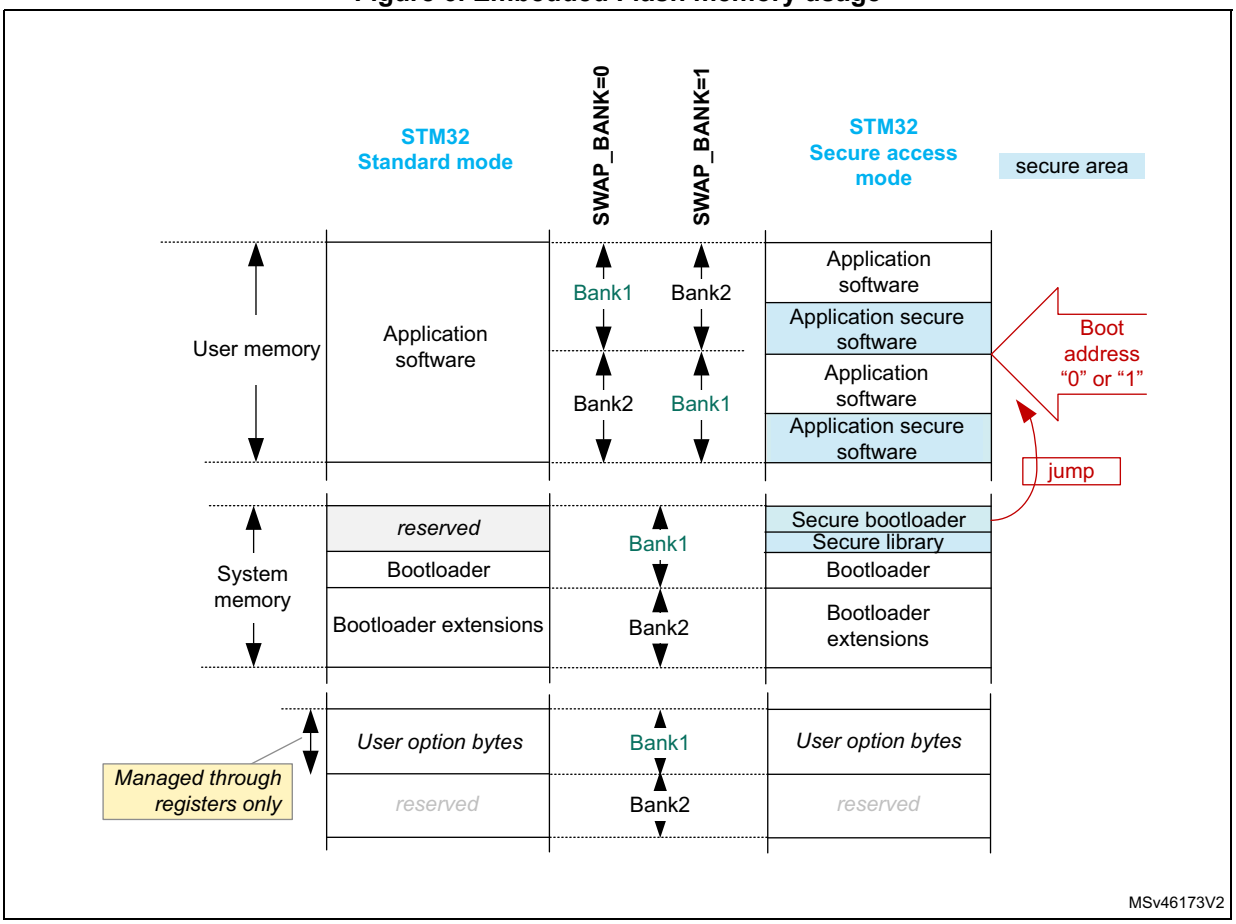
Flash memory area		Address range	Size (bytes)	Region name	Access interface	SNB1/2 (1)
User main memory	Bank 1	0x0800 0000-0x0801 FFFF	128 K	Sector 0	AXI ports	0x0
		0x0802 0000-0x0803 FFFF	128 K	Sector 1		0x1
		...	...	...		...
		0x080E 0000-0x080F FFFF	128 K	Sector 7		0x7
	Bank 2	0x0810 0000-0x0811 FFFF	128 K	Sector 0		0x0
		0x0812 0000-0x0813 FFFF	128 K	Sector 1		0x1
		...	...	...		...
		0x081E 0000-0x081F FFFF	128 K	Sector 7		0x7
System memory	Bank 1	0x1FF0 0000-0x1FF1 FFFF	128 K	System Flash (read-only)	AXI ports	N/A(2)
	Bank 2	0x1FF4 0000-0x1FF5 FFFF	128 K	System Flash		N/A(2)
Option bytes		N/A	-	User option bytes	Registers only	N/A

1. SNB1/2 contains the target sector number for an erase operation. See *Section 3.3.10* for details.
2. Cannot be erased by application software.

Partition usage

Figure 6 shows how the embedded Flash memory is used both by STMicroelectronics and the application software.

Figure 6. Embedded Flash memory usage



User and system memories are used differently according to whether the microcontroller is configured by the application software in Standard mode or in Secure access mode. This selection is done through the SECURITY option bit (see *Section 3.4.6*):

- In Standard mode, the user memory contains the application code and data, while the system memory is loaded with the STM32 bootloader. When a reset occurs, the core jumps to the boot address configured through the BOOT pin and the BOOT_ADD0/1 option bytes.
- In Secure access mode, dedicated libraries can be used for secure boot. They are located in user Flash and system Flash memory:
  - ST libraries in system Flash memory assist the application software boot with special features such as secure boot and secure firmware install (SFI).
  - Application secure libraries in user Flash memory are used for secure firmware update (SFU).

In Secure access mode, the microcontroller always boots into the secure bootloader code (unique entry point). Then, if no secure services are required, this code securely jumps to the requested boot address configured through the BOOT pin and the option bytes, as shown in *Figure 6* (see *Section 4: Secure memory management* for details).

Note: For more information on option byte setup for boot, refer to Section 3.4.7.

Additional partition usage is the following:

- The option bytes are used by STMicroelectronics and by the application software as non-volatile product options (e.g. boot address, protection configuration and reset behaviors).

Note: For further information on STM32 bootloader flashing by STMicroelectronics, refer to application note AN2606 "STM32 microcontroller system memory boot mode" available from www.st.com.

Bank swapping

As shown in *Figure 6*, the embedded Flash memory offers a bank swapping feature that can be configured through the SWAP_BANK bit, always available for the application. For more information please refer to *Section 3.3.13*.

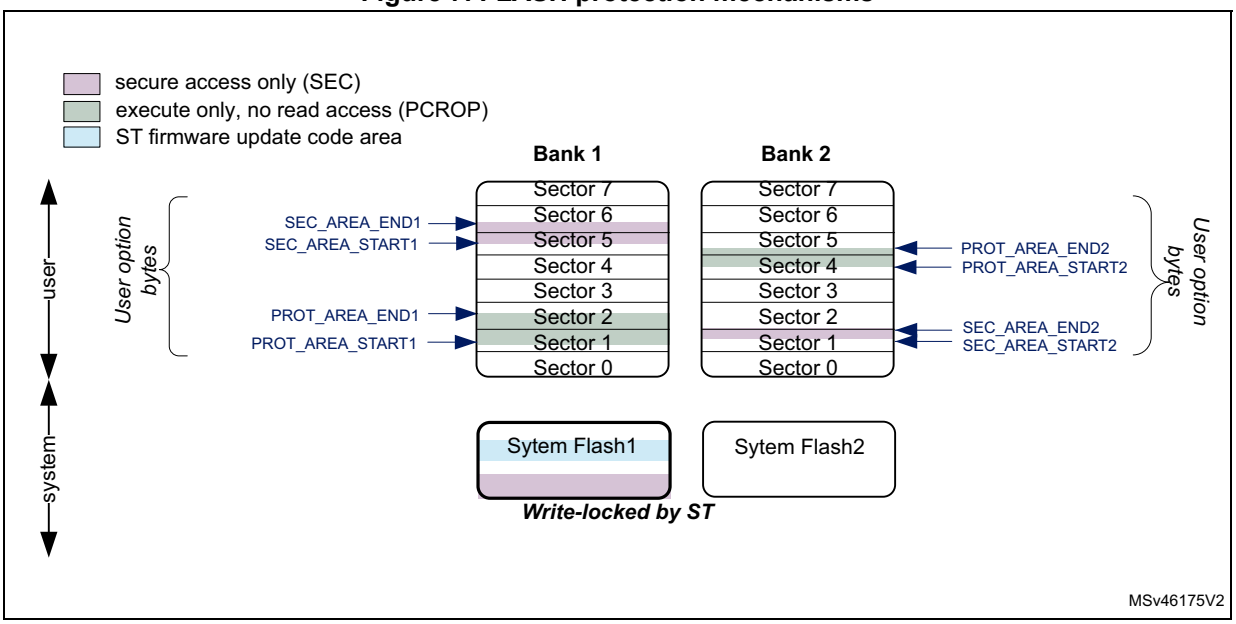
3.3.5 FLASH system performance enhancements

The embedded Flash memory uses read and write command queues (one per bank) in order to enhance Flash operations.

3.3.6 FLASH data protection schemes

Figure 7 gives an overview of the protection mechanisms supported by the embedded Flash memory. A PCROP and a secure-only are a can be defined for each bank. The properties of these protected areas are detailed in *Section 3.5*.

Figure 7. FLASH protection mechanisms



3.3.7 Overview of FLASH operations

Read operations

The embedded Flash memory can perform read operations on the whole non-volatile memory using various granularities: 64 bits, 32 bits, 16 bits or one byte. User and system Flash memories are read through the AXI interface, while the option bytes are read through the register interface.

The embedded Flash memory supports read-while-write operations provided the read and write operations target different banks. Similarly read-while-read operations are supported when two read operations target different banks.

To increase efficiency, the embedded Flash memory implements the buffering of consecutive read requests in the same bank.

For more details on read operations, refer to *Section 3.3.8: FLASH read operations*.

Program/erase operations

The embedded Flash memory supports the following program and erase operations:

- Single Flash word write (256-bit granularity), with the possibility for the application to force-write a user Flash word with less than 256 bits
- Single sector erase
- Bank erase (single or dual)
- Option byte update

Thanks to its dual bank architecture, the embedded Flash memory can perform any of the above write or erase operation on one bank while a read or another program/erase operation is executed on the other bank.

Note: Program and erase operations are subject to the various protection that could be set on the embedded Flash memory, such as write protection and global readout protection (see next sections for details).

To increase efficiency, the embedded Flash memory implements the buffering of consecutive write accesses in the same bank.

For more details refer to *Section 3.3.9: FLASH program operations* and *Section 3.3.10: FLASH erase operations*.

Protection mechanisms

The embedded Flash memory supports different protection mechanisms:

- Global readout protection (RDP)
- Proprietary code readout protection (PCROP)
- Write protection
- Secure access only protection

For more details refer to *Section 3.5: FLASH protection mechanisms*.

Option byte loading

Under specific conditions, the embedded Flash memory reliably loads the non-volatile option bytes stored in non-volatile memory, thus enforcing boot and security options to the whole system when the embedded Flash memory becomes functional again. For more details refer to *Section 3.4: FLASH option bytes*.

Bank/ register swapping

The embedded Flash memory allows swapping bank 1 and bank 2 memory mapping. This feature can be used after a firmware upgrade to restart the microcontroller on the new firmware after a system reset. For more details on the feature, refer to *Section 3.3.13: Flash bank and register swapping (STM32H743/753 devices only)*.

3.3.8 FLASH read operations

Read operation overview

The embedded Flash memory supports, for each memory bank, the execution of one read command while two are waiting in the read command queue. Multiple read access types are also supported as defined in *Section 3.3.3: FLASH architecture and integration in the system*.

The read commands to each bank are associated with a 256-bit read data buffer.

Note: The embedded Flash memory can perform single error correction and double error detection while read operations are being executed (see Section 3.3.12: Flash memory error protections).

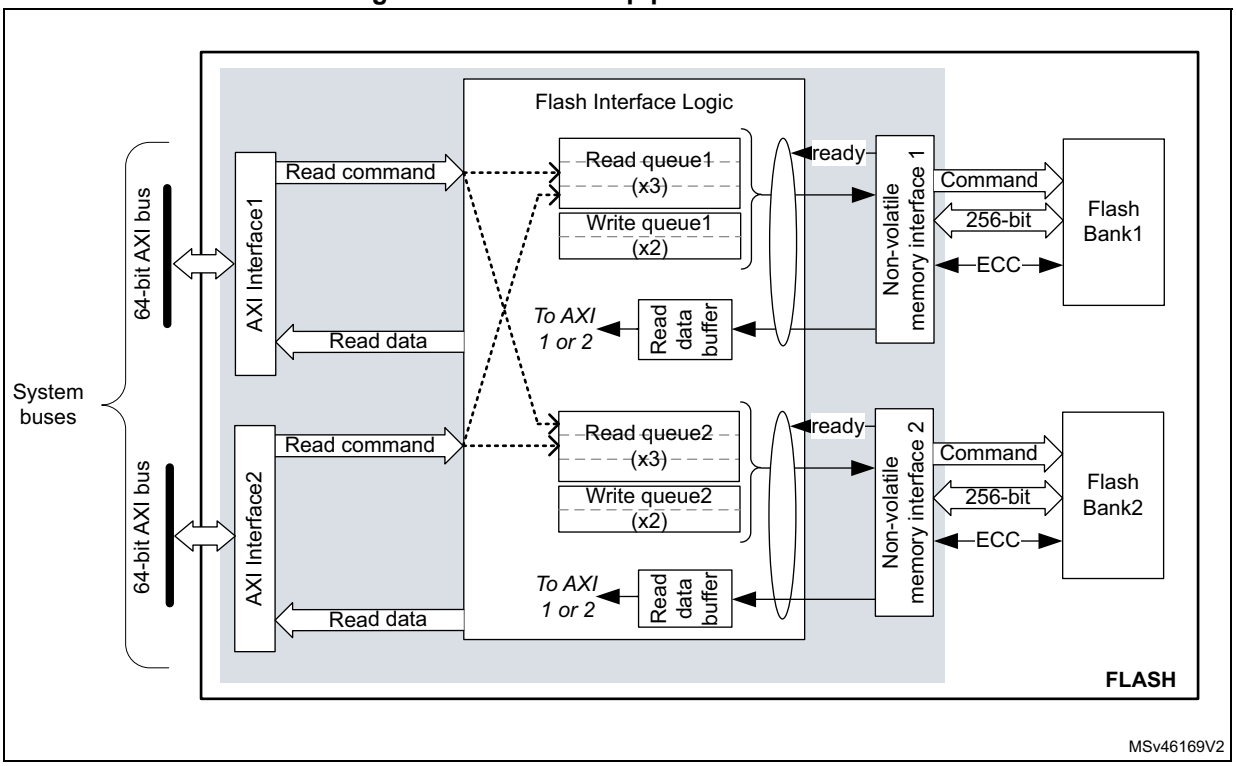
The AXI interface read channel operates as follows:

- When the read command queue is full, any new AXI read request stalls the bus read channel interface and consequently the master that issued that request.
- If several consecutive read accesses request data that belong to the same Flash data word (256 bits), the data are read directly from the current data read buffer, without triggering additional Flash read operations. This mechanism occurs each time a read access is granted. When a read access is rejected for security reasons (e.g. PCROP protected word), the corresponding read error response is issued by the embedded Flash memory and no read operation to Flash memory is triggered.

The Read pipeline architecture is summarized in *Figure 8*.

For more information on bus interfaces, refer to *Section 3.3.3: FLASH architecture and integration in the system*.

Figure 8. FLASH read pipeline architecture



Single read sequence

The recommended simple read sequence is the following:

1. Freely perform read accesses to any AXI-mapped area.
2. The embedded Flash memory effectively executes the read operation from the read command queue buffer as soon as the non-volatile memory is ready and the previously requested operations on this specific bank have been served.

Adjusting read timing constraints

The embedded Flash memory clock must be enabled and running before reading data from non-volatile memory.

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the Flash access control register (FLASH_ACR) according to the embedded Flash memory AXI interface clock frequency (sys_ck) and the internal voltage range of the device (V_{core}).

Table 12 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded Flash memory clock frequency and its supply voltage ranges.

Table 12. FLASH recommended number of wait states and programming delay(1)

Number of wait states (LATENCY)	Programming delay (WRHIGHFREQ)	AXI Interface clock frequency vs V_{CORE} range		
		VOS3 range 0.95 V - 1.05 V	VOS2 range 1.05 V - 1.15 V	VOS1 range 1.15 V - 1.26 V
0 WS (1 FLASH clock cycle)	00	[0 ; 45 MHz]	[0 ; 55 MHz]	[0 ; 70 MHz]
1 WS (2 FLASH clock cycles)	01	]45 MHz ; 90 MHz]	]55 MHz ; 110 MHz]	]70 MHz ; 140 MHz]
2 WS (3 FLASH clock cycles)	01	]90 MHz ; 135 MHz]	]110 MHz ; 165 MHz]	]140 MHz ; 185 MHz]
	10	-	-	]185 MHz ; 210 MHz]
3 WS (4 FLASH clock cycles)	10	]135 MHz ; 180 MHz]	]165 MHz ; 225 MHz]	]210 MHz ; 225 MHz]
4 WS (5 FLASH clock cycles)	10	]180 MHz ; 225 MHz]	225 MHz	-

1. Refer to the Reset and clock control section RCC section for the maximum product F_{ACLK} frequency.

Adjusting system frequency

After power-on, a default 7 wait-state latency is specified in FLASH_ACR register, in order to accommodate AXI interface clock frequencies with a safety margin (see *Table 12*).

When changing the AXI bus frequency, the application software must follow the below sequence in order to tune the number of wait states required to access the non-volatile memory.

To increase the embedded Flash memory clock source frequency:

1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in *Table 12*.
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the embedded Flash memory clock source and/or the AXI bus clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded Flash memory clock source and/or the new AXI bus clock prescaler value are taken in account by reading back the embedded Flash memory clock source status and/or the AXI bus prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the embedded Flash memory clock source frequency:

1. Modify the embedded Flash memory clock source and/or the AXI bus clock prescaler in the RCC_CFGR register of reset and clock controller (RCC).
2. Check that the embedded Flash memory new clock source and/or the new AXI bus clock prescaler value are taken into account by reading back the embedded Flash memory clock source status and/or the AXI interface prescaler value in the RCC_CFGR register of reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in FLASH_ACR register, as described in *Table 12*.
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

Error code correction (ECC)

The embedded Flash memory embeds an error correction mechanism. Single error correction and double error detection are performed for each read operation. For more details, refer to *Section 3.3.12: Flash memory error protections*.

Read errors

When the ECC mechanism is not able to correct the read operation, the embedded Flash memory reports read errors as described in *Section 3.7.7: Error correction code error (SNECCERR/DBECCERR)*.

Read interrupts

See *Section 3.8: FLASH interrupts* for details.

3.3.9 FLASH program operations

Program operation overview

The virgin state of each non-volatile memory bitcell is 1. The embedded Flash memory supports programming operations that can change (reset) any memory bitcell to 0. However these operations do not support the return of a bit to its virgin state. In this case an erase operation of the entire sector is required.

A program operation consists in issuing write commands. The embedded Flash memory supports, for each memory bank, the execution of one write command while one command is waiting in the write command queue. Since a 10-bit ECC code is associated to each 256-bit data Flash word, only write operations by 256 bits are executed in the non-volatile memory.

Note: The application can decide to write as little as 8 bits to a 256 Flash word. In this case, a force-write mechanism to the 256 bits + ECC is used (see FW1/2 bit of FLASH_CR1/2 register).

System Flash memory bank 1 cannot be written by the application software. System Flash memory bank 2 can be written by STMicroelectronics secure firmware only.

It is not recommended to overwrite a Flash word that is not virgin. The result may lead to an inconsistent ECC code that will be systematically reported by the embedded Flash memory, as described in *Section 3.7.7: Error correction code error (SNECCERR/DBECCERR)*.

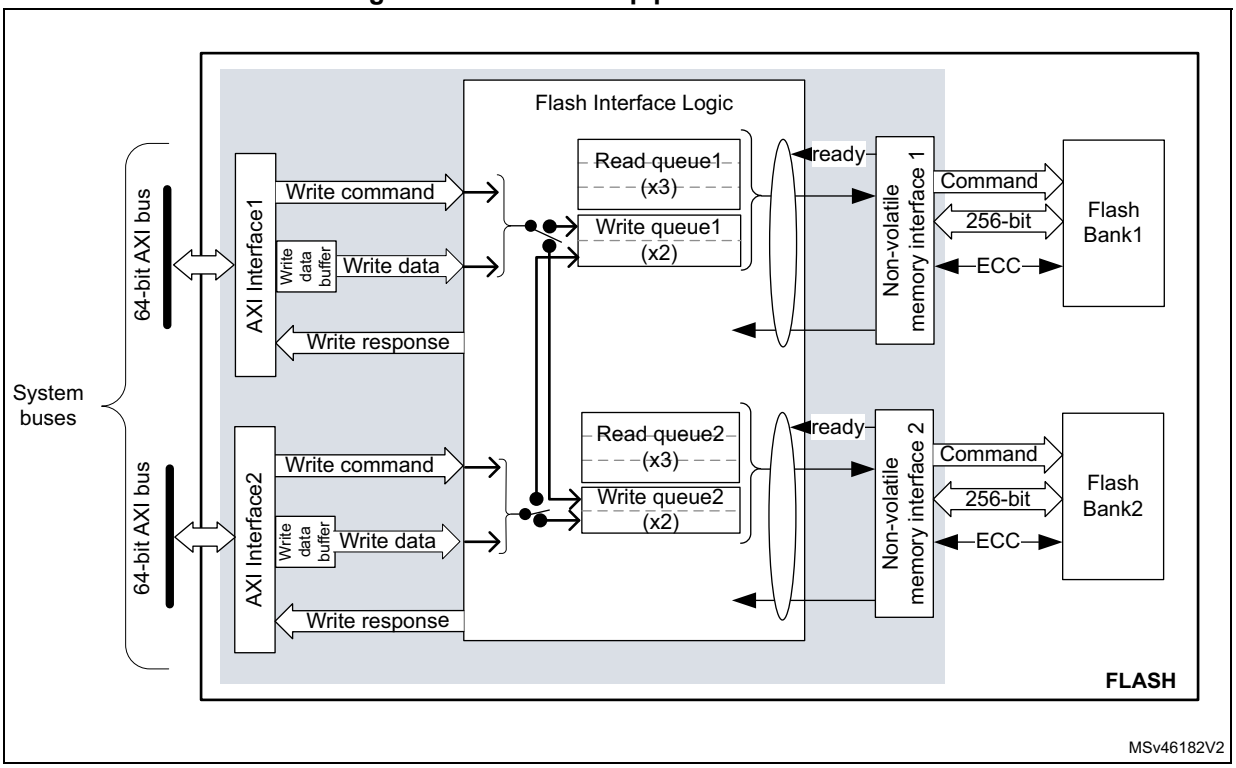
The AXI interface write channel operates as follows:

- A 256-bit write data buffer is associated with each AXI interface. It supports multiple write access types (64 bits, 32 bits, 16 bits and 8 bits).
- When the write queue is full, any new AXI write request stalls the bus write channel interface and consequently the master that issued that request.

The write pipeline architecture is described in *Figure 9*.

For more information on bus interfaces, refer to *Section 3.3.3: FLASH architecture and integration in the system*.

Figure 9. FLASH write pipeline architecture



Managing write protections

Before programming a user sector, the application software must check the protection of the targeted Flash memory area.

The embedded Flash memory checks the protection properties of the write transaction target at the output of the write queue buffer, just before the effective write operation to the non-volatile memory:

- If a write protection violation is detected, the write operation is canceled and write protection error (WRPERR1/2) is raised in FLASH_SR1/2 register.
- If the write operation is valid, the 10-bit ECC code is concatenated to the 256 bits of data and the write to non-volatile memory is effectively executed.

Note: No write protection check is performed when the embedded Flash memory accepts AXI write requests.

The write protection flag does not need to be cleared before performing a new programming operation.

Monitoring ongoing write operations

The application software can use three status flags located in FLASH_SR1/2 in order to monitor ongoing write operations. Those status are available for each bank.

- **BSY1/2**: this bit indicates that an effective write, erase or option byte change operation is ongoing to the non-volatile memory.
- **QW1/2**: this bit indicates that a write, erase or option byte change operation is pending in the write queue or command queue buffer. It remains high until the write operation is complete. It supersedes the BSY1/2 status bit.
- **WBNE1/2**: this bit indicates that the embedded Flash memory is waiting for new data to complete the 256-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, force-writes the operation using FW1/2 bit in FLASH_CR1/2, or disables all write operations in the corresponding bank.

Enabling write operations

Before programming the user Flash memory in bank 1 (respectively bank 2), the application software must make sure that PG1 bit (respectively PG2) is set to 1 in FLASH_CR1 (respectively FLASH_CR2). If it is not the case, an unlock sequence must be used (see *Section 3.5.1: FLASH configuration protection*) and the PG1/2 bit must be set.

When the option bytes need to be modified or a mass erase needs to be started, the application software must make sure that FLASH_OPTCR is unlocked. If it is not the case, an unlock sequence must be used (see *Section 3.5.1: FLASH configuration protection*).

Note: The application software must not unlock a register that is already unlocked, otherwise this register will remain locked until next system reset.

If needed, the application software can update the programming delay and programming parallelism as described at the end of this section.

Single write sequence

The recommended single write sequence in bank 1/2 is the following:

1. Unlock the FLASH_CR1/2 register, as described in *Section 3.5.1: FLASH configuration protection* (only if register is not already unlocked).
2. Enable write operations by setting the PG1/2 bit in the FLASH_CR1/2 register.
3. Check the protection of the targeted memory area.
4. Write one Flash-word corresponding to 32-byte data starting at a 32-byte aligned address.
5. Check that QW1 (respectively QW2) has been raised and wait until it is reset to 0.

If step 4 is executed incrementally (e.g. byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using FW1/2 bit in FLASH_CR1/2 register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW1/2 bit has no effect.

Note: Using a force-write operation prevents the application from updating later the missing bits with a value different from 1, which is likely to lead to a permanent ECC error.

Any write access requested while the PG1/2 bit is cleared to 0 is rejected. In this case, no error is generated on the bus, but the PGSERR1/2 flag is raised.

Clearing the programming sequence error (PGSERR) and inconsistency error (INCERR) is mandatory before attempting a write operation (see Section 3.7: FLASH error management for details).

Adjusting programming timing constraints

Program operation timing constraints depend of the embedded Flash memory clock frequency, which directly impacts the performance. If timing constraints are too tight, the non-volatile memory will not operate correctly, if they are too lax, the programming speed will not be optimal.

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to *Table 12* in *Section 3.3.8: FLASH read operations* for the recommended programming delay depending on the embedded Flash memory clock frequency.

Note: FLASH_ACR configuration register is common to both banks.

The application software must check that no program/erase operation is ongoing before modifying WRHIGHFREQ parameter.

Adjusting programming parallelism

The parallelism is the maximum number of bits that can be written to 0 in one shot during a write operation. The programming parallelism is also used during sector and bank erase.

There is no hardware limitation on programming parallelism. The user can select different parallelisms depending on the application requirements: the lower the parallelism, the lower the peak consumption during a programming operation, but the longer the execution time.

The parallelism is configured through the PSIZE1/2 bits in FLASH_CR1/2 register. Two distinct values can be defined for bank 1 and 2 (refer to *Table 13).*

Table 13. FLASH parallelism parameter

PSIZE1/2	Parallelism
00	8 bits (one byte)
01	16 bits
10	32 bits
11	64 bits

Programming errors

When a program operation fails, an error can be reported as described in *Section 3.7: FLASH error management*.

Programming interrupts

See *Section 3.8: FLASH interrupts* for details.

3.3.10 FLASH erase operations

Erase operation overview

The embedded Flash memory can perform erase operations on 128-Kbyte user sectors, on one user Flash memory bank or on two user Flash memory banks (i.e. mass erase). The system Flash memory (bank 2 only) can also be erased, but by STMicroelectronics secure firmware only.

Note: System Flash memory bank 1 cannot be erased by the application software.

The erase operation forces all non-volatile bit cells to high state, which corresponds to the virgin state. It clears existing data and corresponding ECC, allowing a new write operation to be performed. If the application software reads back a word that has been erased, all the bits will be read at 1, without ECC error.

Erase operations are similar to read or program operations except that the commands are queued in a special buffer (a two-command deep erase queue).

Erase commands are issued through the AHB configuration interface. If the embedded Flash memory receives simultaneously a write and an erase request for the same bank, both operations are accepted but the write operation is executed first.

Erase and security

A user sector can be erased only if it does not contain PCROP, secure-only or write-protected data (see *Section 3.5: FLASH protection mechanisms* for details). In other words, if the application software attempts to erase a user sector with at least one Flash word that is protected, the sector erase operation is aborted and the WRPERR1/2 flag is raised in the FLASH_SR1/2 register, as described in *Section 3.7.2: Write protection error (WRPERR)*.

The embedded Flash memory allows the application software to perform an erase followed by an automatic protection removal (PCROP, secure-only area and write protection), as described hereafter.

Enabling erase operations

Before erasing a sector in bank 1 (respectively bank 2), the application software must make sure that FLASH_CR1 (respectively FLASH_CR2) is unlocked. If it is not the case, an unlock sequence must be used (see *Section 3.5.1: FLASH configuration protection*).

Note: The application software must not unlock a register that is already unlocked, otherwise this register will remain locked until next system reset.

Similar constraints apply to bank erase requests.

Flash sector erase sequence

To erase a 128-Kbyte user sector, proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to *Section 3.7: FLASH error management* for details.
2. Unlock the FLASH_CR1/2 register, as described in *Section 3.5.1: FLASH configuration protection* (only if register is not already unlocked).
3. Set the SER1/2 bit and SNB1/2 bitfield in the corresponding FLASH_CR1/2 register. SER1/2 indicates a sector erase operation, while SNB1/2 contains the target sector number.
4. Set the START1/2 bit in the FLASH_CR1/2 register.
5. Wait for the QW1/2 bit to be cleared in the corresponding FLASH_SR1/2 register.

Note: If a bank erase is requested simultaneously to the sector erase (BER1/2 bit set), the bank erase operation supersedes the sector erase operation.

Standard Flash bank erase sequence

To erase all bank sectors except for those containing secure-only and protected data, proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to *Section 3.7: FLASH error management* for details.
2. Unlock the FLASH_CR1/2 register, as described in *Section 3.5.1: FLASH configuration protection* (only if register is not already unlocked).
3. Set the BER1/2 bit in the FLASH_CR1/2 register corresponding to the targeted bank.
4. Set the START1/2 bit in the FLASH_CR1/2 register to start the bank erase operation. Then wait until the QW1/2 bit is cleared in the corresponding FLASH_SR1/2 register.

Note: BER1/2 and START1/2 bits can be set together, so above steps 3 and 4 can be merged.

If a sector erase is requested simultaneously to the bank erase (SER1/2 bit set), the bank erase operation supersedes the sector erase operation.

Flash bank erase with automatic protection-removal sequence

To erase all bank sectors including those containing secure-only and protected data without performing an RDP regression (see *Section 3.5.3*), proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to *Section 3.7: FLASH error management* for details.
2. Unlock FLASH_OPTCR register, as described in *Section 3.5.1: FLASH configuration protection* (only if register is not already unlocked).
3. If a PCROP-protected area exists set DMEP1/2 bit in FLASH_PRAR_PRG1/2 register. In addition, program the PCROP area end and start addresses so that the difference is negative, i.e. PROT_AREA_END1/2 < PROT_AREA_START1/2.
4. If a secure-only area exists set DMES1/2 bit in FLASH_SCAR_PRG1/2 register. In addition, program the secure-only area end and start addresses so that the difference is negative, i.e. SEC_AREA_END1/2 < SEC_AREA_START1/2.
5. Set all WRPSn1/2 bits in FLASH_WPSN_PRG1/2R to 1 to disable all sector write protection.
6. Unlock FLASH_CR1/2 register, only if register is not already unlocked.
7. Set the BER1/2 bit in the FLASH_CR1/2 register corresponding to the target bank.
8. Set the START1/2 bit in the FLASH_CR1/2 register to start the bank erase with protection removal operation. Then wait until the QW1/2 bit is cleared in the corresponding FLASH_SR1/2 register. At that point a bank erase operation has erased the whole bank including the sectors containing PCROP-protected and/or secure-only data, and an option byte change has been automatically performed so that all the protections are disabled.

Note: BER1/2 and START1/2 bits can be set together, so above steps 8 and 9 can be merged.

Please note the following warnings regarding to above sequence:

- It is not possible to perform the above sequence on one bank while modifying the protection parameters of the other bank.
- No other option bytes than the one indicated above must be changed, and no protection change must be performed in the bank that is not targeted by the bank erase with protection removal request.
- When one or both of the events above occurs, a simple bank erase occurs, no option byte change is performed and no option change error is set.

Flash mass erase sequence

To erase all sectors of both banks simultaneously (STM32H743/753 only), excepted for those containing secure-only and protected data, the application software can set the MER bit to 1 in FLASH_OPTCR register, as described below:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to *Section 3.7: FLASH error management* for details.
2. Unlock the two FLASH_CR1/2 registers and FLASH_OPTCR register, as described in *Section 3.5.1: FLASH configuration protection* (only if the registers are not already unlocked).
3. Set the MER bit to 1 in FLASH_OPTCR register. It automatically sets BER1, BER2, START1 and START2 to 1, thus launching a bank erase operation on both banks. Then wait until both QW1 and QW2 bits are cleared in the corresponding FLASH_SR1/2 register.

Flash mass erase with automatic protection-removal sequence

To erase all sectors of both banks simultaneously (STM32H743/753 only), including those containing secure-only and protected data, and without performing an RDP regression, proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation.
2. Unlock the two FLASH_CR1/2 registers and FLASH_OPTCR register (only if the registers are not already unlocked).
3. If a PCROP-protected area exists, set DMEP1/2 bit in FLASH_PRAR_PRG1/2 register. In addition, program the PCROP area end and start addresses so that the difference is negative. This operation must be performed for both banks.
4. If a secure-only area exists, set DMES1/2 bit in FLASH_SCAR_PRG1/2 register. In addition, program the secure-only area end and start addresses so that the difference is negative. This operation must be performed for both banks.
5. Set all WRPSn1/2 bits in FLASH_WPSN_PRG1/2R to 1 to disable all sector write protections. This operation must be performed for both banks.
6. Set the MER bit to 1 in FLASH_OPTCR register, then wait until the QW1/2 bit is cleared in the corresponding FLASH_SR1/2 register. At that point, a Flash bank erase with automatic protection removal is executed on both banks. The sectors containing PCROP-protected and/or secure-only data become unprotected since an option byte change is automatically performed after the mass erase so that all the protections are disabled.

Caution: No other option bytes than the ones mentioned in the above sequence must be changed, otherwise a simple mass erase is executed, no option byte change is performed and no option change error is raised.

3.3.11 FLASH parallel operations (STM32H743/753 devices only)

As the non-volatile memory is divided into two independent banks , the embedded Flash memory interface can drive different operations at the same time on each bank. For example a read, write or erase operation can be executed on bank 1 while another read, write or erase operation is executed on bank 2.

In all cases, the sequences described in *Section 3.3.8: FLASH read operations*, *Section 3.3.9: FLASH program operations* and *Section 3.3.10: FLASH erase operations* apply.

3.3.12 Flash memory error protections

Error correction codes (ECC)

The embedded Flash memory supports an error correction code (ECC) mechanism. It is based on the SECDED algorithm in order to correct single errors and detects double errors.

This mechanism uses 10 ECC bits per 256-bit Flash word, and applies to user and system Flash memory.

More specifically, during each read operation from a 256-bit Flash word, the embedded Flash memory also retrieves the 10-bit ECC information, computes the ECC of the Flash word, and compares the result with the reference value. If they do not match, the

corresponding ECC error is raised as described in *Section 3.7.7: Error correction code error (SNECCERR/DBECCERR)*.

During each program operation, a 10- bit ECC code is associated to each 256-bit data Flash word, and the resulting 266-bit Flash word information is written in non-volatile memory.

Cyclic redundancy codes (CRC)

The embedded Flash memory implements a cyclic redundancy check (CRC) hardware module. This module checks the integrity of a given user Flash memory area content (see *Figure 4: Detailed FLASH architecture*).

The area processed by the CRC module can be defined either by sectors or by start/end addresses. It can also be defined as the whole bank (user Flash memory area only).

Note: Only one CRC check operation on bank 1 or 2 can be launched at a time. To avoid corruption, do not configure the CRC calculation on the one bank, while calculating the CRC on the other bank.

When enabled, the CRC hardware module performs multiple reads by chunks of 4, 16, 64 or 256 consecutive Flash-word (i.e. chunks of 128, 512, 2048 or 8192 bytes). These consecutive read operations are pushed by the CRC module into the required read command queue together with other AXI read requests, thus avoiding to deny AXI read commands.

CRC computation uses CRC-32 (Ethernet) polynomial 0x4C11DB7:

$$X^{32}+ X^{26}+ X^{23}+ X^{22}+ X^{16}+ X^{12}+ X^{11}+ X^{10}+ X^{8}+ X^{7}+ X^{5}+ X^{4}+ X^{2}+ X + 1$$

The CRC operation is concurrent with option byte change as the same hardware is used for both operations. To avoid the CRC computation from being corrupted, the application shall complete the option byte change (by reading the result of the change) before running a CRC operation, and vice-versa.

The sequence recommended to configure a CRC operation in the bank 1/2 is the following:

1. Unlock FLASH_CR1/2 register, if not already unlocked.
2. Enable the CRC feature by setting the CRC_EN bit in FLASH_CR1/2.
3. Program the desired data size in the CRC_BURST field of FLASH_CRCCR1/2.
4. Define the user Flash memory area on which the CRC has to be computed.Two solutions are possible:
   - Define the area start and end addresses by programing FLASH_CRCSADD1/2R and FLASH_CRCEADD1/2R, respectively,
   - or select the targeted sectors by setting the CRC_BY_SECT bit in FLASH_CRCCR1/2 and by programming consecutively the target sector numbers in the CRC_SECT field of the FLASH_CRCCR1/2 register. Set ADD_SECT bit after each CRC_SECT programming.
5. Start the CRC operation by setting the START_CRC bit.
6. Wait until the CRC_BUSY1/2 flag is reset in FLASH_SR1/2 register.
7. Retrieve the CRC result in the FLASH_CRCDATAR register.

The CRC can be computed for a whole bank by setting the ALL_BANK bit in the FLASH_CRCCR1/2 register.

Note: The application should avoid running a CRC on PCROP- or secure-only user Flash memory area since it may alter the expected CRC value.

CRC computation does not raise standard read error flags such as RDSERR1/2, RDPERR1/2 and DBECCERR1/2.

3.3.13 Flash bank and register swapping (STM32H743/753 devices only)

Flash bank swapping

The embedded Flash memory bank 1 and bank 2 can be swapped in the memory map accessible through AXI interface. This feature can be used after a firmware upgrade to restart the device on the new firmware. Bank swapping is controlled by the SWAP_BANK bit of the FLASH_OPTCR register.

Note: The Flash bank swapping is not available on STM32H750xB devices. The SWAP_BANK option bit must be kept to '0'.

Table 14 shows the memory map that can be accessed from the embedded Flash memory AXI slave interface, depending on the SWAP_BANK bit configuration.

Table 14. FLASH AXI interface memory map vs swapping option

Flash memory area	Flash memory corresponding bank		Start address	End address	Size (bytes)	Region Name
	SWAP_ BANK=0	SWAP_ BANK=1				
User main memory	Bank 1	Bank 2	0x0800 0000	0x0801 FFFF	128 K	Sector 0
			0x0802 0000	0x0803 FFFF	128 K	Sector 1
			...	...	...	...
			0x080E 0000	0x080F FFFF	128 K	Sector 7
	Bank 2	Bank 1	0x0810 0000	0x0811 FFFF	128 K	Sector 0
			0x0812 0000	0x0813 FFFF	128 K	Sector 1
			...	...	...	...
			0x081E 0000	0x081F FFFF	128 K	Sector 7
System memory	Bank 1		0x1FF0 0000	0x1FF1 FFFF	128 K	System Flash memory (bank 1)
	Bank 2		0x1FF4 0000	0x1FF5 FFFF	128 K	System Flash memory (bank 2)

The SWAP_BANK bit in FLASH_OPTCR register is loaded from the SWAP_BANK_OPT option bit **only after system reset or POR**.

To change the SWAP_BANK bit (for example to apply a new firmware update), respect the sequence below:

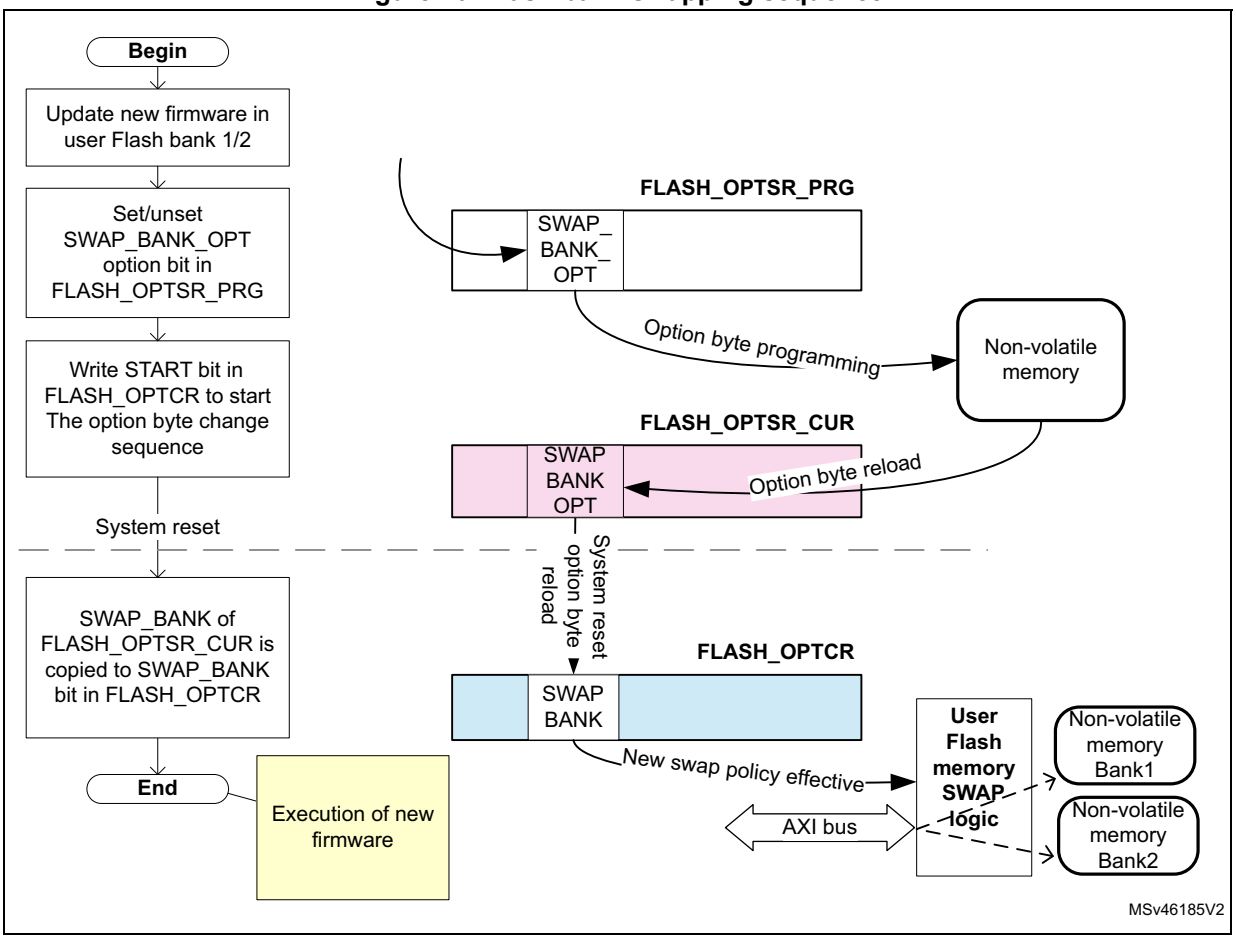
1. Unlock OPTLOCK bit, if not already unlocked.
2. Set the new desired SWAP_BANK_OPT value in the FLASH_OPTSR_PRG register.
3. Start the option byte change sequence by setting the OPTSTART bit in the FLASH_OPTCR register.
4. Once the option byte change has completed, FLASH_OPTSR_CUR contains the expected SWAP_BANK_OPT value, but SWAP_BANK bit in FLASH_OPTCR has not yet been modified and the bank swapping is not yet effective.
5. Force a system reset or a POR. When the reset rises up, the bank swapping is effective (SWAP_BANK value updated in FLASH_OPTCR) and the new firmware shall be executed.

Note: The SWAP_BANK bit in FLASH_OPTCR is read-only and cannot be modified by the application software.

The SWAP_BANK_OPT option bit in FLASH_OPTSR_PRG can be modified whatever the RDP level (i.e. even in level 2), thus allowing advanced firmware upgrade in any level of readout protection.

Figure 10 gives an overview of the bank swapping sequence.

Figure 10. Flash bank swapping sequence



Configuration and option byte register swapping

The embedded Flash memory bank swapping option controlled by the SWAP_BANK bit also swaps the two sets of configuration and option byte registers, as shown in *Table 15*. One set of registers is related to bank 1 while the other is related to bank 2. Since some registers are not specific to any particular bank, they are mapped onto two different addresses so that the swapping does not affect the access to these registers.

Table 15. Flash register map vs swapping option

Address offset (1)		Register name		Register targeting bank (1)			
				SWAP_BANK=0		SWAP_BANK=1	
0x000 or 0x100		FLASH_ACR		N/A		N/A	
0x004	0x104	FLASH_KEYR1	FLASH_KEYR2	Bank 1	Bank 2	Bank 2	Bank 1
0x008 or 0x108		FLASH_OPTKEYR		N/A		N/A	
0x00C	0x10C	FLASH_CR1	FLASH_CR2	Bank 1	Bank 2	Bank 2	Bank 1
0x010	0x110	FLASH_SR1	FLASH_SR2	Bank 1	Bank 2	Bank 2	Bank 1
0x014	0x114	FLASH_CCR1	FLASH_CCR2	Bank 1	Bank 2	Bank 2	Bank 1
0x018 or 0x118		FLASH_OPTCR		N/A		N/A	
0x01C or 0x11C		FLASH_OPTSR_CUR		N/A		N/A	
0x020 or 0x120		FLASH_OPTSR_PRG		N/A		N/A	
0x024 or 0x124		FLASH_OPTCCR		N/A		N/A	
0x028	0x128	FLASH_PRAR_CUR1	FLASH_PRAR_CUR2	Bank 1	Bank 2	Bank 2	Bank 1
0x02C	0x12C	FLASH_PRAR_PRG1	FLASH_PRAR_PRG2	Bank 1	Bank 2	Bank 2	Bank 1
0x030	0x130	FLASH_SCAR_CUR1	FLASH_SCAR_CUR2	Bank 1	Bank 2	Bank 2	Bank 1
0x034	0x134	FLASH_SCAR_PRG1	FLASH_SCAR_PRG2	Bank 1	Bank 2	Bank 2	Bank 1
0x038	0x138	FLASH_WPSGN_CUR1	FLASH_WPSGN_CUR2	Bank 1	Bank 2	Bank 2	Bank 1
0x03C	0x13C	FLASH_WPSGN_PRG1	FLASH_WPSGN_PRG2	Bank 1	Bank 2	Bank 2	Bank 1
0x040 or 0x140		FLASH_BOOT_CUR		N/A		N/A	
0x044 or 0x144		FLASH_BOOT_PRG		N/A		N/A	
0x050	0x150	FLASH_CRCCR1	FLASH_CRCCR2	Bank 1	Bank 2	Bank 2	Bank 1
0x054	0x154	FLASH_CRCSADD1R	FLASH_CRCSADD2R	Bank 1	Bank 2	Bank 2	Bank 1
0x058	0x158	FLASH_CRCEADD1R	FLASH_CRCEADD2R	Bank 1	Bank 2	Bank 2	Bank 1
0x05C	0x15C	FLASH_CRCDATAR		N/A		N/A	
0x060	0x160	FLASH_ECC_FA1R	FLASH_ECC_FA2R	Bank 1	Bank 2	Bank 2	Bank 1

1. As shown above, some registers are not dedicated to a specific bank and can be accessed at two different addresses.

3.3.14 FLASH reset and clocks

Reset management

The embedded Flash memory can be reset by a D1 domain reset (d1_rst), driven by the reset and clock control (RCC). The main effects of this reset are the following:

- All registers, except for option byte registers, are cleared, including read and write latencies. If the bank swapping option is changed, it will be applied.
- Most control registers are automatically protected against write operations. To unprotect them, new unlock sequences must be used as described in *Section 3.5.1: FLASH configuration protection*.

The embedded Flash memory can be reset by a power-on reset (po_rst), driven by the reset and clock control (RCC). When the reset falls, all option byte registers are reset. When the reset rises up, the option bytes are loaded, potentially applying new features. During this loading sequence, the device remains under reset and the embedded Flash memory is not accessible.

The Reset signal can have a critical impact on the embedded Flash memory:

- The contents of the Flash memory are not guaranteed if a device reset occurs during a Flash memory write or erase operation.
- If a reset occurs while the option byte modification is ongoing, the old option byte values are kept. When it occurs, a new option byte modification sequence is required to program the new values.

Clock management

The embedded Flash memory uses the microcontroller system clock (sys_ck), here the AXI interface clock.

Depending on the device clock and internal supply voltage, specific read and write latency settings usually need to be set in the Flash access control register (FLASH_ACR), as explained in *Section 3.3.8: FLASH read operations* and *Section 3.3.9: FLASH program operations*.

3.4 FLASH option bytes

3.4.1 About option bytes

The embedded Flash memory includes a set of non-volatile option bytes. They are loaded at power-on reset and can be read and modified only through configuration registers.

These option bytes are configured by the end-user depending on the application requirements. Some option bytes might have been initialized by STMicroelectronics during manufacturing stage.

This section documents:

- When option bytes are loaded
- How application software can modify them
- What is the detailed list of option bytes, together with their default factory values (i.e. before the first option byte change).

3.4.2 Option byte loading

There are multiple ways of loading the option bytes into embedded Flash memory:

1. **Power-on wakeup**

   When the device is first powered, the embedded Flash memory automatically loads all the option bytes. During the option byte loading sequence, the device remains under reset and the embedded Flash memory cannot be accessed.

2. **Wakeup from system Standby**

   When the D1 power domain, which contains the embedded Flash memory, is switched from DStandby mode to DRun mode, the embedded Flash memory behaves as during a power-on sequence.

3. **Ad-hoc option byte reloading by the application**

   When the user application successfully modifies the option byte content through the embedded Flash memory registers, the non-volatile option bytes are programmed and the embedded Flash memory automatically reloads all option bytes to update the option registers.

Note: The option bytes read sequence is enhanced thanks to a specific error correction code.

In case of security issue, the option bytes may be loaded with default values (see Section 3.4.3: Option byte modification).

3.4.3 Option byte modification

Changing user option bytes

A user option byte change operation can be used to modify the configuration and the protection settings saved in the non-volatile option byte area of memory bank 1.

The embedded Flash memory features two sets of option byte registers:

- The first register set contains the current values of the option bytes. Their names have the _CUR extension. All “_CUR” registers are read-only. Their values are automatically loaded from the non-volatile memory after power-on reset, wakeup from system standby or after an option byte change operation.
- The second register set allows the modification of the option bytes. Their names contain the _PRG extension. All “_PRG” registers can be accessed in read/write mode.

When the OPTLOCK bit in FLASH_OPTCR register is set, modifying the _PRG registers is not possible.

When OPTSTART bit is set to 1, the embedded Flash memory checks if at least one option byte needs to be programmed by comparing the current values (_CUR) with the new ones (_PRG). If this is the case and all the other conditions are met (see *Changing security option bytes*), the embedded Flash memory launches the option byte modification in its non-volatile memory and updates the option byte registers with _CUR extension.

If one of the condition described in *Changing security option bytes* is not respected, the embedded Flash memory sets the OPTCHANGEERR flag to 1 in the FLASH_OPTSR_CUR register and aborts the option byte change operation. In this case, the _PRG registers are not overwritten by current option value. The user application can check what was wrong in their configuration.

Unlocking the option byte modification

After reset, the OPTLOCK bit is set to 1 and the FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option bytes. The FLASH_OPTCR unlock sequence is described in *Section 3.5.1: FLASH configuration protection*.

Option byte modification sequence

To modify user option bytes, follow the sequence below:

1. Unlock FLASH_OPTCR register as described in *Section 3.5.1: FLASH configuration protection*, unless the register is already unlocked.
2. Write the desired new option byte values in the corresponding option registers (FLASH_XXX_PRG1/2).
3. Set the option byte start change OPTSTART bit to 1 in the FLASH_OPTCR register.
4. Wait until OPTBSY bit is cleared.

Note: If a reset or a power-down occurs while the option byte modification is ongoing, the original option byte value is kept. A new option byte modification sequence is required to program the new value.

Changing security option bytes

On top of OPTLOCK bit, there is a second level of protection for security-sensitive option byte fields. Specific rules must be followed to update them:

- **Readout protection (RDP)**

  A detailed description of RDP option bits is given in *Section 3.5.3*. The following rules must be respected to modify these option bits:

  – When RDP is set to level 2, no changes are allowed (except for the SWAP bit). As a result, if the user application attempts to reduce the RDP level, an option byte change error is raised (OPTCHANGEERR bit in FLASH_OPTSR_CUR register), and all the programmed changes are ignored.

  – When the RDP is set to level 1, requiring a change to level 2 is always allowed. When requiring a regression to level 0, an option byte change error can occur if some of the recommendations provided in this chapter have not been followed.

  – When the RDP is set to level 0, switching to level 1 or level 2 is possible without any restriction.

- **Sector write protection (WRPSn1/2)**

  These option bytes manage sector write protection in FLASH_WPSN_CUR1/2R registers. They can be changed without any restriction when the RDP protection level is different from level 2.

- **PCROP area size (PROT_AREA_START1/2 and PROT_AREA_END1/2)**

  These option bytes configure the size of the PCROP areas in FLASH_PRAR_CUR1/2 registers. They can be increased without any restriction by the Arm® Cortex®-M7 core. To remove or reduce a PCROP area, an RDP level 1 to 0 regression (see *Section 3.5.3*) or a bank erase with protection removal (see *Section 3.3.10*) must be

requested at the same time. DMEP must be set to 1 in either FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2, otherwise an option byte change error is raised.

- **DMEP1/2**

  When this option bit is set, the content of the corresponding PCROP area is erased during a RDP level 1 to 0 regression (see *Section 3.5.3*) or a bank erase with protection removal (see *Section 3.3.10*). It is preserved otherwise.

  There are no restrictions in setting DMEP1/2 bit. Resetting DMEP1/2 bit from 1 to 0 can only be done when an RDP level 1 to 0 regression or a bank erase with protection removal is requested at the same time.

- **Secure access mode (SECURITY)**

  The SECURITY option bit activates the secure access mode described in *Section 3.5.5*. This option bit can be freely set by the application software if such mode is activated on the device. If at least one PCROP or secure-only area is defined as not null, the only way to deactivate the security option bit (from 1 to 0) is to perform an RDP level 1 to 0 regression, when DMEP1/2 is set to 1 in either FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2 registers, and DMES1/2 is set to 1 in either FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2.

  If no valid secure-only area and no valid PCROP area are currently defined, the SECURITY option bit can be freely reset.

Note: It is recommended to have both SEC_AREA_START> SEC_AREA_END and PROT_AREA_START> PROT_AREA_END programmed when deactivating the SECURITY option bit during an RDP level 1 to 0 regression.

- **Secure-only area size (SEC_AREA_START1/2 and SEC_AREA_END1/2)**

  These option bytes configure the size of the secure-only areas in FLASH_SCAR_CUR1/2 registers. They can be changed without any restriction by the user secure application or by the ST secure library running on the device. For user non-secure application, the secure-only area size can be removed by performing a bank erase with protection removal (see *Section 3.3.10*), or an RDP level 1 to 0 regression when DMES1/2 set to 1 in either FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2 (otherwise an option byte change error is raised).

- **DMES1/2**

  When this option bit is set, the content of the corresponding secure-only area is erased during an RDP level 1 to 0 regression or a bank erase with protection removal, it is preserved otherwise.

  DMES1/2 bits can be set without any restriction. Resetting DMES1/2 bit from 1 to 0 can only be performed when an RDP level 1 to 0 regression or a bank erase with protection removal is requested at the same time.

- **Secure DTCM size (ST_RAM_SIZE)**

  These bits selects the size of the secure DTCM, an option that is available only when SECURITY option bit is set. It can be modified only when the CPU is running the ST secure library.

3.4.4 Option bytes overview

Table 16 lists all the user option bytes managed through the embedded Flash memory registers, as well as their default values before the first option byte change (default factory value).

Table 16. Option byte organization

Register	Bitfield															
FLASH_OPTSR[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	SWAP_BANK_OP	OPTCHANGEERR	IO_HSLV	Res.	Res.	Res.	Res.	Res	Res.	Res.	SECURITY	ST_RAM_SIZE		IWDG_FZ_SDBY	IWDG_FZ_STOP	Res.
Default factory value	*0*	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*1*	*1*	*0*
FLASH_OPTSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	RDP[7:0]								NRST_STDY_D1	NRST_STOP_D1	Res.	IWDG1_SW	BOR_LEV		Res.	Res.
Default factory value	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
FLASH_BOOT[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	BOOT_ADD1[15:0]															
Default factory value	*0x1FF0*															
FLASH_BOOT[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	BOOT_ADD0[15:0]															
Default factory value	*0x0800*															
FLASH_PRAR_x1[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	DMEP1	Res.	Res.	Res.	PROT_AREA_END1											
Default factory value	*0*	*0*	*0*	*0*	*0x000*											
FLASH_PRAR_x1[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	PROT_AREA_START1											
Default factory value	*0*	*0*	*0*	*0*	*0x0FF*											

Table 16. Option byte organization (continued)

Register	Bitfield															
FLASH_PRAR_x2[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	DMEP2	Res.	Res.	Res.	PROT_AREA_END2											
Default factory value	*0*	*0*	*0*	*0*	*0x000*											
FLASH_PRAR_x2[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	PROT_AREA_START2											
Default factory value	*0*	*0*	*0*	*0*	*0x0FF*											
FLASH_SCAR_x1[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	DMES1	Res.	Res.	Res.	SEC_AREA_END1											
Default factory value	*1*	*0*	*0*	*0*	*0x000*											
FLASH_SCAR_x1[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	SEC_AREA_START1											
Default factory value	*0*	*0*	*0*	*0*	*0x0FF*											
FLASH_SCAR_x2[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	DMES2	Res.	Res.	Res.	SEC_AREA_END2											
Default factory value	*1*	*0*	*0*	*0*	*0x000*											
FLASH_SCAR_x2[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	SEC_AREA_START2											
Default factory value	*0*	*0*	*0*	*0*	*0x0FF*											
FLASH_WPSN_x1[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
Default factory value	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
FLASH_WPSN_x1[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn1[7:0]							
Default factory value	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_WPSN_x2[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 16. Option byte organization (continued)

Register	Bitfield															
Default factory value	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
FLASH_WPSN_x2[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn2[7:0]							
Default factory value	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*

3.4.5 Description of user and system option bytes

Below the list of the general-purpose option bytes that can be used by the application:

- Watchdog management
  - IWDG_FZ_STOP: independent watchdog (IWDG1) counter active in Stop mode if 1 (stop counting or freeze if 0)
  - IWDG_FZ_SDBY: independent watchdog (IWDG1) counter active in Standby mode if 1 (stop counting or freeze if 0)
  - IWDG1_SW: hardware (0) or software (1) IWDG1 watchdog control selection

Note: If the hardware watchdog "control selection" feature is enabled (set to 0), the watchdog is automatically enabled at power-on, thus generating a reset unless the watchdog key register is written to or the down-counter is reloaded before the end-of-count is reached.

Depending on the configuration of IWDG_STOP and IWDG_STBY options, the IWDG can continue counting (1) or not (0) when the device is in Stop or Standby mode, respectively. When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

- Reset management
  - BOR: Brownout level option, indicating the supply level threshold that activates/releases the reset
  - NRST_STDBY_D1/2: generates a reset when D1 (respectively D2) domain enters DStandby mode. It is active low.
  - NRST_STOP_D1/2: generates a reset when D1 (respectively D2) domain enters DStop mode. It is active low.

Note: Whenever a Standby (respectively Stop) mode entry sequence is successfully executed, the device is reset instead of entering Standby (respectively Stop) mode if NRST_STDBY (respectively NRST_STOP) is cleared to 0.

- Bank swapping (see *Section 3.3.13 on page 154*)

  SWAP_BANK_OPT: bank swapping option, set to 1 to swap user sectors and registers after boot.
- Device options
  - IO_HSLV: I/O speed optimization at low-voltage if set to 1.

When STMicroelectronics delivers the device, the values programmed in the general-purpose option bytes are the following:

- Watchdog management
  - IWDG1 active in Standby and Stop modes (option byte value = 0x1)
  - IWDG1 not automatically enabled at power-on (option byte value = 0x1)
- Reset management:
  - BOR: brownout level option (reset level) equals brownout reset threshold 0 (option byte value = 0x0)
  - A reset is not generated when D1 or D2 domain enters DStandby or DStop low-power mode (option byte value = 0x1)
- No bank swapping (option byte value = 0x0)
- Device working in the full voltage range with I/O speed optimization at low-voltage disabled (IO_HSLV=0)

Refer to *Section 3.9: FLASH registers* for details.

3.4.6 Description of data protection option bytes

Below the list of the option bytes that can be used to enhance data protection:

- RDP[7:0]: Readout protection level (see *Section 3.5.3 on page 168* for details).
- WRPSn1/2: write protection option of the corresponding Bank 1 (respectively Bank 2) sector. It is active low. Refer to *Section 3.5.4 on page 172* for details.
- PROT_AREAx: Proprietary code readout protection (refer to *Section 3.5.4 on page 172* for details)
  - PROT_AREA_START1 (respectively PROT_AREA_END1) contains the first (respectively last) 256-byte block of the PCROP zone in Bank 1
  - PROT_AREA_START2 (respectively PROT_AREA_END2) contains the first (respectively last) 256-byte block of the PCROP zone in Bank 2
  - DMEP1/2: when set to 1, the PCROP area in Bank 1 (respectively Bank 2) is erased during a RDP protection level regression (change from level 1 to 0) or a bank erase with protection removal.
- SEC_AREAx: secure access only zones definition (refer to *Section 3.5.5 on page 173* for details).
  - SEC_AREA_START1 (respectively SEC_AREA_END1) contains the first (respectively last) 256-byte block of the secure access only zone in Bank 1
  - SEC_AREA_START2 (respectively SEC_AREA_END2) contains the first (respectively last) 256-byte block of the secure access only zone in Bank 2
  - DMES1/2: when set to 1 the secure access only zone in Bank 1 (respectively Bank 2) is erased during a RDP protection level regression (change from level 1 to 0), or a bank erase with protection removal.
- SECURITY: this non-volatile option can be used by the application to manage secure access mode, as described in *Section 3.5.5*.
- ST_RAM_SIZE: option to set the size of DTCM RAM to be protected (2kB, 4kB, 8kB or 16kB). Only valid if SECURITY is set to 1.

When STMicroelectronics delivers the device, the values programmed in the data protection option bytes are the following:

- RDP level 0 (option byte value = 0xAA)
- Flash bank erase operations impact secure-only data areas when enabled by the application (DMES1/2=1). They doe not impact PCROP data areas (DMEP1/2=0).
- PCROP and secure-only zone protections disabled (start addresses higher than end addresses)
- Write protection enabled (all option byte bits set to 1)
- Secure access mode disabled (SECURITY option byte value= 0)

Refer to *Section 3.9: FLASH registers* for details.

3.4.7 Description of boot address option bytes

Below the list of option bytes that can be used to configure the appropriate boot address for your application:

- BOOT_ADD0/1: MSB of the Arm® Cortex®-M7 boot address when BOOT pin is low (respectively high)

When STMicroelectronics delivers the device, the values programmed in the data protection option bytes are the following:

- Arm® Cortex®-M7 boot address (MSB): 0x0800 (BOOT pin low for user Flash memory) and 0x1FF0 (BOOT pin high for System Flash memory)

Refer to *Section 3.9: FLASH registers* for details.

3.5 FLASH protection mechanisms

Since sensitive information can be stored in the Flash memory, it is important to protect it against unwanted operations such as reading confidential areas, illegal programming of protected area, or illegal Flash memory erasing.

The embedded Flash memory implements the following protection mechanisms that can be used by end-user applications to manage the security of embedded non-volatile storage:

- Configuration protection
- Global device Readout protection (RDP)
- Write protection
- Proprietary code readout protection (PCROP)
- Secure access mode areas

This section provides a detailed description of all these security mechanisms.

3.5.1 FLASH configuration protection

The embedded Flash memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (e.g. software bugs):

- Option bytes change
- Write operations
- Erase commands
- Interrupt masking

More specifically, write operations to embedded Flash memory control registers (FLASH_CR1/2 and FLASH_OPTCR) are not allowed after reset.

The following sequence must be used to unlock FLASH_CR1/2 register:

1. Program KEY1 to 0x45670123 in FLASH_KEYR1/2 key register.
2. Program KEY2 to 0xCDEF89AB in FLASH_KEYR1/2 key register.
3. LOCK1/2 bit is now cleared and FLASH_CR1/2 is unlocked.

The following sequence must be used to unlock FLASH_OPTCR register:

1. Program OPTKEY1 to 0x08192A3B in FLASH_OPTKEYR option key register.
2. Program OPTKEY2 to 0x4C5D6E7F in FLASH_OPTKEYR option key register.
3. OPTLOCK bit is now cleared and FLASH_OPTCR register is unlocked.

Any wrong sequence locks up the corresponding register/bit until the next system reset, and generates a bus error.

The FLASH_CR1/2 (respectively FLASH_OPTCR) register can be locked again by software by setting the LOCK1/2 bit in FLASH_CR1/2 register (respectively OPTLOCK bit in FLASH_OPTCR register).

In addition the FLASH_CR1/2 register remains locked and a bus error is generated when the following operations are executed:

- programming a third key value
- writing to a different register belonging to the same bank than FLASH_KEYR1/2 before FLASH_CR1/2 has been completely unlocked (KEY1 programmed but KEY2 not yet programmed)
- writing less than 32 bits to KEY1 or KEY2.

Similarly the FLASH_OPTCR register remains locked and a bus error is generated when the following operations are executed:

- programming a third key value
- writing to a different register before FLASH_OPTCR has been completely unlocked (OPTKEY1 programmed but OPTKEY2 not yet programmed)
- writing less than 32 bits to OPTKEY1 or OPTKEY2.

The embedded Flash memory configuration registers protection is summarized in *Table 17*.

Table 17. Flash interface register protection summary

Register name	Unlocking register	Protected asset
FLASH_ACR	N/A	-
FLASH_KEYR1/2	N/A	-
FLASH_OPTKEYR	N/A	-
FLASH_CR1/2	FLASH_KEYR1/2	Write operations Erase commands Interrupt generation masking sources
FLASH_SR1/2	N/A	-
FLASH_CCR1/2	N/A	-
FLASH_OPTCR	FLASH_OPTKEYR	Option bytes change Mass erase

Table 17. Flash interface register protection summary (continued)

Register name	Unlocking register	Protected asset
FLASH_OPTSR_PRG	FLASH_OPTCR	Option bytes change
FLASH_OPTCCR	N/A	-
FLASH_PRAR_PRG1/2	FLASH_OPTCR	Option bytes (PCROP)
FLASH_SCAR_PRG1/2	FLASH_OPTCR	Option bytes (security)
FLASH_WPSGN_PRG1/2	FLASH_OPTCR	Option bytes (write protection)
FLASH_BOOT_PRGR	FLASH_OPTCR	Option bytes (boot)
FLASH_CRCCR1/2	N/A	-
FLASH_CRCSADD1/2R		-
FLASH_CRCEADD1/2R		-
FLASH_CRCDATAR		-
FLASH_ECC_FA1/2R	N/A	-

3.5.2 Write protection

The purpose of embedded Flash memory write protection is to protect the embedded Flash memory against unwanted modifications of the non-volatile code and/or data.

Any 128-Kbyte Flash sector can be independently write-protected or unprotected by clearing/setting the corresponding WRPSn1/2 bit in the FLASH_WPSN_PRG1/2R register.

A write-protected sector can neither be erased nor programmed. As a result, a bank erase cannot be performed if one sector is write-protected, unless a bank erase is executed during an RDP level 1 to 0 regression (see *Section : Flash bank erase with automatic protection-removal sequence* for details).

The embedded Flash memory write-protection user option bits can be modified without any restriction when the RDP level is set to level 0 or level 1. When it is set to level 2, the write protection bitfield can no more be changed in the option bytes.

Note: PCROP or secure-only areas are write and erase protected.

Write protection errors are documented in *Section 3.7: FLASH error management*.

3.5.3 Readout protection (RDP)

The embedded Flash memory readout protection is global as it does not apply only to the embedded Flash memory, but also to the other secured regions. This is done by using dedicated security signals.

In this section *other secured regions* are defined as:

- Backup SRAM
- RTC backup registers

The global readout protection level is set by writing the values given in *Table 18* into the readout protection (RDP) option byte (see *Section 3.4.3: Option byte modification*).

Table 18. RDP value vs readout protection level

RDP option byte value	Global readout protection level
0xAA	Level 0
0xCC	Level 2
Any other value	Level 1(1)

1. Default protection level when RDP option byte is erased.

Definitions of RDP global protection level

RDP Level 0 (no protection)

When the global read protection level 0 is set, all read/program/erase operations from/to the user Flash memory are allowed (if no others protections are set). This is true whatever the boot configuration (boot from user or system Flash memory, boot from RAM), and whether the debugger is connected to the device or not. Accesses to the *other secured regions* are also allowed.

RDP Level 1 (Flash memory content protection)

When the global read protection level 1 is set, the below properties apply:

- The Flash memory content is protected against debugger and potential malicious code stored in RAM. Hence as soon as any debugger is connected or has been connected, or a boot is configured in embedded RAM (intrusion), the embedded Flash memory prevents any accesses to Flash memory.
- When no intrusion is detected (no boot in RAM, no boot in System Flash memory and no debugger connected), all read/program/erase operations from/to the user Flash memory are allowed (if no others protections are set). Accesses to the *other secured regions* are also allowed.
- When an intrusion is detected, no accesses to the user Flash memory can be performed. A bus error is generated when a read access is requested to the Flash memory. In addition, no accesses to *other secured regions* (read or write) can be performed.
- When performing an RDP level regression, i.e. programming the RDP protection to level 0, the user Flash memory and the *other secured regions* are mass erased, as described in *RDP protection transitions*.
- When booting from STMicroelectronics non-secure bootloader, only the identification services are available (GET_ID_COMMAND, GET_VER_COMMAND and GET_CMD_COMMAND).

RDP Level 2 (device protection and intrusion prevention)

When the global read protection level 2 is set, the below rules apply:

- All debugging features are disabled.
- Like level 0, all read/write/erase operations from/to the user Flash memory are allowed since the debugger and the boot from RAM and System Flash memory are disabled. Accesses to the *other secured regions* are also allowed.
- Booting from RAM is no more allowed.
- The user option bits described in *Section 3.4* can no longer be changed except for the SWAP bit.

Caution: Memory read protection level 2 is an irreversible operation. When level 2 is activated, the level of protection cannot be changed back to level 0 or level 1.

Note: The JTAG port is permanently disabled when level 2 is active (acting as a JTAG fuse). As a consequence. STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set.

Apply a power-on reset if the global read protection level 2 is set while the debugger is still connected.

The above RDP global protection is summarized in *Table 19*.

Table 19. Protection vs RDP Level(1)

Boot area	Inputs		Effects				Comment
	RDP	Debugger connected	User Flash memory access(2)	System Flash memory access (3)	Other secured regions	Option Bytes access	
User Flash memory	Level 0	Yes(4)/No	R/W/E	R	R/W	R/W	-
	Level 1	Yes(3)	illegal access(5)	R	no access	R/W	
	Level 1	No	R/W/E	R	R/W	R/W	
	Level 2	No	R/W/E	R	R/W	R	
RAM or System Flash memory	Level 0	Yes(3)/No	R/W/E	R	R/W	R/W	-
	Level 1	Yes(3)/No	illegal access(5)	R	no access	R/W	When selected, only ST basic bootloader is executed
	Level 2	No	Not applicable				No boot from RAM or ST system Flash memory in RDP level 2

1. R = read, W =write, E = erase.
2. PCROP (see *Section 3.5.4)* and secure-only access control (see *Section 3.5.5)* applies.
3. Read accesses to secure boot and secure libraries stored in system Flash bank 1 possible only from STMicroelectronics code.
4. JTAG interface disabled while secure libraries are executed.
5. *Read protection error (RDPERR)* with bus error on read operations, *Write protection error (WRPERR)* on write/erase operations.

RDP protection transitions

Figure 11 shows how to switch from one RDP level to another. The transition is effective after successfully writing the option bytes including RDP (refer to *Section 3.4.3* for details on how to change the option bytes).

Figure 11. RDP protection transition scheme

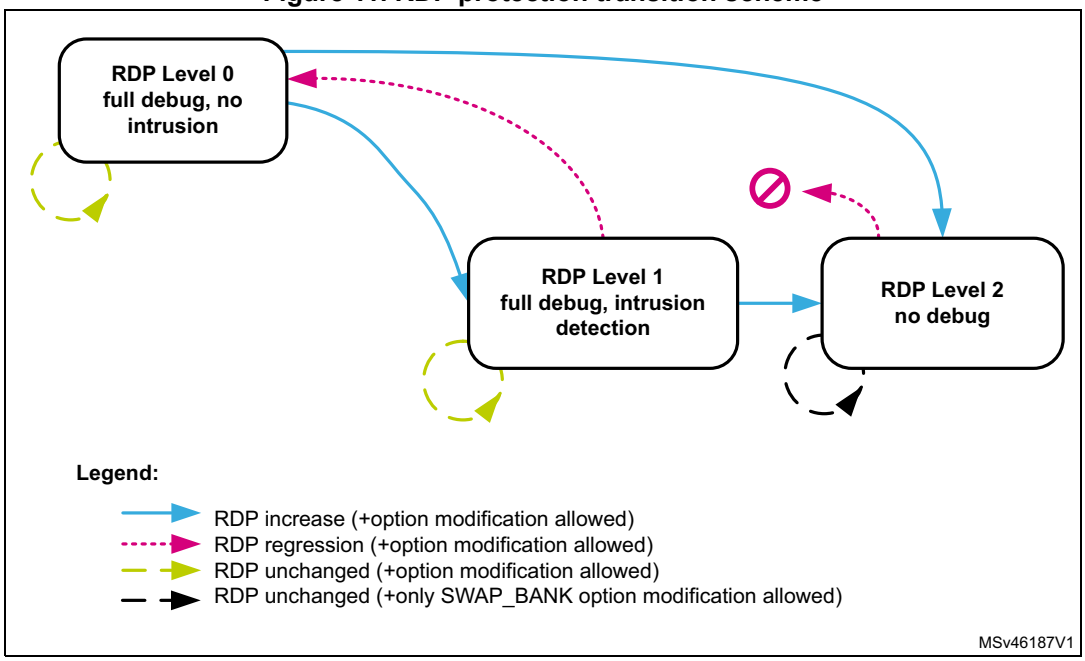


Table 20 details the RDP transitions and their effects on the product.

Table 20. RDP transition and its effects

RDP transition		RDP option update	Effect on device		
Level before change	Level after change		Debugger disconnect	Option bytes change (1)	Mass erase
L0	L1	not 0xAA and not 0xCC	No	Allowed	No
	L2	0xCC	**Yes**	**Not allowed**	No
L1	L2	0xCC	**Yes**	**Not allowed**	No
	L0	0xAA	No	Allowed	**Yes**
L0	L0	0xAA	No	Allowed	No
L1	L1	not 0xAA and not 0xCC			

1. Except for bank swapping option bit.

When the current RDP level is RDP level 1, requesting a new RDP level 0 causes a full mass erase:

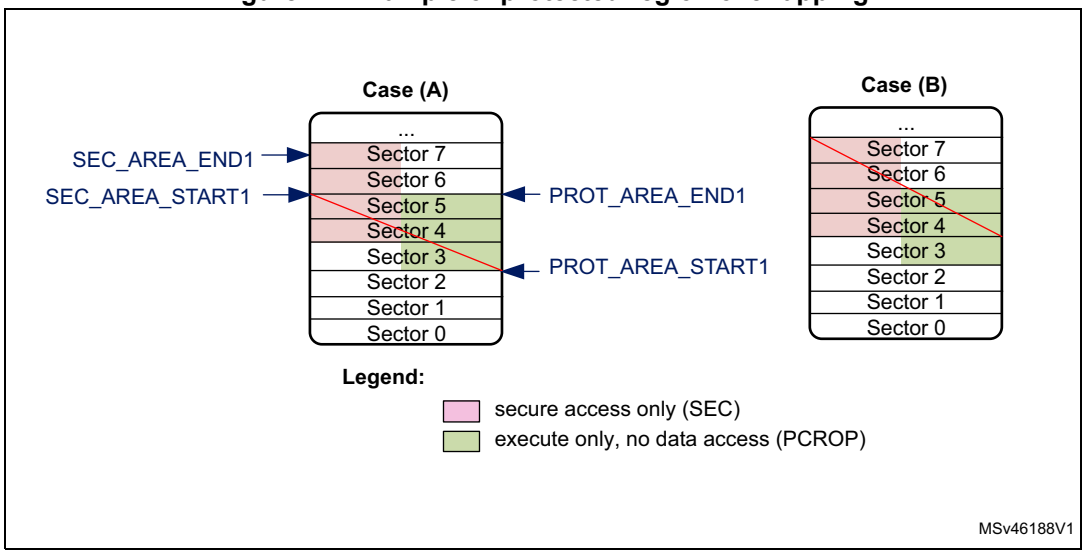
- The user Flash memory area of the embedded Flash memory is mass erased:
  - A partial sector erase occurs if PCROP (respectively secure-only) areas are preserved by the application. It happens when both DMEP1/2 bits (respectively DMES1/2 bits) are cleared to 0 in FLASH_PRAR_CUR1/2 and FLASH_PRAR_PRG1/2 (respectively FLASH_SCAR_CUR1/2 and FLASH_SCAR_PRG1/2). The sectors belonging to the preserved area(s) are not erased.
  - A full bank erase occurs when at least one DMEP1/2 bit is set to 1 in FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2, and at least one DMES1/2 bit is set to 1 in FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2.

Note: Data in write protection area are not preserved during RDP regression.

- The other secured regions are also erased (Backup SRAM and RTC backup registers)

If a PCROP area overlaps with a secure-only area, the embedded Flash memory performs the erase operation (see strike-through areas in red in *Figure 12*). If a level regression is requested while DMES1/2 or DMEP1/2 is equal to 1, the sectors are erased. However, the sector protections are removed only if the protected sector boundaries are modified by the user application. In case **A** DMEP1/2 is set in FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2 (DMES1/2 is cleared), while in case **B** DMES1/2 is set in FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2 (DMEP is cleared).

Figure 12. Example of protected region overlapping



About RDP protection errors

Whatever the RDP level, the corresponding error flag is raised when an illegal read or write access is detected (see *Section 3.7: FLASH error management*).

3.5.4 Proprietary code readout protection (PCROP)

The embedded Flash memory allows the definition of an “executable-only” area in the user area of each Flash memory bank. In this area, only instruction fetch transactions from the system, i.e. no data access (data or literal pool) are allowed. This protection is particularly efficient to protect third party software intellectual property.

Note: Executable-only area usage requires the native code to be compiled accordingly using “execute-only” option.

PCROP area programming

One PCROP area can be defined in bank 1 (respectively bank 2) by setting the PROT_AREA_END1 and PROT_AREA_START1 (respectively PROT_AREA_END2 and PROT_AREA_START2) option bytes so that the END address is strictly higher than the START address. PROT_AREA_START and PROT_AREA_END are defined with a granularity of 256 bytes. This means that the actual PCROP area size (in bytes) is defined by:

[(PROT_AREA_END - PROT_AREA_START) + 1] x 256

As an example, to set a PCROP area on the first 4 Kbytes of user bank 1 (i.e. from address 0x0800 0000 to address 0x0800 0FFF, both included), the embedded Flash memory must be configured as follows:

- PROT_AREA_START1[11:0] = 0x000
- PROT_AREA_END1[11:0] = 0x00F

The protected area size defined above is equal to:

[(PROT_AREA_END - PROT_AREA_START) + 1] x 256 = 16 x 256 bytes = 4 Kbytes.

The minimum execute-only PCROP area that can be set is 16 Flash words (or 512 bytes). The maximum area is the whole user Flash memory, configured by setting to the same value the PCROP area START and END addresses.

Note: It is recommended to align the PCROP area size with the Flash sector granularity in order to avoid access right issues.

PCROP area properties

Each valid PCROP area has the following properties:

- Arm® Cortex®-M7 debug events are ignored while executing code in this area.
- Only the CPU can access it (Master ID filtering), using only instruction fetch transactions. In all other cases, accessing the PCROP area is illegal (see below).
- Illegal transactions to a PCROP area (i.e. data read or write, not fetch) are managed as below:
  - Read operations return a zero, write operations are ignored.
  - No bus error is generated but an error flag is raised (RDPERR for read, WRPERR for write).
- A valid PCROP area is erase-protected. As a result:
  - No erase operations to a sector located in this area is possible (including the sector containing the area start address and the end address)

  - No mass erase can be performed if a single valid PCROP area is defined, except during level regression or erase with protection removal.
- Only the CPU can modify the PCROP area definition and DMEP1/2 bits, as explained in *Changing user option bytes* in *Section 3.4.3*.
- During an RDP level 1 to 0 regression where the PCROP area is not null
  - The PCROP area content is not erased if the corresponding DMEP1/2 bit are both cleared to 0 in FLASH_PRAR_CUR1/2 and FLASH_PRAR_PRG1/2 registers.
  - The PCROP area content is erased if either of the corresponding DMEP1/2 bit is set to 1 in FLASH_PRAR_CUR1/2 or FLASH_PRAR_PRG1/2 register.

For more information on PCROP protection errors, refer to *Section 3.7: FLASH error management*.

3.5.5 Secure access mode

The embedded Flash memory allows the definition of a secure-only area in the user area of each Flash memory bank. This area can be accessed only while the CPU executes secure application code. This feature is available only if the SECURITY option bit is set to 1.

Secure-only areas help isolating secure user code from application non-secure code. As an example, they can be used to protect a customer secure firmware upgrade code, a custom secure boot library or a third party secure library.

Secure-only area programming

One secure-only area can be defined in bank 1 (respectively bank 2) by setting the SEC_AREA_END1 and SEC_AREA_START1 (respectively SEC_AREA_END2 and SEC_AREA_START2) option bytes so that the END address is strictly higher than the START address. SEC_AREA_START and SEC_AREA_END are defined with a granularity of 256 bytes. This means that the actual secure-only area size (in bytes) is defined by:

[(SEC_AREA_END - SEC_AREA_START) + 1] x 256

As an example, to set a secure-only area on the first 8 Kbytes of user bank 2 (i.e. from address 0x0810 0000 to address 0x0810 1FFF, both included), the embedded Flash memory must be configured as follows:

- SEC_AREA_START2[11:0] = 0x000
- SEC_AREA_END2[11:0] = 0x01F

The secure-only area size defined above is equal to:

[(SEC_AREA_END - SEC_AREA_START) + 1] x 256 = 32 x 256 bytes = 8 Kbytes.

Note: These option bytes can be modified only by the CPU running ST security library or application secure code, except during RDP level regression or erase with protection removal.

The minimum secure-only area that can be set is 16 Flash words (or 512 bytes). The maximum area is the whole user Flash memory bank, configured by setting to the same value the secure-only area START and END addresses.

Note: It is recommended to align the secure-only area size with Flash sector granularity in order to avoid access right issues.

Secure access-only area properties

- Arm® Cortex®-M7 debug events are ignored while executing code in this area.
- Only the CPU executing ST secure library or user secure application can access it (Master ID filtering). In all other cases, accessing the secure-only area is illegal (see below).
- Illegal transactions to a secure-only area are managed as follows:
  - Data read transactions return zero. Data write transactions are ignored. No bus error is generated but an error flag is raised (RDSERR for read, WRPERR for write).
  - Read instruction transactions generate a bus error and the RDSERR error flag is raised.
- A valid secure-only area is erase-protected. As a result:
  - No erase operations to a sector located in this area are possible (including the sector containing the area start address and the end address), unless the application software is executed from a valid secure-only area.
  - No mass erase can be performed if a single valid secure-only area is defined, except during level regression, erase with protection removal or when the application software is executed from a valid secure-only area.
- Only the CPU can modify the secure-only area definition and DMES1/2 bits, as explained in *Changing user option bytes* in *Section 3.4.3*.
- During an RDP level 1 to 0 regression where the secure-only area is not null:
  - the secure-only area content is not erased if the corresponding DMES1/2 bit are both cleared to 0 in FLASH_SCAR_CUR1/2 and FLASH_SCAR_PRG1/2 registers.
  - the secure-only area content is erased if either of the corresponding DMES1/2 bit is set to 1 in FLASH_SCAR_CUR1/2 or FLASH_SCAR_PRG1/2 register.

For more information on secure-only protection errors, refer to *Section 3.7: FLASH error management*.

3.6 FLASH low-power modes

Introduction

The table below summarizes the behavior of the embedded Flash memory in the microcontroller low-power modes. The embedded Flash memory belongs to the D1 domain.

Table 21. Effect of low-power modes on the embedded Flash memory

Power mode			D1 domain voltage range	Allowed if FLASH busy	FLASH power mode (in D1 domain)
System state	D1 domain	D2 domain			
Run	DRun	DRun, DStop or DStandby	VOS1/2/3	Yes	Run
	DStop, DStandby	DStop or DStandby	Off	No	Off

Table 21. Effect of low-power modes on the embedded Flash memory (continued)

Power mode			D1 domain voltage range	Allowed if FLASH busy	FLASH power mode (in D1 domain)
System state	D1 domain	D2 domain			
Stop	DStop	DStop or DStandby	SVOS3/4/5	No	Clock gated or Stopped
	DStandby	DStop or DStandby	Off	No	Off
Standby(1)	DStandby	DStandby		No	

1. D3 domain must always be in DStandby mode. When all clocks are stopped and the CPU is in CStop, the VCORE domain is switched off.

When the system state changes or within a given system state, the embedded Flash memory might get a different voltage supply range (VOS) according to the application. The procedure to switch the embedded Flash memory into various power mode (run, clock gated, stopped, off) is described hereafter.

Note: For more information in the microcontroller power states, refer to the Power control section (PWR).

Managing the FLASH domain switching to DStop or DStandby

As explain in *Table 21*, if the embedded Flash memory informs the reset and clock controller (RCC) that it is busy (i.e. BSY1/2, QW1/2, WBNE1/2 is set), the microcontroller cannot switch the D1 domain to DStop or DStandby mode.

Note: CRC_BUSY1/2 is not taken into account.

There are two ways to release the embedded Flash memory:

- Reset the WBNE1/2 busy flag in FLASH_SR1/2 register by any of the following actions:
  a) Complete the write buffer with missing data.
  b) Force the write operation without filling the missing data by activating the FW1/2 bit in FLASH_CR1/2 register. This forces all missing data “high”.
  c) Reset the PG1/2 bit in FLASH_CR1/2 register. This disables the write buffer and consequently lead to the loss of its content.
- Poll QW1/2 busy bits in FLASH_SR1/2 register until they are cleared. This will indicate that all recorded write, erase and option change operations are complete.

The microcontroller can then switch the domain to DStop or DStandby mode.

3.7 FLASH error management

3.7.1 Introduction

The embedded Flash memory automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- *Write protection error (WRPERR)*
- *Programming sequence error (PGSERR)*
- *Strobe error (STRBERR)*
- *Inconsistency error (INCERR)*
- *Operation error (OPERR)*
- *Error correction code error (SNECCERR/DBECCERR)*
- *Read protection error (RDPERR)*
- *Read secure error (RDSERR)*
- *Option byte change error (OPTCHANGEERR)*

The application software can individually enable the interrupt for each error, as detailed in *Section 3.8: FLASH interrupts*.

Note: For some errors, the application software must clear the error flag before attempting a new operation.

Each bank has a dedicated set of error flags in order to identify which bank generated the error. They are available in Flash Status register 1 or 2 (FLASH_SR1/2).

3.7.2 Write protection error (WRPERR)

When an illegal erase/program operation is attempted to the non-volatile memory bank 1 (respectively bank 2), the embedded Flash memory sets the write protection error flag WRPERR1 (respectively WRPERR2) in FLASH_SR1 register (respectively FLASH_SR2).

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A sector belonging to a valid PCROP area (even partially)
- A sector belonging to a valid secure-only area (even partially) except if the application software is executed from a valid secure-only area
- A sector write-locked with WRPSn

An program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- The system Flash memory (bank 2 only) while the device is not executing ST bootloader code
- A user Flash sector belonging to a valid PCROP area while the device is not executing an ST secure library
- A user Flash sector belonging to a valid secure-only area while the device is not executing user secure code or ST secure library
- A user sector write-locked with WRPSn
- The bank 1 system Flash memory
- The user main Flash memory when RDP level is 1 and a debugger has been detected on the device, or the CPU has not booted from user Flash memory.
- A reserved area

When WRPERR1/2 flag is raised, the operation is rejected and nothing is changed in the corresponding bank. If a write burst operation was ongoing, WRPERR1/2 is raised each time a Flash word write operation is processed by the embedded Flash memory.

Note: WRPERR1/2 flag does not block any new erase/program operation.

Not resetting the write protection error flag (WRPERR1/2) does not generate a PGSERR error.

WRPERR1/2 flag is cleared by setting CLR_WRPERR1/2 bit to 1 in FLASH_CCR1/2 register.

If WRPERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when WRPERR1/2 flag is raised (see *Section 3.8: FLASH interrupts* for details).

3.7.3 Programming sequence error (PGSERR)

When the programming sequence to the bank 1 (respectively bank 2) is incorrect, the embedded Flash memory sets the programming sequence error flag PGSERR1 (respectively PGSERR2) in FLASH_SR1 register (respectively FLASH_SR2).

More specifically, PGSERR1/2 flag is set if one of below conditions is met:

- A write operation is requested but the program enable bit (PG1/2) has not been set in FLASH_CR1/2 register prior to the request.
- The inconsistency error (INCERR1/2) has not been cleared to 0 before requesting a new write operation.

When PGSERR1/2 flag is raised, the current program operation is aborted and nothing is changed in the corresponding bank. The corresponding write data buffer is also flushed. If a write burst operation was ongoing, PGSERR1/2 is raised at the end of the burst.

Note: When PGSERR1/2 flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. Hence it is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful by reading back the value in the Flash memory.

The PGSERR1/2 flag also blocks any new program operation. This means that PGSERR1 (respectively 2) must be cleared before starting a new program operation on bank 1 (respectively bank 2).

PGSERR1/2 flag is cleared by setting CLR_PGSERR1/2 bit to 1 in FLASH_CCR1/2 register.

If PGSERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when PGSERR1/2 flag is raised. See *Section 3.8: FLASH interrupts* for details.

3.7.4 Strobe error (STRBERR)

When the application software writes several times to the same byte in bank 1 (respectively bank 2) write buffer, the embedded Flash memory sets the strobe error flag STRBERR1 (respectively STRBERR2) in FLASH_SR1 register (respectively FLASH_SR2).

When STRBERR1/2 flag is raised, the current program operation is not aborted and new byte data replace the old ones. The application can ignore the error, proceed with the current write operation and request new write operations. If a write burst was ongoing, STRBERR1/2 is raised at the end of the burst.

STRBERR1/2 flag is cleared by setting CLR_STRBERR1/2 bit to 1 in FLASH_CCR1/2 register.

If STRBERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when STRBERR1/2 flag is raised. See *Section 3.8: FLASH interrupts* for details.

3.7.5 Inconsistency error (INCERR)

When a programming inconsistency to bank 1 (respectively bank 2) is detected, the embedded Flash memory sets the inconsistency error flag INCERR1 (respectively INCERR2) in register FLASH_SR1 (respectively FLASH_SR2).

More specifically, INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, e.g.
  - The application software starts a write operation to fill the 256-bit write buffer, but sends a new write burst request to a different Flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.
- A wrap burst request issued by a master overlaps two or more 256-bit Flash-word addresses, i.e. wrap bursts must be done within 256-bit Flash-word address boundaries.

Note: INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow the sequence below to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when INCERR1 or INCERR2 flag is raised.
2. Stop all write requests to embedded Flash memory.
3. Verify that the write operations that have been requested just before the INCERR event have been successful by reading back the programmed values from the memory.
4. Clear the corresponding INCERR1/2 bit.
5. Restart the write operations where they have been interrupted.

INCERR1/2 flag is cleared by setting CLR_INCERR1/2 bit to 1 in FLASH_CCR1/2 register.

If INCERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when INCERR1/2 flag is raised (see *Section 3.8: FLASH interrupts* for details).

3.7.6 Operation error (OPERR)

When an error occurred during a write or an erase operation to bank 1 (respectively bank 2), the embedded Flash memory sets the operation error flag OPERR1 (respectively OPERR2) in FLASH_SR1 register (respectively FLASH_SR2). This error may be caused by an incorrect non-volatile memory behavior due to cycling issues or to a previous modify operation stopped by a system reset.

When OPERR1/2 flag is raised, the current program/erase operation is aborted.

OPERR1/2 flag is cleared by setting CLR_OPERR1/2 bit to 1 in FLASH_CCR1/2 register.

If OPERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when OPERR1/2 flag is raised (see *Section 3.8: FLASH interrupts* for details).

3.7.7 Error correction code error (SNECCERR/DBECCERR)

When a single error correction is detected during a read from bank 1 (respectively bank 2) the embedded Flash memory sets the single error correction flag SNECCERR1 (respectively SNECCERR2) in FLASH_SR1 register (respectively FLASH_SR2).

When two ECC errors are detected during a read to bank 1 (respectively bank 2), the embedded Flash memory sets the double error detection flag DBECCERR1 (respectively DBECCERR2) in FLASH_SR1 register (respectively FLASH_SR2). When SNECCERR1/2 flag is raised, the corrected read data are returned. Hence the application can ignore the error and request new read operations.

If a read burst operation was ongoing, SNECCERR1/2 or DBECCERR1/2 flag is raised each time a new data is sent back to the requester through the AXI interface.

When SNECCERR1/2 or DBECCERR1/2 flag is raised, the address of the Flash word that generated the error is saved in the FLASH_ECC_FA1/2R register. If the address corresponds to an OTP or read-only area, the OTP_FAIL_ECC bit is also set to 1 in the FLASH_ECC_FA1R register. This register is automatically cleared when the associated flag that generated the error is reset.

Note: In case of successive single correction or double detection errors, only the address corresponding to the first error is stored in FLASH_ECC_FA1/2R register.

When DBECCERR1/2 flag is raised, a bus error is generated. In case of successive double error detections, a bus error is generated each time a new data is sent back to the requester through the AXI interface.

Note: It is not mandatory to clear SNECCERR1/2 or DBECCERR1/2 flags before starting a new read operation.

SNECCERR1/2 (respectively DBECCERR1/2) flag is cleared by setting to 1 CLR_SNECCERR1/2 bit (respectively CLR_DBECCERR1/2 bit) in FLASH_CCR1/2 register.

If SNECCERR1/2 (respectively DBECCERR1/2) bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when SNECCERR1/2 (respectively DBECCERR1/2) flag is raised. See *Section 3.8: FLASH interrupts* for details.

3.7.8 Read protection error (RDPERR)

When a read operation to a PCROP, a secure-only or a RDP protected area is attempted in non-volatile memory bank 1 (respectively bank 2), the embedded Flash memory sets the read protection error flag RDPERR1 (respectively RDPERR2) in FLASH_SR1 register (respectively FLASH_SR2).

When RDPERR1/2 flag is raised, the current read operation is aborted but the application can request new read operations. If a read burst was ongoing, RDPERR1/2 is raised each time a data is sent back to the requester through the AXI interface.

Note: A bus error is raised if a standard application attempts to execute on a secure-only or a RDP protected area.

RDPERR1/2 flag is cleared by setting CLR_RDPERR1/2 bit to 1 in FLASH_CCR1/2 register.

If RDPERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when RDPERR1/2 flag is raised (see *Section 3.8: FLASH interrupts* for details).

3.7.9 Read secure error (RDSERR)

When a read operation is attempted to a secure address in bank 1 (respectively bank 2), the embedded Flash memory sets the read secure error flag RDSERR1 (respectively RDSERR2) in FLASH_SR1 register (respectively FLASH_SR2). For more information, refer to *Section 3.5.5: Secure access mode*.

When RDSERR1/2 flag is raised, the current read operation is aborted and the application can request new read operations. If a read burst was ongoing, RDSERR1/2 is raised each time a data is sent back to the requester through the AXI interface.

Note: The bus error is raised only if the illegal access is due to an instruction fetch.

RDSERR1/2 flag is cleared by setting CLR_RDSERR1/2 bit to 1 in FLASH_CCR1/2 register.

If RDSERRIE1/2 bit in FLASH_CR1/2 register is set to 1, an interrupt is generated when RDSERR1/2 flag is raised (see *Section 3.8: FLASH interrupts* for details).

3.7.10 Option byte change error (OPTCHANGEERR)

When the embedded Flash memory finds an error during an option change operation, it aborts the operation and sets the option byte change error flag OPTCHANGEERR in FLASH_OPTSR_CUR register.

OPTCHANGEERR flag is cleared by setting CLR_OPTCHANGEERR bit to 1 in FLASH_OPTCCR register.

If OPTCHANGEERRIE bit in FLASH_OPTCR register is set to 1, an interrupt is generated when OPTCHANGEERR flag is raised (see *Section 3.8: FLASH interrupts* for details).

3.7.11 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On AXI system bus:
  - accesses to user Flash memory while RDP is set to 1 and a illegal condition is detected (boot from system Flash memory, boot from RAM, or debugger connected)
  - fetching to secure-only user Flash memory without the correct access rights
- On AHB configuration bus:
  - wrong key input to FLASH_KEYR1/2 or FLASH_OPTKEYR

3.8 FLASH interrupts

The embedded Flash memory can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see *Section 3.7: FLASH error management*)
  - Single ECC error correction during read operation
  - Double ECC error detection during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - Non-volatile memory write/erase error (due to cycling issues)
  - option change operation error
- Security errors (see *Section 3.7: FLASH error management*)
  - Write protection error
  - Read protection error
  - Read secure error
  - CRC computation on PCROP or secure-only area error
- Miscellaneous events (described below)
  - End of programming
  - CRC computation complete

These multiple sources are combined into a single interrupt signal, **flash_it**, which is the only interrupt signal from the embedded Flash memory that drives the NVIC (nested vectored interrupt controller).

You can individually enable or disable embedded Flash memory interrupt sources by changing the mask bits in the FLASH_CR1/2 register. Setting the appropriate mask bit to 1 enables the interrupt.

Note: Prior to writing, FLASH_CR1/2 register must be unlocked as explained in Section 3.5.1: FLASH configuration protection

Table 22 gives a summary of the available embedded Flash memory interrupt features. As mentioned in the table below, some flags need to be cleared before a new operation is triggered.

Table 22. Flash interrupt request

Interrupt event		Event flag	Enable control bit	Clear flag to resume operation	Bus error
End-of-program event	on bank 1	EOP1	EOPIE1	N/A	N/A
	on bank 2	EOP2	EOPIE2		
CRC complete event	on bank 1	CRCEND1	CRCENDIE1	N/A	N/A
	on bank 2	CRCEND2	CRCENDIE2		
Write protection error	on bank 1	WRPERR1	WRPERRIE1	No	No
	on bank 2	WRPERR2	WRPERRIE2		
Programming sequence error	on bank 1	PGSERR1	PGSERRIE1	**Yes**(1)	No
	on bank 2	PGSERR2	PGSERRIE2		
Strobe error	on bank 1	STRBERR1	STRBERRIE1	No	No
	on bank 2	STRBERR2	STRBERRIE2		
Inconsistency error	on bank 1	INCERR1	INCERRIE1	**Yes**(1)	No
	on bank 2	INCERR2	INCERRIE2		
Operation error	on bank 1	OPERR1	OPERRIE1	No	No
	on bank 2	OPERR2	OPERRIE2		
ECC single error correction event	on bank 1	SNECCERR1	SNECCERRIE1	No	No
	on bank 2	SNECCERR2	SNECCERRIE2		
ECC double error detection event	on bank 1	DBECCERR1	DBECCERRIE1	No	**Yes**
	on bank 2	DBECCERR2	DBECCERRIE2		
Read protection error	on bank 1	RDPERR1	RDPERRIE1	No	**Yes**(2)
	on bank 2	RDPERR2	RDPERRIE2		
Read secure error	on bank 1	RDSERR1	RDSERRIE1	No	No (data) **Yes** (fetch)
	on bank 2	RDSERR2	RDSERRIE2		
Option Bytes operation error	all banks	OPTCHANGEERR	OPTCHANGEERRIE	No	No

1. Programming still possible on the AXI interface that did not trigger the inconsistency error, on the Flash bank that does not have the flag bit raised. See *Section 3.7.5: Inconsistency error (INCERR)* for details.
2. Bus error occurs only when accessing RDP protected area, as defined in *Section 3.5.3: Readout protection (RDP)*

The status of the individual maskable interrupt sources described in *Table 22* (except for option byte error) can be read from the FLASH_SR1/2 register. They can be cleared by setting to 1 the adequate bit in FLASH_CCR1/2 register.

Note: No unlocking mechanism is required to clear an interrupt.

End-of-program event

Setting the end-of-operation interrupt enable bit (EOPIE1/2) in the FLASH_CR1/2 register enables the generation of an interrupt at the end of an erase operation, a program operation or an option byte change on bank 1/2. The EOP1/2 bit in the FLASH_SR1/2 register is also set when one of these events occurs.

Setting CLR_EOP1/2 bit to 1 in FLASH_CCR1/2 register clears EOP1/2 flag.

CRC end of calculation event

Setting the CRC end-of-calculation interrupt enable bit (CRCENDIE1/2) in the FLASH_CR1/2 register enables the generation of an interrupt at the end of a CRC operation on bank 1/2. The CRCEND1/2 bit in the FLASH_SR1/2 register is also set when this event occurs.

Setting CLR_CRCEND1/2 bit to 1 in FLASH_CCR1/2 register clears CRCEND1/2 flag.

3.9 FLASH registers

3.9.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000 or 0x100

Reset value: 0x0000 0037

For more details, refer to *Section 3.3.8: FLASH read operations* and *Section 3.3.9: FLASH program operations*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRHIGHFREQ		LATENCY			
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **WRHIGHFREQ**: Flash signal delay

These bits are used to control the delay between non-volatile memory signals during programming operations. The application software has to program them to the correct value depending on the embedded Flash memory interface frequency. Please refer to *Table 12* for details.

Note: No check is performed by hardware to verify that the configuration is correct.

Bits 3:0 **LATENCY**: Read latency

These bits are used to control the number of wait states used during read operations on both non-volatile memory banks. The application software has to program them to the correct value depending on the embedded Flash memory interface frequency and voltage conditions.

0000: zero wait state used to read a word from non-volatile memory

0001: one wait state used to read a word from non-volatile memory

0010: two wait states used to read a word from non-volatile memory

...

1111: 15 wait states used to read from non-volatile memory

Note: No check is performed by hardware to verify that the configuration is correct.

3.9.2 FLASH key register for bank 1 (FLASH_KEYR1)

Address offset: 0x004

Reset value: 0x0000 0000

FLASH_KEYR1 is a write-only register. The following values must be programmed consecutively to unlock FLASH_CR1 register:

1. 1st key = 0x4567 0123
2. 2nd key = 0xCDEF 89AB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY1R															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY1R															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY1R**: Non-volatile memory bank 1 configuration access unlock key

3.9.3 FLASH option key register (FLASH_OPTKEYR)

Address offset: 0x008 or 0x108

Reset value: 0x0000 0000

FLASH_OPTKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_OPTCR register:

1. 1st key = 0x0819 2A3B
2. 2nd key = 0x4C5D 6E7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEYR															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OPTKEYR**: FLASH option bytes control access unlock key

3.9.4 FLASH control register for bank 1 (FLASH_CR1)

Address offset: 0x00C

Reset value: 0x0000 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CRCENDIE1	DBECCERRIE1	SNECCERRIE1	RDSERRIE1	RDPERRIE1	OPERRIE1	INCERRIE1	Res.	STRBERRIE1	PGSERRIE1	WRPERRIE1	EOPIE1
				rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_EN	Res.	Res.	Res.	Res.	SNB1			START1	FW1	PSIZE1		BER1	SER1	PG1	LOCK1
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CRCENDIE1**: Bank 1 CRC end of calculation interrupt enable bit

When CRCENDIE1 bit is set to 1, an interrupt is generated when the CRC computation has completed on bank 1. CRCENDIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when CRC computation complete on bank 1

1: interrupt generated when CRC computation complete on bank 1

Bit 26 **DBECCERRIE1**: Bank 1 ECC double detection error interrupt enable bit

When DBECCERRIE1 bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a read operation from bank 1. DBECCERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when an ECC double detection error occurs on bank 1

1: interrupt generated if an ECC double detection error occurs on bank 1

Bit 25 **SNECCERRIE1**: Bank 1 ECC single correction error interrupt enable bit

When SNECCERRIE1 bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation from bank 1. SNECCERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when an ECC single correction error occurs on bank 1

1: interrupt generated when an ECC single correction error occurs on bank 1

Bit 24 **RDSERRIE1**: Bank 1 secure error interrupt enable bit

When RDSERRIE1 bit is set to 1, an interrupt is generated when a secure error (access to a secure-only protected address) occurs during a read operation from bank 1. RDSERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a secure error occurs on bank 1

1: an interrupt is generated when a secure error occurs on bank 1

Bit 23 **RDPERRIE1**: Bank 1 read protection error interrupt enable bit

When RDPERRIE1 bit is set to 1, an interrupt is generated when a read protection error occurs (access to an address protected by PCROP or by RDP level 1) during a read operation from bank 1. RDPERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a read protection error occurs on bank 1

1: an interrupt is generated when a read protection error occurs on bank 1

Bit 22 **OPERRIE1**: Bank 1 write/erase error interrupt enable bit

When OPERRIE1 bit is set to 1, an interrupt is generated when an error is detected during a write/erase operation to bank 1. OPERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a write/erase error occurs on bank 1

1: interrupt generated when a write/erase error occurs on bank 1

Bit 21 **INCERRIE1**: Bank 1 inconsistency error interrupt enable bit

When INCERRIE1 bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation to bank 1. INCERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a inconsistency error occurs on bank 1

1: interrupt generated when a inconsistency error occurs on bank 1.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **STRBERRIE1**: Bank 1 strobe error interrupt enable bit

When STRBERRIE1 bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation to bank 1. STRBERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a strobe error occurs on bank 1

1: interrupt generated when strobe error occurs on bank 1.

Bit 18 **PGSERRIE1**: Bank 1 programming sequence error interrupt enable bit

When PGSERRIE1 bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation to bank 1. PGSERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a sequence error occurs on bank 1

1: interrupt generated when sequence error occurs on bank 1.

Bit 17 **WRPERRIE1**: Bank 1 write protection error interrupt enable bit

When WRPERRIE1 bit is set to 1, an interrupt is generated when a protection error occurs during a program operation to bank 1. WRPERRIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated when a protection error occurs on bank 1

1: interrupt generated when a protection error occurs on bank 1.

Bit 16 **EOPIE1**: Bank 1 end-of-program interrupt control bit

Setting EOPIE1 bit to 1 enables the generation of an interrupt at the end of a program operation to bank 1. EOPIE1 can be programmed only when LOCK1 is cleared to 0.

0: no interrupt generated at the end of a program operation to bank 1.

1: interrupt enabled when at the end of a program operation to bank 1.

Bit 15 **CRC_EN**: Bank 1 CRC control bit

Setting CRC_EN bit to 1 enables the CRC calculation on bank 1. CRC_EN does not start CRC calculation but enables CRC configuration through FLASH_CRCCR1 register.

When CRC calculation is performed on bank 1, it can only be disabled by setting CRC_EN bit to 0. Resetting CRC_EN clears CRC configuration and resets the content of FLASH_CRCDATAR register.

Clearing CRC_EN to 0 sets CRCDATA to 0x0.

CRC_EN can be programmed only when LOCK1 is cleared to 0.

Bit 14 Reserved, must be kept at reset value.

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:8 **SNB1**: Bank 1 sector erase selection number

These bits are used to select the target sector for a sector erase operation (they are unused otherwise). SNB1 can be programmed only when LOCK1 is cleared to 0.

000: sector 0 of bank 1

001: sector 1 of bank 1

...

111: sector 7 of bank 1

Bit 7 **START1**: Bank 1 erase start control bit

START1 bit is used to start a sector erase or a bank erase operation. START1 can be programmed only when LOCK1 is cleared to 0.

The embedded Flash memory resets START1 when the corresponding operation has been acknowledged. The user application cannot access any embedded Flash memory register until the operation is acknowledged.

Bit 6 **FW1:** Bank 1 write forcing control bit

FW1 forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW1 can be programmed only when LOCK1 is cleared to 0.

The embedded Flash memory resets FW1 when the corresponding operation has been acknowledged. The user application cannot access any embedded Flash memory register until the operation is acknowledged.

Note: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it will lead to permanent ECC error.

Write forcing is effective only if the write buffer is not empty (in particular, FW1 does not start several write operations when the force-write operations are performed consecutively).

Bits 5:4 **PSIZE1**: Bank 1 program size

PSIZE1 selects the parallelism used by the non-volatile memory during write and erase operations to bank 1. PSIZE1 can be programmed only when LOCK1 is cleared to 0.

00: programming executed with byte parallelism

01: programming executed with half-word parallelism

10: programming executed with word parallelism

11: programming executed with double word parallelism

Bit 3 **BER1**: Bank 1 erase request

Setting BER1 bit to 1 requests a bank erase operation on bank 1 (user Flash memory only). BER1 can be programmed only when LOCK1 is cleared to 0.

BER1 has a higher priority than SER1: if both are set, the embedded Flash memory executes a bank erase.

0: bank erase not requested on bank 1

1: bank erase requested on bank 1

Note: Write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2 **SER1**: Bank 1 sector erase request

Setting SER1 bit to 1 requests a sector erase on bank 1. SER1 can be programmed only when LOCK1 is cleared to 0.

BER1 has a higher priority than SER1: if both bits are set, the embedded Flash memory executes a bank erase.

0: sector erase not requested on bank 1

1: sector erase requested on bank 1

Note: Write protection error is triggered when a sector erase is required on a protected sector.

Bit 1 **PG1**: Bank 1 internal buffer control bit

Setting PG1 bit to 1 enables internal buffer for write operations to bank 1. This allows preparing program operations even if a sector or bank erase is ongoing.

PG1 can be programmed only when LOCK1 is cleared to 0. When PG1 is reset, the internal buffer is disabled for write operations to bank 1, and all the data stored in the buffer but not sent to the operation queue are lost.

0: Internal buffer disabled for write operations to bank 1

1: Internal buffer enabled for write operations to bank 1

Bit 0 **LOCK1**: Bank 1 configuration lock bit

This bit locks the FLASH_CR1 register. The correct write sequence to FLASH_KEYR1 register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_KEYR1 is performed twice, this bit remains locked until the next system reset.

LOCK1 can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK1 changes from 0 to 1, the other bits of FLASH_CR1 register do not change.

0: FLASH_CR1 register unlocked

1: FLASH_CR1 register locked

3.9.5 FLASH status register for bank 1 (FLASH_SR1)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CRCEND1	DBECCERR1	SNECCERR1	RDSERR1	RDPERR1	OPERR1	INCERR1	Res.	STRBERR1	PGSERR1	WRPERR1	EOP1
				r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BUSY1	QW1	WBNE1	BSY1
												r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CRCEND1**: Bank 1 CRC end of calculation flag

CRCEND1 bit is raised when the CRC computation has completed on bank 1. An interrupt is generated if CRCENDIE1 is set to 1. It is not necessary to reset CRCEND1 before restarting CRC computation. Writing 1 to CLR_CRCEND1 bit in FLASH_CCR1 register clears CRCEND1.

0: CRC computation not complete on bank 1

1: CRC computation complete on bank 1

Bit 26 **DBECCERR1**: Bank 1 ECC double detection error flag

DBECCERR1 flag is raised when an ECC double detection error occurs during a read operation from bank 1. An interrupt is generated if DBECCERRIE1 is set to 1. Writing 1 to CLR_DBECCERR1 bit in FLASH_CCR1 register clears DBECCERR1.

0: no ECC double detection error occurred on bank 1

1: ECC double detection error occurred on bank 1

Bit 25 **SNECCERR1**: Bank 1 single correction error flag

SNECCERR1 flag is raised when an ECC single correction error occurs during a read operation from bank 1. An interrupt is generated if SNECCERRIE1 is set to 1. Writing 1 to CLR_SNECCERR1 bit in FLASH_CCR1 register clears SNECCERR1.

0: no ECC single correction error occurs on bank 1

1: ECC single correction error occurs on bank 1

Bit 24 **RDSERR1**: Bank 1 secure error flag

RDSERR1 flag is raised when a read secure error (read access to a secure-only protected word) occurs on bank 1. An interrupt is generated if RDSERRIE1 is set to 1. Writing 1 to CLR_RDSERR1 bit in FLASH_CCR1 register clears RDSERR1.

0: no secure error occurs on bank 1

1: a secure error occurs on bank 1

Bit 23 **RDPERR1: Bank 1** read protection error flag

RDPERR1 flag is raised when an read protection error (read access to a PCROP-protected or a RDP-protected area) occurs on bank 1. An interrupt is generated if RDPERRIE1 is set to 1. Writing 1 to CLR_RDPERR1 bit in FLASH_CCR1 register clears RDPERR1.

0: no read protection error occurs on bank 1

1: a read protection error occurs on bank 1

Bit 22 **OPERR1**: Bank 1 write/erase error flag

OPERR1 flag is raised when an error occurs during a write/erase to bank 1. An interrupt is generated if OPERRIE1 is set to 1. Writing 1 to CLR_OPERR1 bit in FLASH_CCR1 register clears OPERR1.

0: no write/erase error occurs on bank 1

1: a write/erase error occurs on bank 1

Bit 21 **INCERR1:** Bank 1 inconsistency error flag

INCERR1 flag is raised when a inconsistency error occurs on bank 1. An interrupt is generated if INCERRIE1 is set to 1. Writing 1 to CLR_INCERR1 bit in the FLASH_CCR1 register clears INCERR1.

0: no inconsistency error occurs on bank 1

1: a inconsistency error occurs on bank 1

Bit 20 Reserved, must be kept at reset value.

Bit 19 **STRBERR1:** Bank 1 strobe error flag

STRBERR1 flag is raised when a strobe error occurs on bank 1 (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE1 bit is set to 1. Writing 1 to CLR_STRBERR1 bit in FLASH_CCR1 register clears STRBERR1.

0: no strobe error occurs on bank 1

1: a strobe error occurs on bank 1

Bit 18 **PGSERR1:** Bank 1 programming sequence error flag

PGSERR1 flag is raised when a sequence error occurs on bank 1. An interrupt is generated if the PGSERRIE1 bit is set to 1. Writing 1 to CLR_PGSERR1 bit in FLASH_CCR1 register clears PGSERR1.

0: no sequence error occurs on bank 1

1: a sequence error occurs on bank 1

Bit 17 **WRPERR1**: Bank 1 write protection error flag

WRPERR1 flag is raised when a protection error occurs during a program operation to bank 1. An interrupt is also generated if the WRPERRIE1 is set to 1. Writing 1 to CLR_WRPERR1 bit in FLASH_CCR1 register clears WRPERR1.

0: no write protection error occurs on bank 1

1: a write protection error occurs on bank 1

Bit 16 **EOP1**: Bank 1 end-of-program flag

EOP1 flag is set when a programming operation to bank 1 completes. An interrupt is generated if the EOPIE1 is set to 1. It is not necessary to reset EOP1 before starting a new operation. EOP1 bit is cleared by writing 1 to CLR_EOP1 bit in FLASH_CCR1 register.

0: no programming operation completed on bank 1

1: a programming operation completed on bank 1

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CRC_BUSY1**: Bank 1 CRC busy flag

CRC_BUSY1 flag is set when a CRC calculation is ongoing on bank 1. This bit cannot be forced to 0. The user must wait until the CRC calculation has completed or disable CRC computation on bank 1.

0: no CRC calculation ongoing on bank 1

1: CRC calculation ongoing on bank 1

Bit 2 **QW1**: Bank 1 wait queue flag

QW1 flag is set when a write, erase or option byte change operation is pending in the command queue buffer of bank 1. It is not possible to know what type of programming operation is present in the queue.

This flag is reset by hardware when all write, erase or option byte change operations have been executed and thus removed from the waiting queue(s). This bit cannot be forced to 0. It is reset after a deterministic time if no other operations are requested.

0: no write, erase or option byte change operations waiting in the operation queues of bank 1

1: at least one write, erase or option byte change operation is waiting in the operation queue of bank 1

Bit 1 **WBNE1**: Bank 1 write buffer not empty flag

WBNE1 flag is set when the embedded Flash memory is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE1 is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:

- the application software forces the write operation using FW1 bit in FLASH_CR1
- the embedded Flash memory detects an error that involves data loss
- the application software has disabled write operations in this bank

This bit cannot be forced to 0. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer of bank 1 empty or full

1: write buffer of bank 1 waiting data to complete

Bit 0 **BSY1**: Bank 1 busy flag

BSY1 flag is set when an effective write, erase or option byte change operation is ongoing on bank 1. It is not possible to know what type of operation is being executed.

BSY1 cannot be forced to 0. It is automatically reset by hardware every time a step in a write, erase or option byte change operation completes.

0: no programming, erase or option byte change operation being executed on bank 1

1: programming, erase or option byte change operation being executed on bank 1

3.9.6 FLASH clear control register for bank 1 (FLASH_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CLR_CRCEND1	CLR_DBECCERR1	CLR_SNECCERR1	CLR_RDSERR1	CLR_RDPERR1	CLR_OPERR1	CLR_INCERR1	Res.	CLR_STRBERR1	CLR_PGSERR1	CLR_WRPERR1	CLR_EOP1
				w	w	w	w	w	w	w		w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CLR_CRCEND1**: Bank 1 CRCEND1 flag clear bit

Setting this bit to 1 resets to 0 CRCEND1 flag in FLASH_SR1 register.

Bit 26 **CLR_DBECCERR1**: Bank 1 DBECCERR1 flag clear bit

Setting this bit to 1 resets to 0 DBECCERR1 flag in FLASH_SR1 register. If the SNECCERR1 flag of FLASH_SR1 register is cleared to 0, FLASH_ECC_FA1R register is reset to 0 as well.

Bit 25 **CLR_SNECCERR1**: Bank 1 SNECCERR1 flag clear bit

Setting this bit to 1 resets to 0 SNECCERR1 flag in FLASH_SR1 register. If the DBECCERR1 flag of FLASH_SR1 register is cleared to 0, FLASH_ECC_FA1R register is reset to 0 as well.

Bit 24 **CLR_RDSERR1**: Bank 1 RDSERR1 flag clear bit

Setting this bit to 1 resets to 0 RDSERR1 flag in FLASH_SR1 register.

Bit 23 **CLR_RDPERR1**: Bank 1 RDPERR1 flag clear bit

Setting this bit to 1 resets to 0 RDPERR1 flag in FLASH_SR1 register.

Bit 22 **CLR_OPERR1**: Bank 1 OPERR1 flag clear bit

Setting this bit to 1 resets to 0 OPERR1 flag in FLASH_SR1 register.

Bit 21 **CLR_INCERR1**: Bank 1 INCERR1 flag clear bit

Setting this bit to 1 resets to 0 INCERR1 flag in FLASH_SR1 register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CLR_STRBERR1**: Bank 1 STRBERR1 flag clear bit

Setting this bit to 1 resets to 0 STRBERR1 flag in FLASH_SR1 register.

Bit 18 **CLR_PGSERR1**: Bank 1 PGSERR1 flag clear bit

Setting this bit to 1 resets to 0 PGSERR1 flag in FLASH_SR1 register.

Bit 17 **CLR_WRPERR1**: Bank 1 WRPERR1 flag clear bit
Setting this bit to 1 resets to 0 WRPERR1 flag in FLASH_SR1 register.

Bit 16 **CLR_EOP1**: Bank 1 EOP1 flag clear bit
Setting this bit to 1 resets to 0 EOP1 flag in FLASH_SR1 register.

Bits 15:0 Reserved, must be kept at reset value.

3.9.7 FLASH option control register (FLASH_OPTCR)

Address offset: 0x018 or 0x118

Reset value: 0xX000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	OPTCHANGEERRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MER	Res.	Res.	OPTSTART	OPTLOCK
											w			w	rs

Bit 31 **SWAP_BANK**: Bank swapping option configuration bit
SWAP_BANK controls whether bank 1 and bank 2 are swapped or not. This bit is loaded with the SWAP_BANK_OPT bit of FLASH_OPTSR_CUR register only after reset or POR.
0: bank 1 and bank 2 not swapped
1: bank 1 and bank 2 swapped

Note: The Flash bank swapping is not available on STM32H750xB devices. The SWAP_BANK option bit must be kept to '0'.

Bit 30 **OPTCHANGEERRIE**: Option byte change error interrupt enable bit
OPTCHANGEERRIE bit controls if an interrupt has to be generated when an error occurs during an option byte change.
0: no interrupt is generated when an error occurs during an option byte change
1: an interrupt is generated when and error occurs during an option byte change.

Bits 29:5 Reserved, must be kept at reset value.

Bit 4 **MER**: mass erase request
Setting this bit launches a non-volatile memory bank erase on both banks (i.e. mass erase). Programming MER bit to 1 automatically sets BER1, BER2, START1 and START2 to 1. FLASH_OPTCR, FLASH_CR1 and FLASH_CR2 must be unlocked prior to setting MER high.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **OPTSTART**: Option byte start change option configuration bit

OPTSTART triggers an option byte change operation. The user can set OPTSTART only when the OPTLOCK bit is cleared to 0. The embedded Flash memory resets OPTSTART when the option byte change operation has been acknowledged.

The user application cannot modify any embedded Flash memory register until the option change operation has been completed.

Before setting this bit, the user has to write the required values in the FLASH_XXX_PRG registers. The FLASH_XXX_PRG registers will be locked until the option byte change operation has been executed in non-volatile memory.

It is not possible to start an option byte change operation if a CRC calculation is ongoing on bank 1 or bank 2. Trying to set OPTSTART when CRC_BUSY1/2 of FLASH_SR1/2 register is set has not effect; the option byte change does not start and no error is generated.

Bit 0 **OPTLOCK**: FLASH_OPTCR lock option configuration bit

The OPTLOCK bit locks the FLASH_OPTCR register as well as all _PRG registers. The correct write sequence to FLASH_OPTKEYR register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_OPTKEYR is performed twice, this bit remains locked until next system reset.

It is possible to set OPTLOCK by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When OPTLOCK changes from 0 to 1, the others bits of FLASH_OPTCR register do not change.

0: FLASH_OPTCR register unlocked

1: FLASH_OPTCR register locked.

3.9.8 FLASH option status register (FLASH_OPTSR_CUR)

Address offset: 0x01C or 0x11C

Reset value: 0xXXXX XXXX (see *Table 16: Option byte organization*)

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK_OPT	OPTCHANGEERR	IO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECURITY	ST_RAM_SIZE[1:0]		IWDG_FZ_SDBY	IWDG_FZ_STOP	Res.
r	r	r								r	r	r	r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDP								NRST_STDY_D1	NRST_STOP_D1	Res.	IWDG1_SW	BOR_LEV		Res.	OPT_BUSY
r	r	r	r	r	r	r	r	r	r		r	r	r		r

Bit 31 **SWAP_BANK_OPT**: Bank swapping option status bit

SWAP_BANK_OPT reflects whether bank 1 and bank 2 are swapped or not. SWAP_BANK_OPT is loaded to SWAP_BANK of FLASH_OPTCR after a reset.

0: bank 1 and bank 2 not swapped

1: bank 1 and bank 2 swapped

Note: The Flash bank swapping is not available on STM32H750xB devices. The SWAP_BANK option bit must be kept to '0'.

Bit 30 **OPTCHANGEERR**: Option byte change error flag

OPTCHANGEERR flag indicates that an error occurred during an option byte change operation. When OPTCHANGEERR is set to 1, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTCHANGEERRIE bit of FLASH_OPTCR register is set to 1.

Writing 1 to CLR_OPTCHANGEERR of register FLASH_OPTCCR clears OPTCHANGEERR.

0: no option byte change errors occurred

1: one or more errors occurred during an option byte change operation.

Bit 29 **IO_HSLV**: I/O high-speed at low-voltage status bit

This bit indicates that the product operates below 2.5 V.

0: Product working in the full voltage range, I/O speed optimization at low-voltage disabled

1: Product operating below 2.5 V, I/O speed optimization at low-voltage feature allowed

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:24 Reserved, must be kept at reset value.

Bits 23: 22 Reserved, must be kept at reset value.

Bit 21 **SECURITY**: Security enable option status bit

0: Security feature disabled

1: Security feature enabled.

Bits 20:19 **ST_RAM_SIZE[1:0]**: ST RAM size option status

00: 2 Kbytes reserved to ST code

01: 4 Kbytes reserved to ST code

10: 8 Kbytes reserved to ST code

11: 16 Kbytes reserved to ST code

Note: This bitfield is effective only when the security is enabled (SECURITY = 1).

Bit 18 **IWDG_FZ_SDBY**: IWDG Standby mode freeze option status bit

When set the independent watchdog IWDG1 is frozen in system Standby mode.

0: Independent watchdog frozen in Standby mode

1: Independent watchdog keep running in Standby mode.

Bit 17 **IWDG_FZ_STOP**: IWDG Stop mode freeze option status bit

When set the independent watchdog IWDG1 is in system Stop mode.

0: Independent watchdog frozen in system Stop mode

1: Independent watchdog keep running in system Stop mode.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RDP**: Readout protection level option status byte
0xAA: global readout protection level 0
0xCC: global readout protection level 2
others values: global readout protection level 1.

Bit 7 **NRST_STDY_D1**: D1 domain DStandby entry reset option status bit
0: a reset is generated when entering DStandby mode on D1 domain
1: no reset generated when entering DStandby mode on D1 domain

Bit 6 **NRST_STOP_D1**: D1 domain DStop entry reset option status bit
0: a reset is generated when entering DStop mode on D1 domain
1: no reset generated when entering DStop mode on D1 domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IWDG1_SW**: IWDG1 control mode option status bit
0: IWDG1 watchdog is controlled by hardware
1: IWDG1 watchdog is controlled by software

Bits 3:2 **BOR_LEV**: Brownout level option status bit
These bits reflects the power level that generates a system reset.
00: V_{BOR0}, brownout reset threshold 0
01: V_{BOR1}, brownout reset threshold 1
02: V_{BOR2}, brownout reset threshold 2
03: V_{BOR3}, brownout reset threshold 3
Note: Refer to device datasheet for the values of V_{BORx} V_{DD} *reset thresholds.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OPT_BUSY**: Option byte change ongoing flag
OPT_BUSY indicates if an option byte change is ongoing. When this bit is set to 1, the embedded Flash memory is performing an option change and it is not possible to modify any embedded Flash memory register.
0: no option byte change ongoing
1: an option byte change ongoing and all write accesses to Flash registers are blocked until the option byte change completes.

3.9.9 FLASH option status register (FLASH_OPTSR_PRG)

Address offset: 0x020 or 0x120

Reset value: 0xXXXX XXXX (see *Table 16: Option byte organization*)

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK_OPT	Res.	IO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECURITY	ST_RAM_SIZE[1:0]		IWDG_FZ_SDBY	IWDG_FZ_STOP	Res.
rw		rw								rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDP								NRST_STDY_D1	NRST_STOP_D1	Res.	IWDG1_SW	BOR_LEV		Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		

Bit 31 **SWAP_BANK_OPT**: Bank swapping option configuration bit

SWAP_BANK_OPT option bit is used to configure whether the bank 1 and bank 2 are swapped or not. This bit is loaded with the SWAP_BANK_OPT bit of FLASH_OPTSR_CUR register after a reset.

0: bank 1 and bank 2 not swapped

1: bank 1 and bank 2 swapped

Bit 30 Reserved, must be kept at reset value.

Bit 29 **IO_HSLV**: I/O high-speed at low-voltage configuration bit

This bit indicates that the product operates below 2.5 V.

0: Product working in the full voltage range, I/O speed optimization at low-voltage disabled

1: Product operating below 2.5 V, I/O speed optimization at low-voltage feature allowed

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:24 Reserved, must be kept at reset value.

Bits 23: 22 Reserved, must be kept at reset value.

Bit 21 **SECURITY**: Security enable option configuration bit

The SECURITY option bit enables the secure access mode of the device during an option byte change. The change will be taken into account at next reset (next boot sequence). Once it is enabled, the security feature can be disabled if no areas are protected by PCROP or Secure access mode. If there are secure-only or PCROP protected areas, perform a level regression (from level 1 to 0) and set all the bits to unprotect secure-only areas and PCROP areas.

0: Security feature disabled

1: Security feature enabled.

Bits 20:19 **ST_RAM_SIZE[1:0]**: ST RAM size option configuration bits
00: 2 Kbytes reserved to ST code
01: 4 Kbytes reserved to ST code
10: 8 Kbytes reserved to ST code
11: 16 Kbytes reserved to ST code
Note: This bitfield is effective only when the security is enabled (SECURITY = 1).

Bit 18 **IWDG_FZ_SDBY**: IWDG Standby mode freeze option configuration bit
This option bit is used to freeze or not the independent watchdog IWDG1 in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode.

Bit 17 **IWDG_FZ_STOP**: IWDG Stop mode freeze option configuration bit
This option bit is used to freeze or not the independent watchdog IWDG1 in system Stop mode.
0: Independent watchdog frozen in system Stop mode
1: Independent watchdog keep running in system Stop mode.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RDP**: Readout protection level option configuration bits
RDP bits are used to change the readout protection level. This change is possible only when the current protection level is different from level 2. The possible configurations are:
0xAA: global readout protection level 0
0xCC: global readout protection level 2
others values: global readout protection level 1.

Bit 7 **NRST_STDY_D1**: D1 domain DStandby entry reset option configuration bit
0: a reset is generated when entering DStandby mode on D1 domain.
1: no reset generated when entering DStandby mode on D1 domain

Bit 6 **NRST_STOP_D1**: D1 domain DStop entry reset option configuration bit
0: a reset is generated when entering DStop mode on D1 domain.
1: no reset generated when entering DStop mode on D1 domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IWDG1_SW**: IWDG1 control mode option configuration bit
IWDG1_SW option bit is used to select if IWDG1 independent watchdog is controlled by hardware or by software.
0: IWDG1 watchdog is controlled by hardware
1: IWDG1 watchdog is controlled by software

Bits 3:2 **BOR_LEV**: Brownout level option configuration bit
These option bits are used to define the power level that generates a system reset.
00: V_{BOR0}, brownout reset threshold 0
01: V_{BOR1}, brownout reset threshold 1
02: V_{BOR2}, brownout reset threshold 2
03: V_{BOR3}, brownout reset threshold 3
Note: Refer to device datasheet for the values of V_{BORx} V_{DD} *reset thresholds.*

Bits 1:0 Reserved, must be kept at reset value.

3.9.10 FLASH option clear control register (FLASH_OPTCCR)

Address offset: 0x024 or 0x124

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CLR_OPTCHANGEERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CLR_OPTCHANGEERR**: OPTCHANGEERR reset bit
This bit is used to reset the OPTCHANGEERR flag in FLASH_OPTSR_CUR register.
FLASH_OPTCCR is write-only.
It is reset by programming it to 1.

Bits 29:0 Reserved, must be kept at reset value.

3.9.11 FLASH protection address for bank 1 (FLASH_PRAR_CUR1)

Address offset: 0x028

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMEP1	Res.	Res.	Res.	PROT_AREA_END1											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PROT_AREA_START1											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **DMEP1**: Bank 1 PCROP protected erase enable option status bit
If DMEP1 is set to 1, the PCROP protected area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **PROT_AREA_END1**: Bank 1 PCROP area end status bits
These bits contain the last 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_START1, the whole bank 1 is PCROP protected.
If this address is lower than PROT_AREA_START1, no protection is set on bank 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PROT_AREA_START1**: Bank 1 PCROP area start status bits
These bits contain the first 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_END1, the whole bank 1 is PCROP protected.
If this address is higher than PROT_AREA_END1, no protection is set on bank 1.

3.9.12 FLASH protection address for bank 1 (FLASH_PRAR_PRG1)

Address offset: 0x02C

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMEP1	Res.	Res.	Res.	PROT_AREA_END1											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PROT_AREA_START1											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DMEP1**: Bank 1 PCROP protected erase enable option configuration bit
If DMEP1 is set to 1, the PCROP protected area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **PROT_AREA_END1**: Bank 1 PCROP area end configuration bits
These bits contain the last 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_START1, the whole bank 1 is PCROP protected.
If this address is lower than PROT_AREA_START1, no protection is set on bank 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PROT_AREA_START1**: Bank 1 PCROP area start configuration bits
These bits contain the first 256-byte block of the PCROP area in bank 1.
If this address is equal to PROT_AREA_END1, the whole bank 1 is PCROP protected.
If this address is higher than PROT_AREA_END1, no protection is set on bank 1.

3.9.13 FLASH secure address for bank 1 (FLASH_SCAR_CUR1)

Address offset: 0x030

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMES1	Res.	Res.	Res.	SEC_AREA_END1											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SEC_AREA_START1											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **DMES1**: Bank 1 secure access protected erase enable option status bit
If DMES1 is set to 1, the secure access only area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SEC_AREA_END1**: Bank 1 secure-only area end status bits
These bits contain the last 256-byte block of the secure-only area in bank 1.
If this address is equal to SEC_AREA_START1, the whole bank 1 is secure access only.
If this address is lower than SEC_AREA_START1, no protection is set on bank 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SEC_AREA_START1**: Bank 1 secure-only area start status bits
These bits contain the first 256 bytes of block of the secure-only area in bank 1.
If this address is equal to SEC_AREA_END1, the whole bank 1 is secure access only.
If this address is higher than SEC_AREA_END1, no protection is set on bank 1.

3.9.14 FLASH secure address for bank 1 (FLASH_SCAR_PRG1)

Address offset: 0x034

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMES1	Res.	Res.	Res.	SEC_AREA_END1											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SEC_AREA_START1											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DMES1**: Bank 1 secure access protected erase enable option configuration bit
If DMES1 is set to 1, the secure access only area in bank 1 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SEC_AREA_END1**: Bank 1 secure-only area end configuration bits
These bits contain the last block of 256 bytes of the secure-only area in bank 1.
If this address is equal to SEC_AREA_START1, the whole bank 1 is secure access only.
If this address is lower than SEC_AREA_START1, no protection is set on bank 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SEC_AREA_START1**: Bank 1 secure-only area start configuration bits
These bits contain the first block of 256 bytes of the secure-only area in bank 1.
If this address is equal to SEC_AREA_END1, the whole bank 1 is secure access only.
If this address is higher than SEC_AREA_END1, no protection is set on bank 1.

3.9.15 FLASH write sector protection for bank 1 (FLASH_WPSN_CUR1R)

Address offset: 0x038

Reset value: 0x0000 00XX

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn1							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSn1**: Bank 1 sector write protection option status byte
Each FLASH_WPSGN_CUR1R bit reflects the write protection status of the corresponding bank 1 sector (0: sector is write protected; 1: sector is not write protected)

3.9.16 FLASH write sector protection for bank 1 (FLASH_WPSN_PRG1R)

Address offset: 0x03C

Reset value: 0x0000 00XX

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn1							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSn1**: Bank 1 sector write protection option status byte

Setting each FLASH_WPSGN_PRG1R bit to 0 write-protects the corresponding bank 1 sector (0: sector is write protected; 1: sector is not write protected)

3.9.17 FLASH register boot address FLASH_BOOT_CURR)

Address offset: 0x040 or 0x140

Reset value: 0xXXXX XXXX (see *Table 16: Option byte organization*)

This register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOOT_ADD1															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT_ADD0															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **BOOT_ADD1**: Arm® Cortex®-M7 boot address 1

These bits reflect the MSB of the Arm® Cortex®-M7 boot address when the BOOT pin is high.

Bits 15:0 **BOOT_ADD0**: Arm® Cortex®-M7 boot address 0

These bits reflect the MSB of the Arm® Cortex®-M7 boot address when the BOOT pin is low.

3.9.18 FLASH register boot address FLASH_BOOT_PRGR)

Address offset: 0x044 or 0x144

Reset value: 0xXXXX XXXX (see *Table 16: Option byte organization*)

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOOT_ADD1															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT_ADD0															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **BOOT_ADD1**: Arm® Cortex®-M7 boot address 1 configuration
These bits allow configuring the MSB of the Arm® Cortex®-M7 boot address when the BOOT pin is high.

Bits 15:0 **BOOT_ADD0**: Arm® Cortex®-M7 boot address 0 configuration
These bits allow configuring the MSB of the Arm® Cortex®-M7 boot address when the BOOT pin is low.

3.9.19 FLASH CRC control register for bank 1 (FLASH_CRCCR1)

Address offset: 0x050

Reset value: 0x001C 0000

This register can be modified only if CRC_EN bit is set to 1 in FLASH_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BURST		Res.	Res.	CLEAN_CRC	START_CRC
										rw	rw			w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CLEAN_SECT	ADD_SECT	CRC_BY_SECT	ALL_BANK	Res.	Res.	Res.	Res.	CRC_SECT		
					w	w	rw	w					rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **CRC_BURST**: Bank 1 CRC burst size
CRC_BURST bits set the size of the bursts that are generated by the CRC calculation unit.
00: every burst has a size of 4 Flash words (256-bit)
01: every burst has a size of 16 Flash words (256-bit)
10: every burst has a size of 64 Flash words (256-bit)
11: every burst has a size of 256 Flash words (256-bit)

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CLEAN_CRC**: Bank 1 CRC clear bit
Setting CLEAN_CRC to 1 clears the current CRC result stored in the FLASH_CRCDATAR register.

Bit 16 **START_CRC**: Bank 1 CRC start bit

START_CRC bit triggers a CRC calculation on bank 1 using the current configuration. No CRC calculation can launched when an option byte change operation is ongoing because all write accesses to embedded Flash memory registers are put on hold until the option byte change operation has completed.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CLEAN_SECT**: Bank 1 CRC sector list clear bit

Setting CLEAN_SECT to 1 clears the list of sectors on which the CRC is calculated.

Bit 9 **ADD_SECT**: Bank 1 CRC sector select bit

Setting ADD_SECT to 1 adds the sector whose number is CRC_SECT to the list of sectors on which the CRC is calculated.

Bit 8 **CRC_BY_SECT**: Bank 1 CRC sector mode select bit

When CRC_BY_SECT is set to 1, the CRC calculation is performed at sector level, on the sectors present in the list of sectors. To add a sector to this list, use ADD_SECT and CRC_SECT bits. To clean the list, use CLEAN_SECT bit.

When CRC_BY_SECT is reset to 0, the CRC calculation is performed on all addresses between CRC_START_ADDR and CRC_END_ADDR.

Bit 7 **ALL_BANK**: Bank 1 CRC select bit

When ALL_BANK is set to 1, all bank 1 user sectors are added to list of sectors on which the CRC is calculated.

Bits 6:3 Reserved, must be kept at reset value.

Bits 2:0 **CRC_SECT**: Bank 1 CRC sector number

CRC_SECT is used to select one or more user Flash sectors to be added to the list of sectors on which the CRC is calculated. The CRC can be computed either between two addresses (using registers FLASH_CRCSADD1R and FLASH_CRCEADD1R) or on a list of sectors. If this latter option is selected, it is possible to add a sector to the list of sectors by programming the sector number in CRC_SECT and then setting ADD_SECT to 1.

The list of sectors can be erased either by setting CLEAN_SECT bit or by disabling the CRC computation. CRC_SECT can be set only when CRC_EN of FLASH_CR register is set to 1.

000: sector 0 of bank 1

001: sector 1 of bank 1

...

111: sector 7 of bank 1

3.9.20 FLASH CRC start address register for bank 1 (FLASH_CRCSADD1R)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_START_ADDR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_START_ADDR[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:2 **CRC_START_ADDR[19:2]**: CRC start address on bank 1

CRC_START_ADDR is used when CRC_BY_SECT is cleared to 0. It must be programmed to the start address of the bank 1 memory area on which the CRC calculation is performed.

Bits 1:0 Reserved, must be kept at reset value.

3.9.21 FLASH CRC end address register for bank 1 (FLASH_CRCEADD1R)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_END_ADDR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_END_ADDR[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:2 **CRC_END_ADDR[19:2]**: CRC end address on bank 1

CRC_END_ADDR is used when CRC_BY_SECT is cleared to 0. It must be programmed to the end address of the bank 1 memory area on which the CRC calculation is performed

Bits 1:0 Reserved, must be kept at reset value.

3.9.22 FLASH CRC data register (FLASH_CRCDATAR)

Address offset: 0x05C or 0x15C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_DATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_DATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CRC_DATA**: CRC result

CRC_DATA bits contain the result of the last CRC calculation.

3.9.23 FLASH ECC fail address for bank 1 (FLASH_ECC_FA1R)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FAIL_ECC_ADDR1														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **FAIL_ECC_ADDR1**: Bank 1 ECC error address

When an ECC error occurs (both for single correction or double detection) during a read operation from bank 1, the FAIL_ECC_ADDR1 bitfield contains the address that generated the error.

FAIL_ECC_ADDR1 is reset when the flag error in the FLASH_SR1 register (CLR_SNECCERR1 or CLR_DBECCERR1) is reset.

The embedded Flash memory programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.

The address in FAIL_ECC_ADDR1 is relative to the Flash memory area where the error occurred (user Flash memory, system Flash memory).

3.9.24 FLASH key register for bank 2 (FLASH_KEYR2)

Address offset: 0x104

Reset value: 0x0000 0000

FLASH_KEYR2 is a write-only register. The following values must be programmed consecutively to unlock FLASH_CR2 register and allow programming/erasing it:

1. 1st key = 0x4567 0123
2. 2nd key = 0xCDEF 89AB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY2R															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY2R															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY2R**: Bank 2 access configuration unlock key

3.9.25 FLASH control register for bank 2 (FLASH_CR2)

Address offset: 0x10C

Reset value: 0x0000 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CRCENDIE2	DBECCERRIE2	SNECCERRIE2	RDSERRIE2	RDPERRIE2	OPERRIE2	INCERRIE2	Res.	STRBERRIE2	PGSERRIE2	WRPERRIE2	EOPIE2
				rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_EN	SPSS2	Res.	Res.	Res.	SNB2			START2	FW2	PSIZE2		BER2	SER2	PG2	LOCK2
rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CRCENDIE2**: Bank 2 CRC end of calculation interrupt enable bit

When CRCENDIE2 bit is set to 1, an interrupt is generated when the CRC computation has completed on bank 2. CRCENDIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when CRC computation complete on bank 2

1: interrupt generated when CRC computation complete on bank 2

Bit 26 **DBECCERRIE2**: Bank 2 ECC double detection error interrupt enable bit

When DBECCERRIE2 bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a read operation from bank 2. DBECCERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when an ECC double detection error occurs on bank 2

1: interrupt generated if an ECC double detection error occurs on bank 2

Bit 25 **SNECCERRIE2**: Bank 2 ECC single correction error interrupt enable bit

When SNECCERRIE2 bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation from bank 2. SNECCERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when an ECC single correction error occurs on bank 2

1: interrupt generated when an ECC single correction error occurs on bank 2

Bit 24 **RDSERRIE2**: Bank 2 secure error interrupt enable bit

When RDSERRIE2 bit is set to 1, an interrupt is generated when a secure error (access to a secure-only protected address) occurs during a read operation from bank 2. RDSERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a secure error occurs on bank 2

1: an interrupt is generated when a secure error occurs on bank 2

Bit 23 **RDPERRIE2**: Bank 2 read protection error interrupt enable bit

When RDPERRIE2 bit is set to 1, an interrupt is generated when a read protection error occurs (access to an address protected by PCROP or by RDP level 1) during a read operation from bank 2. RDPERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a read protection error occurs on bank 2

1: an interrupt is generated when a read protection error occurs on bank 2

Bit 22 **OPERRIE2**: Bank 2 write/erase error interrupt enable bit

When OPERRIE2 bit is set to 1, an interrupt is generated when an error is detected during a write/erase operation to bank 2. OPERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a write/erase error occurs on bank 2

1: interrupt generated when a write/erase error occurs on bank 2

Bit 21 **INCERRIE2**: Bank 2 inconsistency error interrupt enable bit

When INCERRIE2 bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation to bank 2. INCERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a inconsistency error occurs on bank 2

1: interrupt generated when a inconsistency error occurs on bank 2.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **STRBERRIE2**: Bank 2 strobe error interrupt enable bit

When STRBERRIE2 bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation to bank 2. STRBERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a strobe error occurs on bank 2

1: interrupt generated when strobe error occurs on bank 2.

Bit 18 **PGSERRIE2**: Bank 2 programming sequence error interrupt enable bit

When PGSERRIE2 bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation to bank 2. PGSERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a sequence error occurs on bank 2

1: interrupt generated when sequence error occurs on bank 2.

Bit 17 **WRPERRIE2**: Bank 2 write protection error interrupt enable bit

When WRPERRIE2 bit is set to 1, an interrupt is generated when a protection error occurs during a program operation to bank 2. WRPERRIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated when a protection error occurs on bank 2

1: interrupt generated when a protection error occurs on bank 2.

Bit 16 **EOPIE2**: Bank 2 end-of-program interrupt control bit

Setting EOPIE2 bit to 1 enables the generation of an interrupt at the end of a program operation to bank 2. EOPIE2 can be programmed only when LOCK2 is cleared to 0.

0: no interrupt generated at the end of a program operation to bank 2.

1: interrupt enabled when at the end of a program operation to bank 2.

Bit 15 **CRC_EN**: Bank 2 CRC control bit

Setting CRC_EN bit to 1 enables the CRC calculation on bank 2. CRC_EN does not start CRC calculation but enables CRC configuration through FLASH_CRCCR2 register.

When CRC calculation is performed on bank 2, it can only be disabled by setting CRC_EN bit to 0. Resetting CRC_EN clears CRC configuration and resets the content of FLASH_CRCDATAR register.

CRC_EN can be programmed only when LOCK2 is cleared to 0.

Bit 14 **SPSS2**: Bank 2 special sector selection bit

Set this bit when accessing non-user sectors of the Flash memory. This bit can be programmed only when LOCK2 is cleared to 0.

0: User sectors selection in bank 2

1: Non-user, special sectors selection in bank 2

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:8 **SNB2**: Bank 2 sector erase selection number

These bits are used to select the target sector for a sector erase operation (they are unused otherwise). SNB2 can be programmed only when LOCK2 is cleared to 0.

000: sector 0 of bank 2

001: sector 1 of bank 2

...

111: sector 7 of bank 2

Bit 7 **START2**: Bank 2 erase start control bit

START2 bit is used to start a sector erase or a bank erase operation. START2 can be programmed only when LOCK2 is cleared to 0.

The embedded Flash memory resets START2 when the corresponding operation has been acknowledged. The user application cannot access any embedded Flash memory register until the operation is acknowledged.

Bit 6 **FW2**: Bank 2 write forcing control bit

FW2 forces a write operation even if the write buffer is not full. FW2 can be programmed only when LOCK2 is cleared to 0.

The embedded Flash memory resets FW2 when the corresponding operation has been acknowledged. The user application cannot access any Flash register until the operation is acknowledged.

Write forcing is effective only if the write buffer is not empty. In particular, FW2 does not start several write operations when the write operations are performed consecutively.

Bits 5:4 **PSIZE2**: Bank 2 program size

PSIZE2 selects the maximum number of bits that can be written to 0 in one shot during a write operation (programming parallelism). PSIZE2 can be programmed only when LOCK2 is cleared to 0.

00: programming parallelism is byte (8 bits)

01: programming parallelism is half-word (16 bits)

10: programming parallelism is word (32 bits)

11: programming parallelism is double-word (64 bits)

Bit 3 **BER2**: Bank 2 erase request

Setting BER2 bit to 1 requests a bank erase operation on bank 2 (user Flash memory only). BER2 can be programmed only when LOCK2 is cleared to 0.

BER2 has a higher priority than SER2: if both are set, the embedded Flash memory executes a bank erase.

0: bank erase not requested on bank 2

1: bank erase requested on bank 2

Note: Write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2 **SER2**: Bank 2 sector erase request

Setting SER2 bit to 1 requests a sector erase on bank 2. SER2 can be programmed only when LOCK2 is cleared to 0.

BER2 has a higher priority than SER2: if both are set, the embedded Flash memory executes a bank erase.

0: sector erase not requested on bank 2

1: sector erase requested on bank 2

Note: Write protection error is triggered when a sector erase is required on protected sector(s).

Bit 1 **PG2**: Bank 2 internal buffer control bit

Setting PG2 bit to 1 enables internal buffer for write operations to bank 2. This allows the preparation of program operations even if a sector or bank erase is ongoing.

PG2 can be programmed only when LOCK2 is cleared to 0. When PG2 is reset, the internal buffer is disabled for write operations to bank 2 and all the data stored in the buffer but not sent to the operation queue are lost.

Bit 0 **LOCK2**: Bank 2 configuration lock bit

This bit locks the FLASH_CR2 register. The correct write sequence to FLASH_KEYR2 register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_KEYR2 is performed twice, this bit remains locked until next system reset.

LOCK2 can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK2 changes from 0 to 1, the other bits of FLASH_CR2 register do not change.

0: FLASH_CR2 register unlocked

1: FLASH_CR2 register locked

3.9.26 FLASH status register for bank 2 (FLASH_SR2)

Address offset: 0x110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CRCEND2	DBECCERR2	SNECCERR2	RDSERR2	RDPERR2	OPERR2	INCERR2	Res.	STRBERR2	PGSERR2	WRPERR2	EOP2
				r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BUSY2	QW2	WBNE2	BSY2
												r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CRCEND2**: Bank 2 CRC end of calculation flag

CRCEND2 bit is raised when the CRC computation has completed on bank 2. An interrupt is generated if CRCENDIE2 is set to 1. It is not necessary to reset CRCEND2 before restarting CRC computation. Writing 1 to CLR_CRCEND2 bit in FLASH_CCR2 register clears CRCEND2.

0: CRC computation not complete on bank 2

1: CRC computation complete on bank 2

Bit 26 **DBECCERR2**: Bank 2 ECC double detection error flag

DBECCERR2 flag is raised when an ECC double detection error occurs during a read operation from bank 2. An interrupt is generated if DBECCERRIE2 is set to 1. Writing 1 to CLR_DBECCERR2 bit in FLASH_CCR2 register clears DBECCERR2.

0: no ECC double detection error occurs on bank 2

1: ECC double detection error occurs on bank 2

Bit 25 **SNECCERR2**: Bank 2 single correction error flag

SNECCERR2 flag is raised when an ECC single correction error occurs during a read operation from bank 2. An interrupt is generated if SNECCERRIE2 is set to 1. Writing 1 to CLR_SNECCERR2 bit in FLASH_CCR2 register clears SNECCERR2.

0: no ECC single correction error occurs on bank 2

1: ECC single correction error occurs on bank 2

Bit 24 **RDSERR2**: Bank 2 secure error flag

RDSERR2 flag is raised when a read secure error (read access to a secure-only protected word) occurs on bank 2. An interrupt is generated if RDSERRIE2 is set to 1. Writing 1 to CLR_RDSERR2 bit in FLASH_CCR2 register clears RDSERR2.

0: no secure error occurs on bank 2

1: a secure error occurs on bank 2

Bit 23 **RDPERR2**: Bank 2 read protection error flag

RDPERR2 flag is raised when a read protection error (read access to a PCROP-protected word or a RDP-protected area) occurs on bank 2. An interrupt is generated if RDPERRIE2 is set to 1. Writing 1 to CLR_RDPERR2 bit in FLASH_CCR2 register clears RDPERR2.

0: no read protection error occurs on bank 2

1: a read protection error occurs on bank 2

Bit 22 **OPERR2**: Bank 2 write/erase error flag

OPERR2 flag is raised when an error occurs during a write/erase to bank 2. An interrupt is generated if OPERRIE2 is 1 set to 1. Writing 1 to CLR_OPERR2 bit in FLASH_CCR2 register clears OPERR2.

0: no write/erase error occurred on bank 2

1: a write/erase error occurred on bank 2

Bit 21 **INCERR2**: Bank 2 inconsistency error flag

INCERR2 flag is raised when a inconsistency error occurs on bank 2. An interrupt is generated if INCERRIE2 is set to 1. Writing 1 to CLR_INCERR2 bit in the FLASH_CCR2 register clears INCERR2.

0: no inconsistency error occurred on bank 2

1: an inconsistency error occurred on bank 2.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **STRBERR2**: Bank 2 strobe error flag

STRBERR2 flag is raised when a strobe error occurs on bank 2 (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE2 bit is set to 1. Writing 1 to CLR_STRBERR2 bit in FLASH_CCR2 register clears STRBERR2.

0: no strobe error occurred on bank 2

1: a strobe error occurred on bank 2.

Bit 18 **PGSERR2**: Bank 2 programming sequence error flag

PGSERR2 flag is raised when a sequence error occurs on bank 2. An interrupt is generated if the PGSERRIE2 bit is set to 1. Writing 1 to CLR_PGSERR2 bit in FLASH_CCR2 register clears PGSERR2.

0: no sequence error occurred on bank 2

1: a sequence error occurred on bank 2.

Bit 17 **WRPERR2**: Bank 2 write protection error flag

WRPERR2 flag is raised when a protection error occurs during a program operation to bank 2. An interrupt is also generated if the WRPERRIE2 is set to 1. Writing 1 to CLR_WRPERR2 bit in FLASH_CCR2 register clears WRPERR2.

0: no write protection error occurred on bank 2

1: a write protection error occurred on bank 2

Bit 16 **EOP2**: Bank 2 end-of-program flag

EOP2 flag is set when a programming operation to bank 2 completes. An interrupt is generated if the EOPIE2 is set to 1. It is not necessary to reset EOP2 before starting a new operation. EOP2 bit is cleared by writing 1 to CLR_EOP2 bit in FLASH_CCR2 register.

0: no programming operation completed on bank 2

1: a programming operation completed on bank 2

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CRC_BUSY2**: Bank 2 CRC busy flag

CRC_BUSY2 flag is set when a CRC calculation is ongoing on bank 2. This bit cannot be forced to 0. The user must wait until the CRC calculation has completed or disable CRC computation on bank 2.

0: no CRC calculation ongoing on bank 2

1: CRC calculation ongoing on bank 2.

Bit 2 **QW2**: Bank 2 wait queue flag

QW2 flag is set when a write or erase operation is pending in the command queue buffer of bank 2. It is not possible to know what type of operation is present in the queue. This flag is reset by hardware when all write/erase operations have been executed and thus removed from the waiting queue(s). This bit cannot be forced to 0. It is reset after a deterministic time if no other operations are requested.

0: no write or erase operation is waiting in the operation queues of bank 2

1: at least one write or erase operation is pending in the operation queues of bank 2

Bit 1 **WBNE2**: Bank 2 write buffer not empty flag

WBNE2 flag is set when embedded Flash memory is waiting for new data to complete the write buffer. In this state the write buffer is not empty. WBNE2 is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:

- the application software forces the write operation using FW2 bit in FLASH_CR2
- the embedded Flash memory detects an error that involves data loss
- the application software has disabled write operations in this bank

This bit cannot be forced to 0. To reset it, clear the write buffer by performing any of the above listed actions or send the missing data.

0: write buffer of bank 2 empty or full

1: write buffer of bank 2 waiting data to complete

Bit 0 **BSY2**: Bank 2 busy flag

BSY2 flag is set when an effective write or erase operation is ongoing to bank 2. It is not possible to know what type of operation is being executed.

BSY2 cannot be forced to 0. It is automatically reset by hardware every time a step in a write, or erase operation completes.

0: no write or erase operation is executed on bank 2

1: a write or an erase operation is being executed on bank 2.

3.9.27 FLASH clear control register for bank 2 (FLASH_CCR2)

Address offset: 0x114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CLR_CRCEND2	CLR_DBECCERR2	CLR_SNECCERR2	CLR_RDSERR2	CLR_RDPERR2	CLR_OPERR2	CLR_INCERR2	Res.	CLR_STRBERR2	CLR_PGSERR2	CLR_WRPERR2	CLR_EOP2
				w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CLR_CRCEND2**: Bank 2 CRCEND2 flag clear bit
Setting this bit to 1 resets to 0 CRCEND2 flag in FLASH_SR2 register.

Bit 26 **CLR_DBECCERR2**: Bank 2 DBECCERR2 flag clear bit
Setting this bit to 1 resets to 0 DBECCERR2 flag in FLASH_SR2 register. If the SNECCERR2 flag of FLASH_SR2 register is cleared to 0, FLASH_ECC_FA2R register is reset to 0 as well.

Bit 25 **CLR_SNECCERR2**: Bank 2 SNECCERR2 flag clear bit
Setting this bit to 1 resets to 0 SNECCERR2 flag in FLASH_SR2 register. If the DBECCERR2 flag of FLASH_SR2 register is cleared to 0, FLASH_ECC_FA2R register is reset to 0 as well.

Bit 24 **CLR_RDSERR2**: Bank 2 RDSERR2 flag clear bit
Setting this bit to 1 resets to 0 RDSERR2 flag in FLASH_SR2 register.

Bit 23 **CLR_RDPERR2**: Bank 2 RDPERR2 flag clear bit
Setting this bit to 1 resets to 0 RDPERR2 flag in FLASH_SR2 register.

Bit 22 **CLR_OPERR2**: Bank 2 OPERR2 flag clear bit
Setting this bit to 1 resets to 0 OPERR2 flag in FLASH_SR2 register.

Bit 21 **CLR_INCERR2**: Bank 2 INCERR2 flag clear bit
Setting this bit to 1 resets to 0 INCERR2 flag in FLASH_SR2 register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CLR_STRBERR2**: Bank 2 STRBERR2 flag clear bit
Setting this bit to 1 resets to 0 STRBERR2 flag in FLASH_SR2 register.

Bit 18 **CLR_PGSERR2**: Bank 2 PGSERR2 flag clear bit
Setting this bit to 1 resets to 0 PGSERR2 flag in FLASH_SR2 register.

Bit 17 **CLR_WRPERR2**: Bank 2 WRPERR2 flag clear bit
Setting this bit to 1 resets to 0 WRPERR2 flag in FLASH_SR2 register.

Bit 16 **CLR_EOP2**: Bank 2 EOP2 flag clear bit
Setting this bit to 1 resets to 0 EOP2 flag in FLASH_SR2 register.

Bits 15:0 Reserved, must be kept at reset value.

3.9.28 FLASH protection address for bank 2 (FLASH_PRAR_CUR2)

Address offset: 0x128

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMEP2	Res.	Res.	Res.	PROT_AREA_END2											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PROT_AREA_START2											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **DMEP2**: Bank 2 PCROP protected erase enable option status bit
If DMEP2 is set to 1, the PCROP protected area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **PROT_AREA_END2**: Bank 2 PCROP area end status bits
These bits contain the last 256-byte block of the PCROP area in bank 2.
If this address is equal to PROT_AREA_START2, the whole bank 2 is PCROP protected.
If this address is lower than PROT_AREA_START2, no protection is set on bank 2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PROT_AREA_START2**: Bank 2 PCROP area start status bits
These bits contain the first 256-byte block of the PCROP area in bank 2.
If this address is equal to PROT_AREA_END2, the whole bank 2 is PCROP protected.
If this address is higher than PROT_AREA_END2, no protection is set on bank 2.

3.9.29 FLASH protection address for bank 2 (FLASH_PRAR_PRG2)

Address offset: 0x12C

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMEP2	Res.	Res.	Res.	PROT_AREA_END2											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PROT_AREA_START2											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DMEP2**: Bank 2 PCROP protected erase enable option configuration bit
If DMEP2 is set to 1, the PCROP protected area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **PROT_AREA_END2**: Bank 2 PCROP area end configuration bits
These bits contain the last 256-byte block of the PCROP area in bank 2.
If this address is equal to PROT_AREA_START2, the whole bank 2 is PCROP protected.
If this address is lower than PROT_AREA_START2, no protection is set on bank 2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PROT_AREA_START2**: Bank 2 PCROP area start configuration bits
These bits contain the first 256-byte block of the PCROP area in bank 2.
If this address is equal to PROT_AREA_END2, the whole bank 2 is PCROP protected.
If this address is higher than PROT_AREA_END2, no protection is set on bank 2.

3.9.30 FLASH secure address for bank 2 (FLASH_SCAR_CUR2)

Address offset: 0x130

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMES2	Res.	Res.	Res.	SEC_AREA_END2											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SEC_AREA_START2											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **DMES2**: Bank 2 secure protected erase enable option status bit
If DMES2 is set to 1, the secure protected area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SEC_AREA_END2**: Bank 2 secure-only area end status bits
These bits contain the last 256-byte block of the secure-only area in bank 2.
If this address is equal to SEC_AREA_START2, the whole bank 2 is secure protected.
If this address is lower than SEC_AREA_START2, no protection is set on bank 2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SEC_AREA_START2**: Bank 2 secure-only area start status bits
These bits contain the first 256-byte block of the secure-only area in bank 2.
If this address is equal to SEC_AREA_END2, the whole bank 2 is secure protected.
If this address is higher than SEC_AREA_END2, no protection is set on bank 2.

3.9.31 FLASH secure address for bank 2 (FLASH_SCAR_PRG2)

Address offset: 0x134

Reset value: 0xXXXX 0XXX (see *Table 16: Option byte organization*)

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMES2	Res.	Res.	Res.	SEC_AREA_END2											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SEC_AREA_START2											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DMES2**: Bank 2 secure access protected erase enable option configuration bit
If DMES2 is set to 1, the secure access only area in bank 2 is erased when a protection level regression (change from level 1 to 0) or a bank erase with protection removal occurs.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **SEC_AREA_END2**: Bank 2 secure-only area end configuration bits
These bits contain the last of 256 bytes block of the secure-only area in bank 2.
If this address is equal to SEC_AREA_START2, the whole bank 2 is secure access only.
If this address is lower than SEC_AREA_START2, no protection is set on bank 2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SEC_AREA_START2**: Bank 2 secure-only area start configuration bits
These bits contain the first of 256 bytes block of the secure-only area in bank 2.
If this address is equal to SEC_AREA_END2, the whole bank 2 is secure access only.
If this address is higher than SEC_AREA_END2, no protection is set on bank 2.

3.9.32 FLASH write sector protection for bank 2 (FLASH_WPSN_CUR2R)

Address offset: 0x138

Reset value: 0x0000 00XX

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn2							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSn2**: Bank 2 sector write protection option status byte

Each FLASH_WPSGN_CUR2R bit reflects the write protection status of the corresponding bank 2 sector (0: sector is write protected; 1: sector is not write protected)

3.9.33 FLASH write sector protection for bank 2 (FLASH_WPSN_PRG2R)

Address offset: 0x13C

Reset value: 0x0000 00XX

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn2							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSn2**: Bank 2 sector write protection option status byte

Setting WRPSn2 bits to 0 write protects the corresponding bank 2 sector (0: sector is write protected; 1: sector is not write protected)

3.9.34 FLASH CRC control register for bank 2 (FLASH_CRCCR2)

Address offset: 0x150

Reset value: 0x001C 0000

The values in this register can be changed only if CRC_EN bit is set to 1 in FLASH_CR2 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BURST		Res.	Res.	CLEAN_CRC	START_CRC
										rw	rw			w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CLEAN_SECT	ADD_SECT	CRC_BY_SECT	ALL_BANK	Res.	Res.	Res.	Res.	CRC_SECT		
					w	w	rw	w					rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **CRC_BURST**: Bank 2 CRC burst size

CRC_BURST bits set the size of the bursts that are generated by the CRC calculation unit.

00: every burst has a size of 4 Flash words (256-bit)
01: every burst has a size of 16 Flash words (256-bit)
10: every burst has a size of 64 Flash words (256-bit)
11: every burst has a size of 256 Flash words (256-bit)

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CLEAN_CRC**: Bank 2 CRC clear bit

Setting CLEAN_CRC to 1 clears the current CRC result stored in the FLASH_CRCDATAR register.

Bit 16 **START_CRC**: Bank 2 CRC start bit

START_CRC bit triggers a CRC calculation on bank 2 using the current configuration. It is not possible to start a CRC calculation when an option byte change operation is ongoing because all write accesses to embedded Flash memory registers are put on hold until the option byte change operation has completed.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CLEAN_SECT**: Bank 2 CRC sector list clear bit

Setting CLEAN_SECT to 1 clears the list of sectors on which the CRC is calculated.

Bit 9 **ADD_SECT**: Bank 2 CRC sector select bit

Setting ADD_SECT to 1 adds the sector whose number is CRC_SECT to the list of sectors on which the CRC is calculated.

Bit 8 **CRC_BY_SECT**: Bank 2 CRC sector mode select bit

When CRC_BY_SECT is set to 1, the CRC calculation is performed at sector level, on the sectors selected by CRC_SECT.

When CRC_BY_SECT is reset to 0, the CRC calculation is performed on all addresses between CRC_START_ADDR and CRC_END_ADDR.

Bit 7 **ALL_BANK**: Bank 2 CRC select bit

When ALL_BANK is set to 1, all bank 2 user sectors are added to the list of sectors on which the CRC is calculated.

Bits 6:3 Reserved, must be kept at reset value.

Bits 2:0 **CRC_SECT**: Bank 2 CRC sector number

CRC_SECT is used to select one or more user Flash sectors to be added to the CRC calculation. The CRC can be computed either between two addresses (using registers FLASH_CRCSADD2R and FLASH_CRCEADD2R) or on a list of sectors. If this latter option is selected, it is possible to add a sector to the list of sectors by programming the sector number in CRC_SECT and then setting ADD_SECT. to 1

The list of sectors can be erased either by setting CLEAN_SECT bit or by disabling the CRC computation. CRC_SECT can be set only when CRC_EN of FLASH_CR register is set to 1.

000: sector 0 of bank 2

001: sector 1 of bank 2

...

111: sector 7 of bank 2

3.9.35 FLASH CRC start address register for bank 2 (FLASH_CRCSADD2R)

Address offset: 0x154

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_START_ADDR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_START_ADDR[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:2 **CRC_START_ADDR[19:2]**: CRC start address on bank 2

CRC_START_ADDR is used when CRC_BY_SECT is cleared to 0. It must be programmed to the start address of the bank 2 memory area on which the CRC calculation is performed.

Bits 1:0 Reserved, must be kept at reset value.

3.9.36 FLASH CRC end address register for bank 2 (FLASH_CRCEADD2R)

Address offset: 0x158

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_END_ADDR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_END_ADDR[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:2 **CRC_END_ADDR[19:2]**: CRC end address on bank 2

CRC_END_ADDR is used when CRC_BY_SECT is cleared to 0. It must be programmed to the end address of the bank 2 memory area on which the CRC calculation is performed.

Bits 1:0 Reserved, must be kept at reset value.

3.9.37 FLASH ECC fail address for bank 2 (FLASH_ECC_FA2R)

Address offset: 0x160

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FAIL_ECC_ADDR2														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **FAIL_ECC_ADDR2**: Bank 2 ECC error address

When an ECC error occurs (both for single error correction or double detection) during a read operation from bank 2, the FAIL_ECC_ADDR2 bitfield contains the address that generated the error.

FAIL_ECC_ADDR2 is reset when the flag error in the FLASH_SR2 register (CLR_SNECCERR2 or CLR_DBECCERR2) is reset.

The embedded Flash memory programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.

3.10 FLASH register map and reset values

Table 23. Register map and reset value table

Offset	Register name reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRHIGHFREQ		LATENCY			
	0x00000037																											1	1	0	1	1	1
0x004	FLASH_KEYR1	KEY1R																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	FLASH_ OPTKEYR	OPTKEYR																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	FLASH_CR1	Res.	Res.	Res.	Res.	CRCENDIE1	DBECCERRIE1	SNECCERRIE1	RDSERRIE1	RDPERRIE1	OPERRIE1	INCERRIE1	Res.	STRBERRIE1	PGSERRIE1	WRPERRIE1	EOPIE1	CRC_EN	Res.	Res.	Res.	Res.	SNB1			START1	FW1	PSIZE1		BER1	SER1	PG1	LOCK1	
	0x00000031					0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	1	1	0	0	0	1
0x010	FLASH_SR1	Res.	Res.	Res.	Res.	CRCEND1	DBECCERR1	SNECCERR1	RDSERR1	RDPERR1	OPERRIE1	INCERR1	Res.	STRBERR1	PGSERR1	WRPERR1	EOP1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BUSY1	QW1	WBNE1	BSY1
	0x00000000					0	0	0	0	0	0	0		0	0	0	0													0	0	0	0
0x014	FLASH_CCR1	Res.	Res.	Res.	Res.	CLR_CRCEND1	CLR_DBECCERR1	CLR_SNECCERR1	CLR_RDSERR1	CLR_RDPERR1	CLR_OPERRIE1	CLR_INCERR1	Res.	CLR_STRBERR1	CLR_PGSERR1	CLR_WRPERR1	CLR_EOP1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	0x00000000					0	0	0	0	0	0	0		0	0	0	0																
0x018	FLASH_OPTCR	SWAP_BANK	OPTCHANGEERRIE																										MER			OPTSTART	OPTLOCK
	0x00000001	0	0																										0			0	1
0x01C	FLASH_OPTSR_ CUR	SWAP_BANK_OPT	OPTCHANGEERR	IO_HSLV	Res.	Res.	Res..	Res.	Res.	Res.	Res.	SECURITY	ST_RAM_SIZE		IWDG_FZ_SDBY	IWDG_FZ_STOP	Res.	RDP								NRST_STBY_D1	NRST_STOP_D1	Res.	IWDG1_SW	BOR_LEV		Res.	OPT_BUSY
	0xXXXX XXXX	X	X	X								X	X	X	X	X		X	X	X	X	X	X	X	X	X	X		X	X	X		X

Table 23. Register map and reset value table (continued)

Offset	Register name reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	FLASH_OPTSR_PRG	SWAP_BANK_OPT	Res.	IO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECURITY		ST_RAM_SIZE		IWDG_FZ_SDBY	IWDG_FZ_STOP	Res.	RDP							NRST_STBY_D1	NRST_STOP_D1	Res.	IWDG1_SW	BOR_LEV		Res.	Res.
	0xXXXX XXXX	X		X								X	X	X	X	X		X	X	X	X	X	X	X	X	X	X		X	X	X		
0x024	FLASH_OPTCCR	Res.	CLR_OPTCHANGEERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	0x00000000		0																														
0x028	FLASH_PRAR_CUR1	DMEP1	Res.	Res.	Res.	PROT_AREA_END1[11:0]												Res.	Res.	Res.	Res.	PROT_AREA_START1[11:0]											
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x02C	FLASH_PRAR_PRG1	DMEP1	Res.	Res.	Res.	PROT_AREA_END1[11:0]												Res.	Res.	Res.	Res.	PROT_AREA_START1[11:0]											
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x030	FLASH_SCAR_CUR1	DMES1	Res.	Res.	Res.	SEC_AREA_END1[11:0]												Res.	Res.	Res.	Res.	SEC_AREA_START1[11:0]											
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x034	FLASH_SCAR_PRG1	DMES1	Res.	Res.	Res.	SEC_AREA_END1[11:0]												Res.	Res.	Res.	Res.	SEC_AREA_START1[11:0]											
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x038	FLASH_WPSN_CUR1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn1							
	0x0000 00XX	0																								X	X	X	X	X	X	X	X
0x03C	FLASH_WPSN_PRG1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn1							
	0x0000 00XX																									X	X	X	X	X	X	X	X
0x040	FLASH_BOOT_CURR	BOOT_ADD1[15:0]																BOOT_ADD0[15:0]															
	0xXXXX XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x044	FLASH_BOOT_PRGR	BOOT_ADD1[15:0]																BOOT_ADD0[15:0]															
	0xXXXX XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x050	FLASH_CRCCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BURST		Res.	Res.	CLEAN_CRC	START_CRC	Res.	Res.	Res.	Res.	Res.	CLEAN_SECT	ADD_SECT	CRC_BY_SECT	ALL_BANK	Res.	Res.	Res.	Res.	CRC_SECT		
	0x001C0000											0	1			0	0						0	0	0	0					0	0	0

Table 23. Register map and reset value table (continued)

Offset	Register name reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	FLASH_ CRCSADD1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_START_ADDR[31:0]																		Res.	Res.
	0x00000000													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x058	FLASH_ CRCEADD1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_END_ADDR[31:0]																		Res.	Res.
	0x00000000													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x05C	FLASH_ CRCDATAR	CRC_DATA[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	FLASH_ECC_ FA1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAIL_ECC_ADDR1[14:0]														
	0x00000000																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRHIGHFREQ		LATENCY			
	0x00000037																											1	1	0	1	1	1
0x104	FLASH_KEYR2	KEY2R																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	FLASH_ OPTKEYR	OPTKEYR																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	FLASH_CR2	Res.	Res.	Res.	Res.	CRCENDIE2	DBECCERRIE2	SNECCERRIE2	RDSERRIE2	RDPERRIE2	OPERRIE2	INCERRIE2	Res.	STRBERRIE2	PGSERRIE2	WRPERRIE2	EOPIE2	CRC_EN	SPSS2	Res.	Res.	Res.	SNB2[2:0]			START2	FW2	PSIZE2		BER2	SER2	PG2	LOCK2
	0x00000031					0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	1	1	0	0	0	1
0x110	FLASH_SR2	Res.	Res.	Res.	Res.	CRCEND2	DBECCERR2	SNECCERR2	RDSERR2	RDPERR2	OPERR2.	INCERR2	Res.	STRBERR2	PGSERR2	WRPERR2	EOP2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BUSY2	QW2	WBNE2	BSY2
	0x00000000					0	0	0	0	0	0	0		0	0	0	0													0	0	0	0
0x114	FLASH_CCR2	Res.	Res.	Res.	Res.	CLR_CRCEND2	CLR_DBECCERR2	CLR_SNECCERR2	CLR_RDSERR2	CLR_RDPERR2	CLR_OPERR2	CLR_INCERR2	Res.	CLR_STRBERR2	CLR_PGSERR2	CLR_WRPERR2	CLR_EOP2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	0x00000000					0	0	0	0	0	0	0		0	0	0	0																

Table 23. Register map and reset value table (continued)

Offset	Register name reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x118	FLASH_OPTCR	SWAP_BANK	OPTCHANGEERRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MER	Res.	Res.	OPTSTART	OPTLOCK
	0x00000001	0	0																										0			0	1
0x11C	FLASH_OPTSR_CUR	SWAP_BANK_OPT	OPTCHANGEERR	IO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECURITY	ST_RAM_SIZE		IWDG_FZ_SDBY	IWDG_FZ_STOP	Res.	RDP								NRST_STBY_D1	NRST_STOP_D1	Res.	IWDG1_SW	BOR_LEV		Res.	OPT_BUSY
	0xXXXX XXXX	X	0	X								X	X	X	X	X		X	X	X	X	X	X	X	X	X	X		X	X	X		X
0x120	FLASH_OPTSR_PRG	SWAP_BANK_OPT	Res.	IO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECURITY	ST_RAM_SIZE		FZ_IWDG_SDBY	FZ_IWDG_STOP	Res.	RDP								nRST_STDY	nRST_STOP	Res.	IWDG1_SW	BOR_LEV		Res.	Res.
	0xXXXX XXXX	X		X								X	X	X	X	X		X	X	X	X	X	X	X	X	X	X		X	X	X		
0x124	FLASH_OPTCCR	Res.	CLR_OPTCHANGEERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	0x00000000		0																														
0x128	FLASH_PRAR_CUR2	DMEP2	Res.	Res.	Res.	PROT_AREA_END2[11:0]												Res.	Res.	Res.	Res.	PROT_AREA_START2[11:0]												
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x12C	FLASH_PRAR_PRG2	DMEP2	Res.	Res.	Res.	PROT_AREA_END2[11:0]												Res.	Res.	Res.	Res.	PROT_AREA_START2[11:0]												
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x130	FLASH_SCAR_CUR2	DMES2	Res.	Res.	Res.	SEC_AREA_END2[11:0]												Res.	Res.	Res.	Res.	SEC_AREA_START2[11:0]												
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x134	FLASH_SCAR_PRG2	DMES2	Res.	Res.	Res.	SEC_AREA_END2[11:0]												Res.	Res.	Res.	Res.	SEC_AREA_START2[11:0]												
	0xXXXX 0XXX	X				X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x138	FLASH_WPSN_CUR2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn2							
	0x0000 00XX																									X	X	X	X	X	X	X	X

Table 23. Register map and reset value table (continued)

Offset	Register name reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x13C	FLASH_WPSN_PRG2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSn2							
	0x0000 00XX																									X	X	X	X	X	X	X	X
0x140	FLASH_BOOT_CURR	BOOT_ADD1[15:0]																BOOT_ADD0[15:0]															
	0xXXXX XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x144	FLASH_BOOT_PRGR	BOOT_ADD1[15:0]																BOOT_ADD0[15:0]															
	0xXXXX XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x150	FLASH_CRCCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_BURST		Res.	Res.	CLEAN_CRC	START_CRC	Res.	Res.	Res.	Res.	Res.	CLEAN_SECT	ADD_SECT	CRC_BY_SECT	ALL_BANK	Res.	Res.	Res.	Res.	CRC_SECT		
	0x001C0000											0	1			0	0						0	0	0	0					0	0	0
0x154	FLASH_CRCS ADD2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_START_ADDR[19:2]																		Res.	Res.
	0x00000000													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x158	FLASH_CRCEADD2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_END_ADDR[19:2]																		Res.	Res.
	0x00000000													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x15C	FLASH_CRCDATAR	CRC_DATA[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x160	FLASH_ECC_FA2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAIL_ECC_ADDR2[15:0]															
	0x00000000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

4 Secure memory management

4.1 Introduction

STM32H743/753 and STM32H750 microcontrollers offer a first set of protection mechanisms, which are similar to the latest STM32 Series (STM32F4/F7 and STM32L0/L4):

- Global readout device protection
- Write protection
- Proprietary code readout protection (PCROP)

A summary of these protection mechanisms is given in *Section 4.3: Flash protections*.

STM32H743/753 and STM32H750 also offer a new enhanced protection mode, the Secure access mode, that makes possible the development of user-defined secure services (e.g. secure firmware update) and guarantees of a safe execution and protection of both code and data. This new mechanism is described in details in *Section 4.4: Secure access mode*, *Section 4.5: Root secure services (RSS)* and *Section 4.6: Secure user software*.

The secure memory management unit is contained inside the D1 domain.

4.2 Glossary

The following terms will be used in herein:

Table 24. List of preferred terms

Term	Description
Device Security Level	
Standard mode	Device state which allows accessing the user Flash memory, the option bytes and the bootloader area
Secure access mode	Device state which allows to access all the memory areas of the device
Memory areas	
System memory	ST reserved memory area used to store ST ROM code
User Flash memory	Flash memory area used to store user code and data
Secure user memory/area	This area can be configured to be accessed once after reset and be hidden for the firmware stored in the user Flash memory after the code stored in this area is executed.

Table 24. List of preferred terms (continued)

Term	Description
SW services	
Bootloader	STMicroelectronics software executed at reset which allows downloading firmware from regular communication port
Root secure services (RSS)	STMicroelectronics software which allows encrypted firmware/module decryption and installation into secure and non-secure user memory
Secure bootloader	Complementary services composed by an STMicroelectronics bootloader plus ST root secure services.
Secure user software	User software executed once after reset, which can be used to store SFU and firmware initialization. Secure User SW is located in Secure User Memory
Secure Firmware Update (SFU)	User software used to download and update installed firmware from user Flash memory

4.3 Flash protections

Three protection mechanisms can be used in the end-user application or during development involving proprietary code (such as third-party libraries):

- Readout device protection (RDP)

  This is a global protection mechanism, which is suitable for end products. It prevents the device from being accessed externally through the debug port or boot from RAM or bootloader.
- Proprietary code readout protection (PCROP)

  This mechanism can be set on specific Flash memory areas that must be execute-only. PCROP is used to avoid sensitive software dumping.
- Write protection

  This protection can be set at Flash sector level. It prevents code or data from being accidentally erased.

Table 25 summarizes the protection mechanisms that are available on STM32H743/753 and STM32H750 as well as most STM32 devices. On STM32H743/753 and STM32H750, they are available both in Standard and Secure access modes.

For further details, refer to *Section 3: Embedded Flash memory (FLASH)*.

Table 25. Flash protection mechanisms

Flash Protection	Scope	Description
RDP	Global (All sectors of all banks + backup SRAM)	Protection Level 1: – No access (read, erase and program) to Flash memory or backup SRAM can be performed when the debugger is connected or when the boot configuration is different from user Flash memory. – Protection can be removed by performing Flash mass erase Protection level 2: – Debug access (SW or JTAG) disabled – No System bootloader access(1) – No RAM boot allowed – Protection level2 is permanent
Write protection	At sector level (128 Kbytes)	Protection against unwanted write/modification of user Flash memory sectors
PCROP/execute-only area	At area level (256-byte granularity)	Only execute accesses are allowed. This protection is set on sensitive code to prevent debug accesses and code copy by other software.

1. Access to root secure services in Secure access mode is allowed. Refer to *Section 4.4: Secure access mode*.

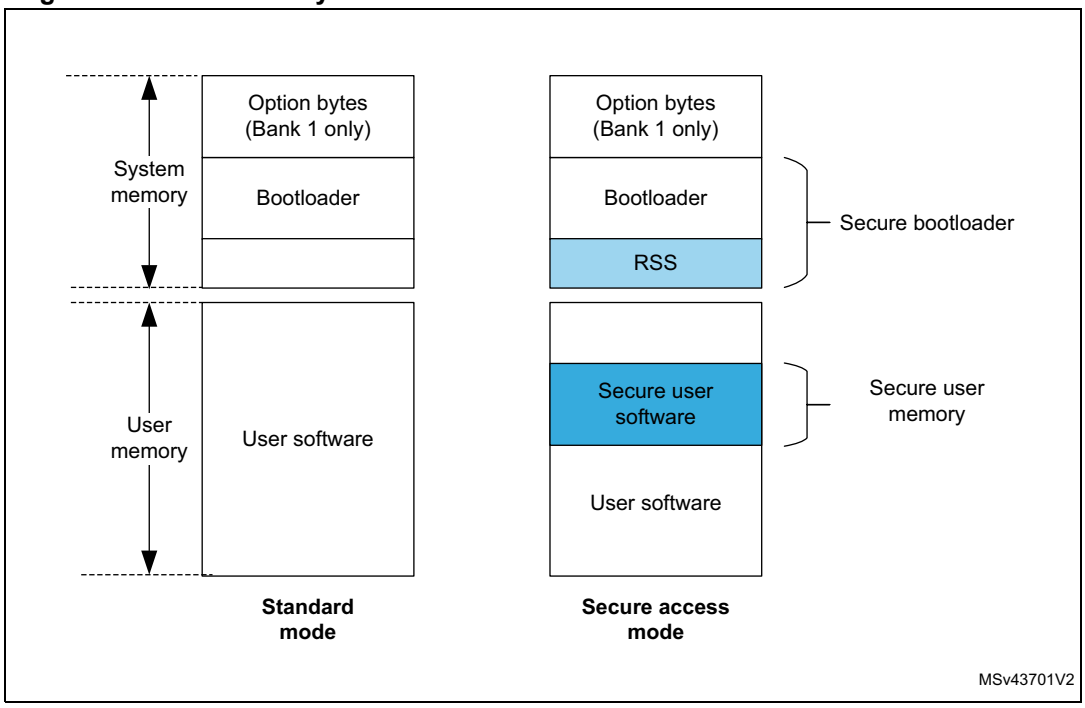
4.4 Secure access mode

Some sensitive functions require safe execution from potential malicious software attacks. Secure firmware update (SFU) software is a good example of code that requires a high level of protection since it handles secret data (such as encrypted firmware or cryptographic keys) that shall not be retrieved by other processes.

STM32H743/753 and STM32H750 microcontrollers feature secure memory areas with restricted access. They allow building secure services that will be executed prior to any user application. These secure areas, together with the software they contain, are only accessible when configuring a new device in Secure access mode.

Figure 13 gives an overview of Flash memory areas and services in Standard and Secure access modes.

Figure 13. Flash memory areas and services in Standard and Secure access modes



1. The protected areas that can only be accessed in Secure access mode are shown in blue.

4.4.1 Associated features

The Secure access mode can be configured through option bytes. When it is set, it enables access to:

- the secure bootloader, which embeds the bootloader plus some secure services provided by STMicroelectronics (see *Section 4.5: Root secure services (RSS)*)
- Secure user memory which embeds secure user code and data.

For a summary of access rights for each core, refer to *Section 4.7: Summary of Flash protection mechanisms*.

4.4.2 Boot state machine

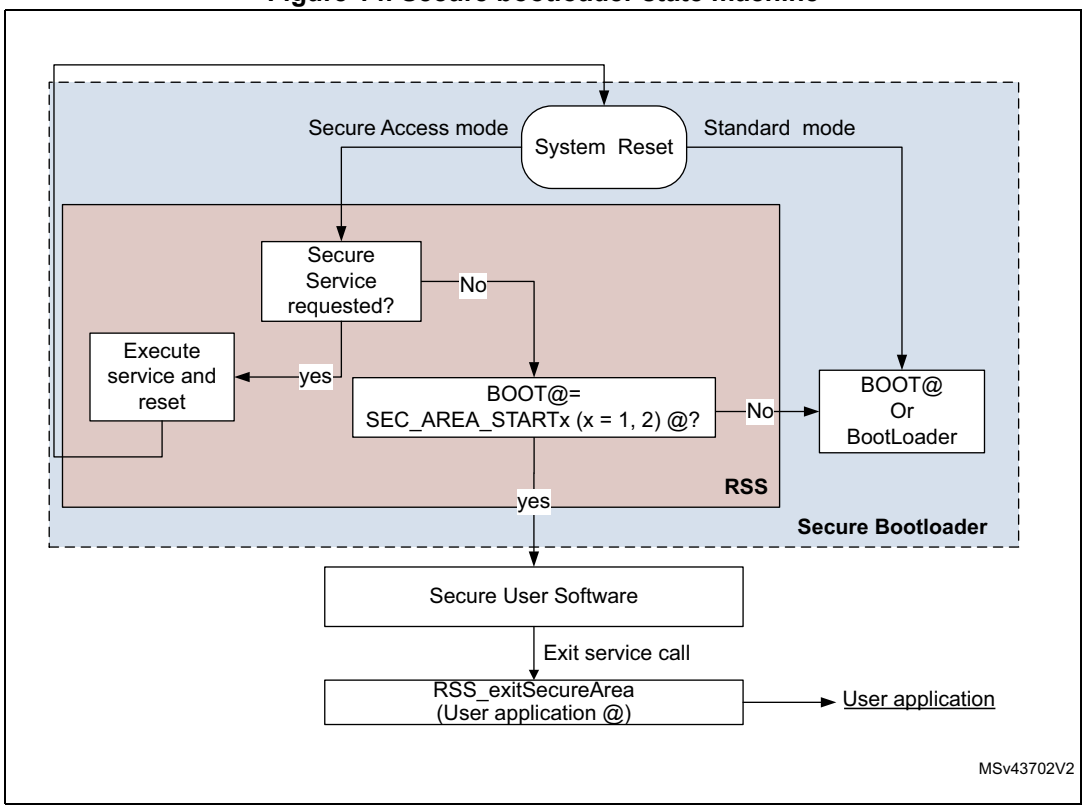
In Secure access mode, the secure bootloader replaces the standard bootloader. Booting in the RSS is forced independently from boot configuration (boot pins and boot addresses). Root secure services and Secure user software will hence preempt all other codes in the device (including classical bootloader).

Execution priority

1. Root secure services have the highest priority after a system reset. The service called will be executed and a new system reset will be triggered immediately afterwards. The available secure services are detailed in *Section 4.5: Root secure services (RSS)*.
2. If no service is requested, the Secure user software will be executed provided it has been defined and the boot address has been set accordingly. Refer to *Section 4.6: Secure user software* for Secure user software settings.
3. Otherwise, the classical boot state-machine will be followed: bootloader or boot from Flash memory.

Figure 14 shows STM32H743/753 and STM32H750 boot state machine.

Figure 14. Secure bootloader state machine



1. On STM32H750xB devices, only SEC_AREA_START1 is applicable.

4.4.3 Secure access mode configuration

Enabling Secure access mode

There is no restriction on how to activate Secure access mode on the device. It is configured through the SECURITY option bit in FLASH_OPTSR_CUR register (see *Section 3.9.8: FLASH option status register (FLASH_OPTSR_CUR)*).

The Secure access mode becomes active after a system reset.

Disabling Secure access mode

Disabling Secure access mode is a more sensitive task as it can only be done if no more protected code exists on the device. As a result, to come back to Standard mode, Secure user memories and PCROP/execute-only areas shall be removed before clearing the SECURITY option bit in the FLASH_OPTSR_CUR register.

Protected areas can be removed either by performing a Flash mass erase, a bank erase or by calling a dedicated secure service (for PCROP areas only):

- Refer to *Section 3.5.4: Proprietary code readout protection (PCROP)* and to *RSS_resetAndDestroyPCROPArea* secure service for removing PCROP protection.
- Refer to *Section 3.5: FLASH protection mechanisms* for an explanation on how to remove secure areas.

4.5 Root secure services (RSS)

STMicroelectronics provides services to configure secure areas. These root secure services are executed after a system reset in Secure access mode and prior to any other software stored in the device.

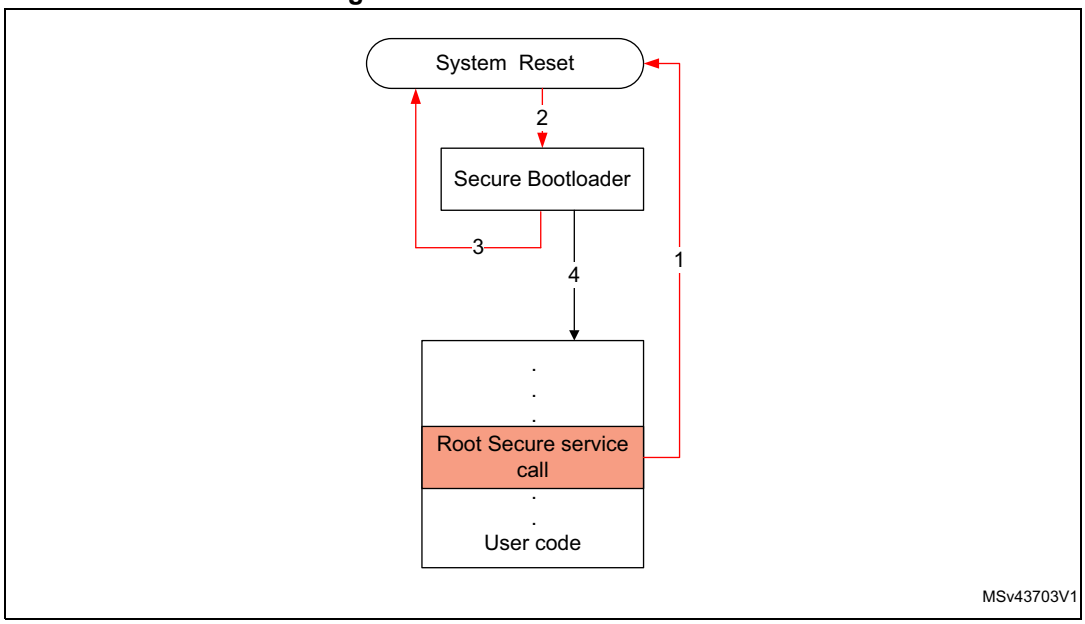
RSS software cannot be accessed (read, write, execute and debug) when the STM32H743/753 and STM32H750 operates in Standard mode.

When the STM32H743/753 and STM32H750 is configured in Secure access mode, the RSS software is executed only once at reset. When RSS software execution is complete, it cannot be accessed anymore (no other code can jump directly to RSS). Any RSS routine is called by performing a reset.

4.5.1 Calling root secure services

A secure service is called through a public API that triggers a software reset (see *Figure 15*, step 1) so that the secure bootloader is executed with RSS code available (see *Figure 15*, step 2). Once executed, most services trigger a system reset (see *Figure 15*, step 3) and as no more service is required, the secure bootloader jumps to the user software (see *Figure 15*, step 4).

Figure 15. Root secure service call



4.5.2 Root secure services description

The following secure services are available for managing PCROP, secure user memory and secure mode settings.

RSS_resetAndInitializeSecureAreas

Prototype	void RSS_resetAndInitializeSecureAreas(RSS_SecureArea_t area)
Arguments	Secure user areas start and end addresses. One or two Secure user areas can be set.
Description	This service sets Secure user area boundaries, following the values stored in the option byte registers: – SEC_AREA_START1 and SEC_AREA_END1 for bank 1 – SEC_AREA_START2 and SEC_AREA_END2 for bank 12 This service can be used only when a secure area is set for the first time. A system reset is triggered after service completion.

RSS_exitSecureArea

Prototype	void RSS_exitSecureArea(unsigned int vectors)
Arguments	Address of application vectors where to jump after exit
Description	This service is used to exit from secure user software and jump to user main application. There is no system reset triggered by this service

RSS_resetAndDestroyPCROPArea

Prototype	RSS_resetAndDestroyPCROPArea(RSS_FlashBank_et bank)
Arguments	Targeted bank number
Description	This service overwrites PCROP area and removes the PCROP protection. Avoid a Flash/bank mass erase. A system reset is triggered after service completion

4.6 Secure user software

A Secure user software is a trusted piece of code that is executed after device power-on or after a system reset. It allows building secure applications such as:

- code signature or integrity checking (user secure boot).
- software license checking
- secure firmware update

4.6.1 Access rules

Only accessible in Secure access mode, the Secure user software is stored in the secure user memory, a configurable protected area which is part of the user main memory.

Only one user secure area per bank can be configured.

Only root secure services have higher priority and can preempt secure user software execution (see *Section 4.4.2: Boot state machine*).

After secure user software execution, the code shall jump to the main user application and prevent access to the secure user area. This is done by calling *RSS_exitSecureArea*s secure service with the application code address given as parameter.

Once in the application code, any access to the secure user area triggers a Flash error.

4.6.2 Setting secure user memory areas

One secure area of configurable size can be set in each bank. The size of each area can be set from 512 bytes to full bank with a granularity of 256 bytes:

- Secure area in bank 1

  Boundaries are configured through SEC_AREA_START1 and SEC_AREA_END1 option bits in FLASH_SCAR_CUR1 (see *Section 3.9.13: FLASH secure address for bank 1 (FLASH_SCAR_CUR1)*).
- Secure area in bank 2

  Boundaries are configured through SEC_AREA_START2 and SEC_AREA_END2 option bits in FLASH_SCAR_CUR2 (see *Section 3.9.30: FLASH secure address for bank 2 (FLASH_SCAR_CUR2)*).

Note: If the secure area start address is equal to the secure area end address, the full bank is considered as secure protected.

Active setting then differs depending on whether a secure user area is defined or not in the device:

- No secure user area exists (first time setting):

  A dedicated root secure service, *RSS_resetAndInitializeSecureAreas*, shall be called to initialize the empty address in the ST boot state machine. Both secure areas (one per bank) can be set in the same operation.
- A secure user area already exists:

  In this case, the secure user area code can update its own secure user area size or create a new one in the other bank.

  Note that *RSS_resetAndInitializeSecureAreas* function cannot be used to create a second user secure area.

4.6.3 Removing secure user memory areas

Secure user areas can only be erased by performing a Flash mass erase or a bank erase operation.

- A Flash mass erase is triggered by a level regression from RDP level 1 to RDP level 0.
- A bank erase is triggered by setting BER1/2 bit of FLASH_CR1/2 register (see *Section 3.9.4: FLASH control register for bank 1 (FLASH_CR1)* and *Section 3.9.25: FLASH control register for bank 2 (FLASH_CR2)*).

In both cases, the DMES1/2 bit that defines the erase policy in the FLASH_PRAR_CUR1/2 register (see *Section 3.9.11: FLASH protection address for bank 1 (FLASH_PRAR_CUR1)* and *Section 3.9.28: FLASH protection address for bank 2 (FLASH_PRAR_CUR2)* shall be set to 1 and the area start address shall be superior to its end address.

4.6.4 Selecting secure user software

The current boot address selects one of the two secure user software (one per bank). As two boot addresses can be defined, several use-cases have to be considered.

The selection of the boot address between BOOT_ADD0 and BOOT_ADD1 in FLASH_BOOT_CURR register is done by the BOOT pin (see *Section 2.6: Boot configuration*) or by the swap bank mechanism (see *Section : Bank swapping*).

Table 26 shows different boot address register setting and the potential associated scenarios.

Table 26. Secure user software selection use-cases(1)

BOOT_ADD0 (boot pin=0)	BOOT_ADD1 (boot pin=1)	Use case
SEC_AREA_START1	SEC_AREA_START2	Use the BOOT pin to switch between two secure user software (current and recovery or current and previous)(2)
SEC_AREA_START1	SEC_AREA_START1	Use the bank swapping feature to switch between two secure user software starting at same offset in each bank (previous secure boot to current) (2)
SEC_AREA_START1	Recovery firmware	Use the BOOT pin to switch between secure user software and non-secure recovery boot
SEC_AREA_START1	Bootloader	Use BOOT pin for bootloader based recovery (in non-secure mode)

1. Bank number and boot address register number are not related.
2. This case is not applicable to STM32H750xB devices.

If the boot address is different from the two secure user areas, the system jump directly to the selected address, bypassing the secure user software.

4.7 Summary of Flash protection mechanisms

Figure 16 and *Table 27* summarize the access rights of the different Flash memory areas, both in Secure access and Standard modes.

Figure 16. Core access to Flash memory areas

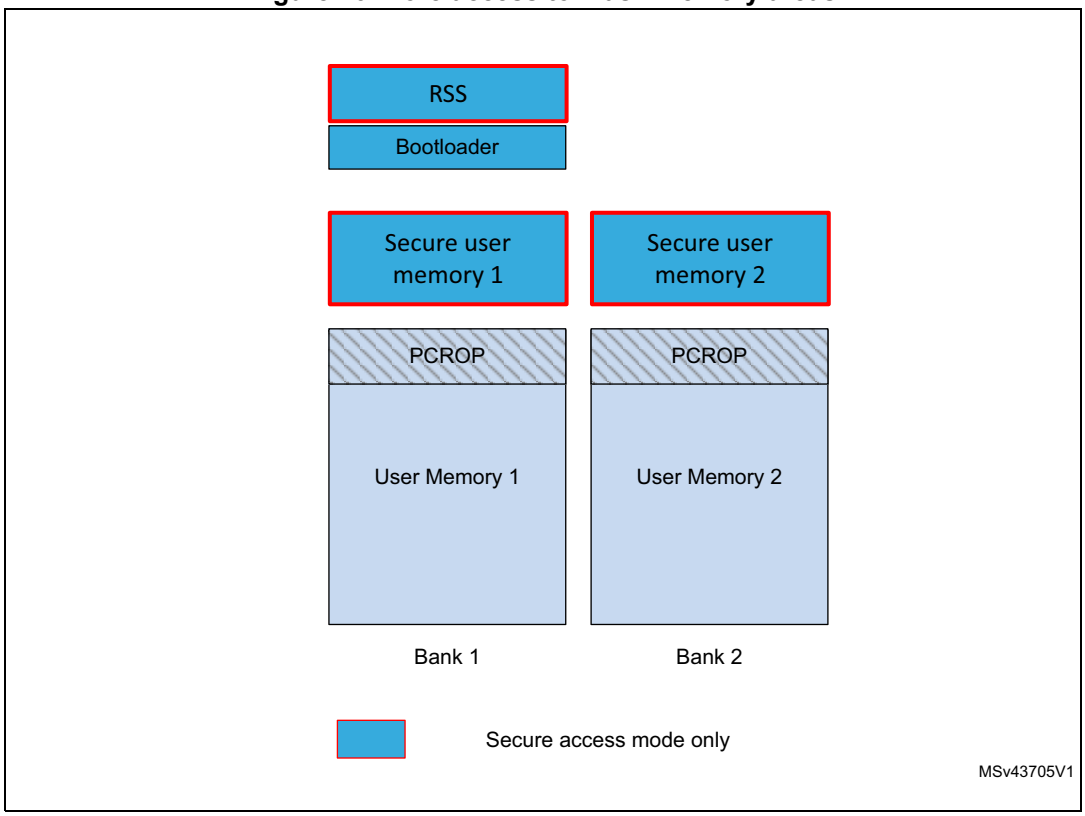


Table 27. Summary of Flash protected areas access rights

Access type	SW Area	Security Mode	Access
Execution	PCROP	Any	✓
	Secure user software	Secure access	✓ (1)
	Root secure services	Secure access	✓ (1)
Read access	PCROP	Any	No
	Secure User software	Secure access	✓(1)
	Root secure services	Secure access	✓ (1)
Debug access	PCROP	Any	No
	Secure User software	Secure access	No
	Root secure services	Secure access	No

1. Access rights granted after reset until code completion only.

5 Power control (PWR)

5.1 Introduction

The Power control section (PWR) provides an overview of the supply architecture for the different power domains and of the supply configuration controller.

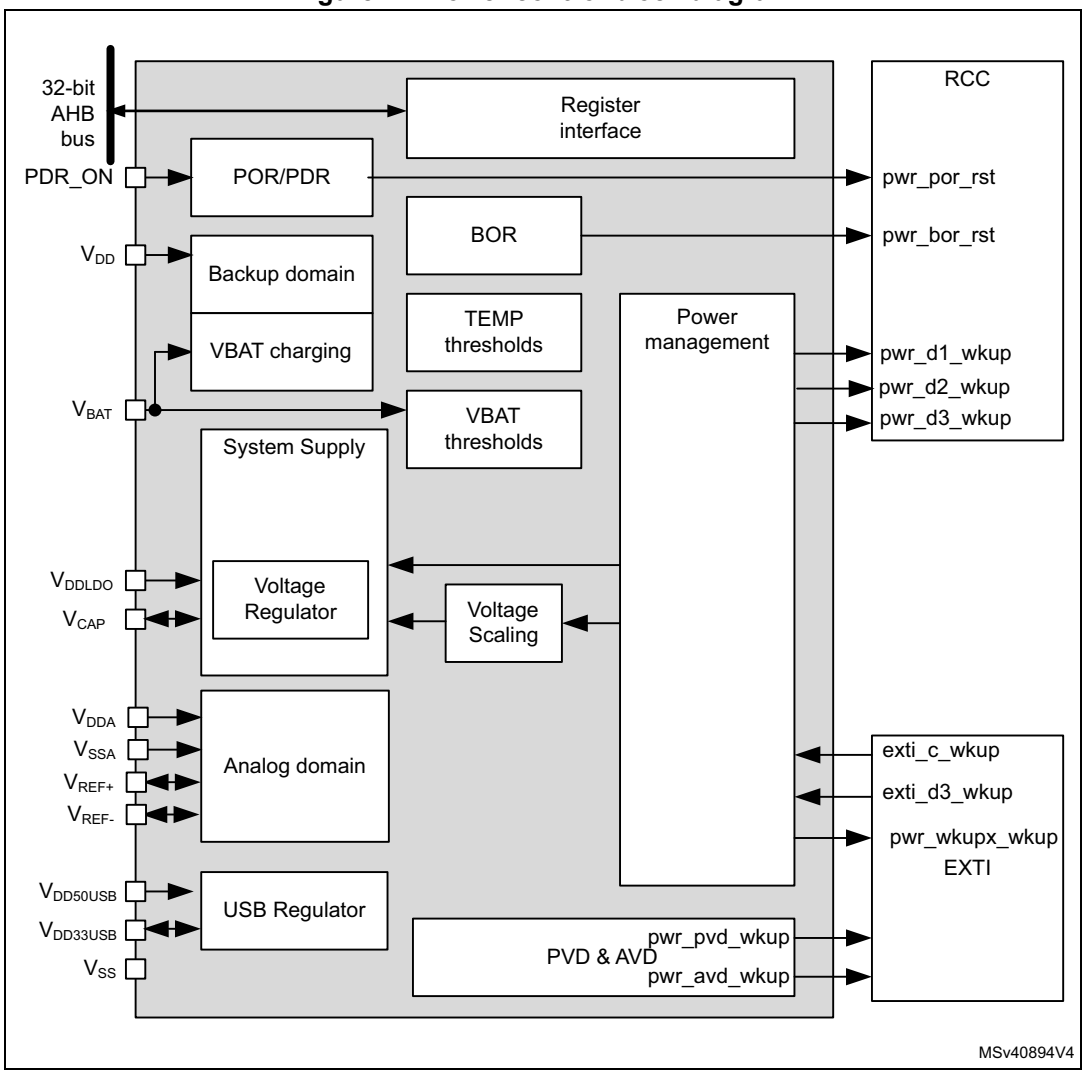
It also describes the features of the power supply supervisors and explains how the V_{CORE} supply domain is configured depending on the operating modes, the selected performance (clock frequency) and the voltage scaling.

5.2 PWR main features

- Power supplies and supply domains
  - Core domains (V_{CORE})
  - V_{DD} domain
  - Backup domain (V_{SW}, V_{BKP})
  - Analog domain (V_{DDA})
- System supply voltage regulation
  - Voltage regulator (LDO)
- Peripheral supply regulation
  - USB regulator
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - V_{BAT} thresholds
  - Temperature thresholds
- Power management
  - V_{BAT} battery charging
  - Operating modes
  - Voltage scaling control
  - Low-power modes

5.3 PWR block diagram

Figure 17. Power control block diagram



5.3.1 PWR pins and internal signals

Table 28 lists the PWR inputs and output signals connected to package pins or balls, while *Table 29* shows the internal PWR signals.

Table 28. PWR input/output signals connected to package pins or balls

Pin name	Signal type	Description
V_{DD}	Supply input	Main I/O and V_{DD} domain supply input
V_{DDA}	Supply input	External analog power supply for analog peripherals
V_{REF+},V_{REF-}	Supply Input/ Outputs	External reference voltage for ADCs and DAC
V_{BAT}	Supply input	Backup battery supply input
V_{DDLDO}	Supply input	Voltage regulator supply input
V_{CAP}	Supply Input/ Output	Digital core domain supply
$V_{DD50USB}$	Supply input	USB regulator supply input
$V_{DD33USB}$	Supply Input/ Output	USB regulator supply output
V_{SS}	Supply input	Main ground
AHB	Digital inputs/ outputs	AHB register interface
PDR_ON	Digital input	Power Down Reset enable

Table 29. PWR internal input/output signals

Signal name	Signal type	Description
pwr_pvd_wkup	Digital output	Programmable voltage detector output
pwr_avd_wkup	Digital output	Analog voltage detector output
pwr_wkupx_wkup	Digital output	CPU wakeup signals (x=1 to 6)

Table 29. PWR internal input/output signals (continued)

Signal name	Signal type	Description
pwr_por_rst	Digital output	Power-on reset
pwr_bor_rst	Digital output	Brownout reset
exti_c_wkup	Digital input	CPU wakeup request
exti_d3_wkup	Digital input	D3 domain wakeup request
pwr_d1_wkup	Digital output	D1 domain bus matrix clock wakeup request
pwr_d2_wkup	Digital output	D2 domain bus matrix clock wakeup request
pwr_d3_wkup	Digital output	D3 domain bus matrix clock wakeup request

5.4 Power supplies

The device requires V_{DD} power supply as well as independent supplies for V_{DDLDO}, V_{DDA}, V_{DDUSB}, and V_{CAP}. It also provides regulated supplies for specific functions (voltage regulator, USB regulator).

- V_{DD} external power supply for I/Os and system analog blocks such as reset, power management and oscillators
- V_{BAT} optional external power supply for backup domain when V_{DD} is not present (V_{BAT} mode)

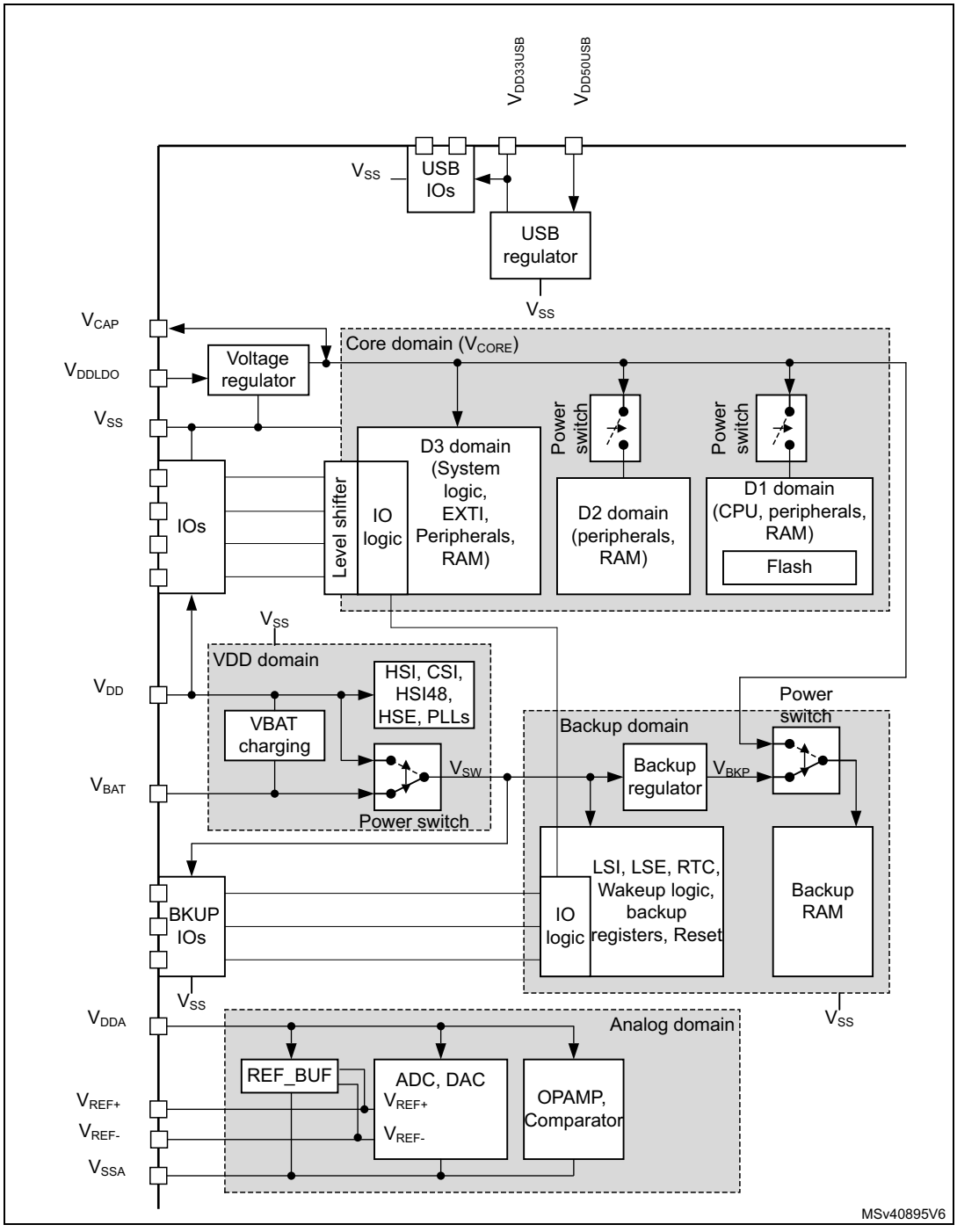
  This power supply shall be connected to V_{DD} when no battery is used.
- V_{DDLDO} external power supply for voltage regulator
- V_{CAP} digital core domain supply

  This power supply is independent from all the other power supplies:
  - When the voltage regulator is enabled, V_{CORE} is delivered by the internal voltage regulator.
  - When the voltage regulator is disabled, V_{CORE} is delivered by an external power supply through V_{CAP} pin.
- V_{DDA} external analog power supply for ADCs, DACs, OPAMPs, comparators and voltage reference buffers

  This power supply is independent from all the other power supplies.
- V_{REF+} external reference voltage for ADC and DAC.
  - When the voltage reference buffer is enabled, V_{REF+} and V_{REF-} are delivered by the internal voltage reference buffer.
  - When the voltage reference buffer is disabled, V_{REF+} is delivered by an independent external reference supply.
- V_{SSA} separate analog and reference voltage ground.
- $V_{DD50USB}$ external power supply for USB regulator.
- $V_{DD33USB}$ USB regulator supply output for USB interface.
  - When the USB regulator is enabled, $V_{DD33USB}$ is delivered by the internal USB regulator.
  - When he USB regulator is disabled, $V_{DD33USB}$ is delivered by an independent external supply input.
- V_{SS} common ground for all supplies and analog regulator.

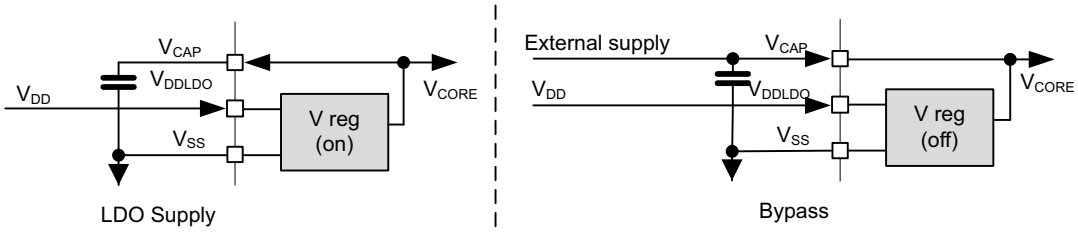
Note: Depending on the operating power supply range, some peripherals might be used with limited features and performance. For more details, refer to section "General operating conditions" of the device datasheets.

Figure 18. Power supply overview



By configuring the voltage regulator the supply configurations shown in *Figure 19* are supported for the V_{CORE} core domain and an external supply.

Figure 19. System supply configurations



MSv41901V1

The different supply configurations are controlled through the LDOEN and BYPASS bits in *PWR control register 3 (PWR_CR3)* register according to *Table 30*.

Table 30. Supply configuration control

Supply configuration	LDOEN	BYPASS	Description
Default configuration	1	0	– V_{CORE} Power Domains are supplied from the LDO according to VOS.
LDO supply	1	0	– V_{CORE} Power Domains are supplied from the LDO according to VOS. – LDO power mode (Main, LP, Off) will follow system low-power modes.
LDO Bypass	0	1	– V_{CORE} supplied from external source – LDO bypassed, voltage monitoring still active.
Illegal	0	0	– Illegal combination, the default configuration is kept. (write data will be ignored).
	1	1	

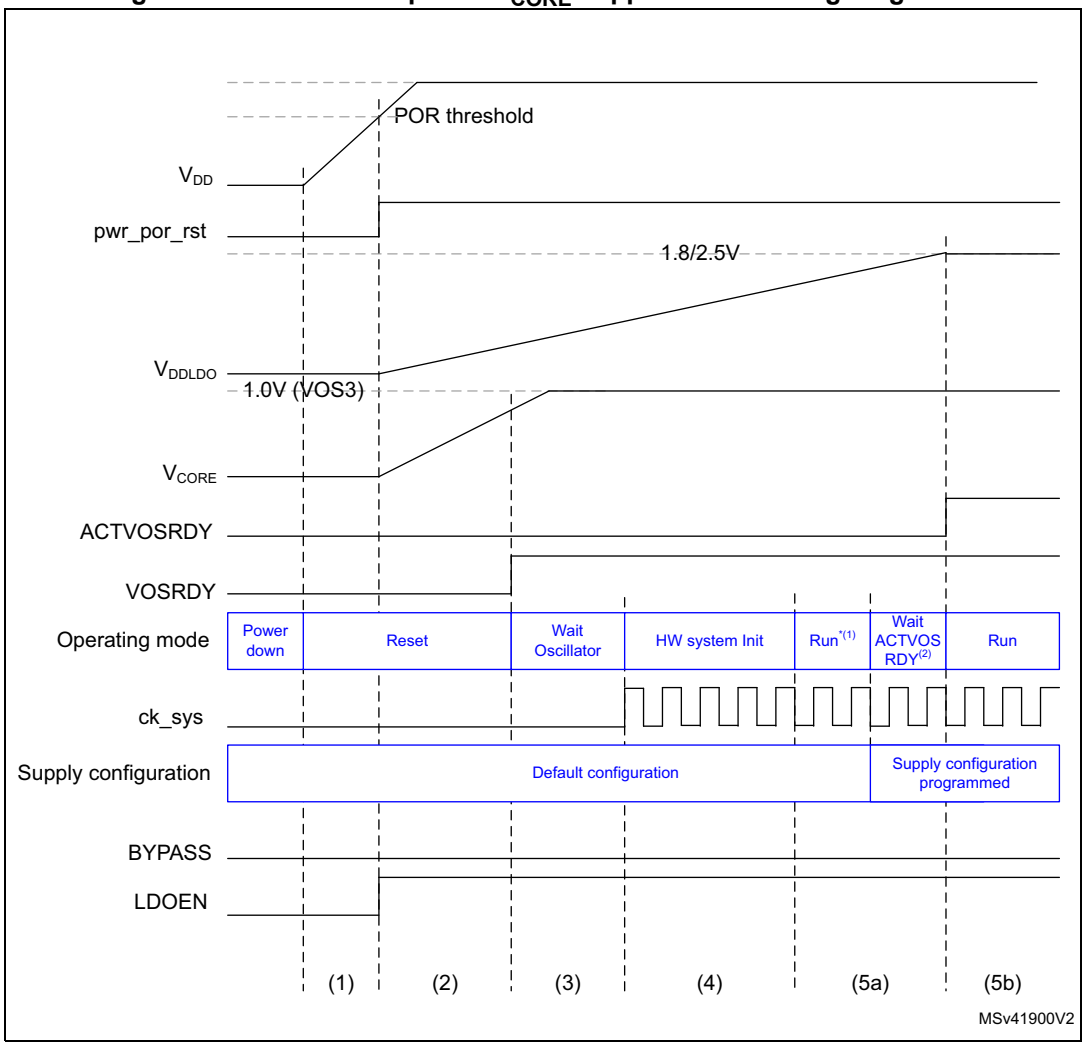
5.4.1 System supply startup

The system startup sequence from power-on in different supply configurations is the following (see *Figure 20* for LDO supply):

1. When the system is powered on, the POR monitors V_{DD} supply. Once V_{DD} is above the POR threshold level, the voltage regulator is enabled in the default supply configuration:
   – The Voltage converter output level is set at 1.0 V in accordance with the VOS3 level configured in *PWR D3 domain control register (PWR_D3CR)*.
2. The system is kept in reset mode as long as V_{CORE} is not correct.
3. Once V_{CORE} is correct, the system is taken out of reset and the HSI oscillator is enabled.
4. Once the oscillator is stable, the system is initialized: Flash memory and option bytes are loaded and the CPU starts in limited run mode (Run*).
5. The software shall then initialize the system including supply configuration programming in *PWR control register 3 (PWR_CR3)*. Once the supply configuration has been configured, the ACTVOSRDY bit in *PWR control status register 1 (PWR_CSR1)* shall be checked to guarantee valid voltage levels:
   a) As long as ACTVOSRDY indicates that voltage levels are invalid, the system is in Run* mode, write accesses to the RAMs are not permitted and VOS shall not be changed.
   b) Once ACTVOSRDY indicates that voltage levels are valid, the system is in normal Run mode, write accesses to RAMs are allowed and VOS can be changed.

The software has to program the supply configuration in *PWR control register 3 (PWR_CR3)*.

Figure 20. Device startup with V_{CORE} supplied from voltage regulator



1. In Run* mode, write operations to RAM are not allowed.
2. Write operations to RAM are allowed and VOS can be changed only when ACTVOSRDY is valid.

When exiting from Standby mode, the supply configuration is known by the system since the *PWR control register 3 (PWR_CR3)* register content is retained. However the software shall still wait for the ACTVOSRDY bit to be set in *PWR control status register 1 (PWR_CSR1)* to indicate V_{CORE} voltage levels are valid, before performing write accesses to RAM or changing VOS.

5.4.2 Core domain

The V_{CORE} core domain supply can be provided by the voltage regulator or by an external supply (V_{CAP}). V_{CORE} supplies all the digital circuitries except for the backup domain and the Standby circuitry. The V_{CORE} domain is split into 3 sections:

- D1 domain containing the CPU (Cortex®-M7), Flash memory and peripherals.
- D2 domain containing peripherals.
- D3 domain containing the system control, I/O logic and low-power peripherals.

When a system reset occurs, the voltage regulator is enabled and supplies V_{CORE}. This allows the system to start up in any supply configurations (see *Figure 19*).

After a system reset, the software shall configure the used supply configuration in *PWR control register 3 (PWR_CR3)* register before changing VOS in *PWR D3 domain control register (PWR_D3CR)* or the RCC ck_sys frequency. The different system supply configurations are controlled as shown in *Table 30*.

Voltage regulator

The embedded voltage regulator (LDO) requires external capacitors to be connected to V_{CAP} pins.

The voltage regulator provides three different operating modes: Main (MR), Low-power (LP) or Off. These modes will be used depending on the system operating modes (Run, Stop and Standby).

- Run mode

  The LDO regulator is in Main mode and provides full power to the V_{CORE} domain (core, memories and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS1, VOS2, and VOS3) that are configured through VOS bits in *PWR D3 domain control register (PWR_D3CR)*. The VOS voltage scaling allows optimizing the power consumption when the system is clocked below the maximum frequency. By default VOS3 is selected after system reset. VOS can be changed on-the-fly to adapt to the required system performance.

- Stop mode

  The voltage regulator supplies the V_{CORE} domain to retain the content of registers and internal memories.

  The regulator can be kept in Main mode to allow fast exit from Stop mode, or can be set in LP mode to obtain a lower V_{CORE} supply level and extend the exit-from-Stop latency. The regulator mode is selected through the SVOS and LPDS bits in *PWR control register 1 (PWR_CR1)*. Main mode and LP mode are allowed if SVOS3 voltage scaling is selected, while only LP mode is possible for SVOS4 and SVOS5 scaling. Due to a lower voltage level for SVOS4 and SVOS5 scaling, the Stop mode consumption can be further reduced.

- Standby mode

  The voltage regulator is OFF and the V_{CORE} domains are powered down. The content of the registers and memories is lost except for the Standby circuitry and the backup domain.

Note: For more details, refer to the voltage regulator section in the datasheets.

5.4.3 PWR external supply

When V_{CORE} is supplied from an external source, different operating modes can be used depending on the system operating modes (Run, Stop or Standby):

- In Run mode

  The external source supplies full power to the V_{CORE} domain (core, memories and digital peripherals). The external source output voltage is scalable through different voltage levels (VOS1, VOS2 and VOS3). The externally applied voltage level shall be

reflected in the VOS bits of PWR_D3CR register. The RAMs shall only be accessed for write operations when the external applied voltage level matches VOS settings.

- In Stop mode

  The external source supplies V_{CORE} domain to retain the content of registers and internal memories. The regulator can select a lower V_{CORE} supply level to reduce the consumption in Stop mode.

- In Standby mode

  The external source shall be switched OFF and the V_{CORE} domains powered down. The content of registers and memories is lost except for the Standby circuitry and the backup domain. The external source shall be switched ON when exiting Standby mode.

5.4.4 Backup domain

To retain the content of the backup domain (RTC, backup registers and backup RAM) when V_{DD} is turned off, V_{BAT} pin can be connected to an optional standby voltage which is supplied from a battery or from an another source.

The switching to V_{BAT} is controlled by the power-down reset embedded in the Reset block that monitors the V_{DD} supply.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the a startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (see the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the V_{BAT} pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the V_{BAT} pin.

When the V_{DD} supply is present, the backup domain is supplied from V_{DD}. This allows saving V_{BAT} power supply battery life time.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} through a 100 nF external ceramic capacitor.

When the V_{DD} supply is present and higher than the PDR threshold, the backup domain is supplied by V_{DD} and the following functions are available:

- PC14 and PC15 can be used either as GPIO or as LSE pins.
- PC13 can be used either as GPIO or as RTC_AF1 or RTC_TAMP1 pin assuming they have been configured by the RTC.
- PI8/RTC_TAMP2 and PC1/RTC_TAMP3 when they are configured by the RTC as tamper pins.

Note: Since the switch only sinks a limited amount of current, the use of PC1, PC13 to PC15 and PI8 GPIOs is restricted: only one I/O can be used as an output at a time, at a speed limited to 2 MHz with a maximum load of 30 pF. These I/Os must not be used as current sources (e.g. to drive an LED).

In V_{BAT} mode, when the V_{DD} supply is absent and a supply is present on V_{BAT}, the backup domain is supplied by V_{BAT} and the following functions are available:

- PC14 and PC15 can be used as LSE pins only.
- PC13 can be used as RTC_AF1 or RTC_TAMP1 pin assuming they have been configured by the RTC.
- PI8/RTC_TAMP2 and PC1/RTC_TAMP3 when they are configured by the RTC as tamper pins.

Accessing the backup domain

After reset, the backup domain (RTC registers and RTC backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, set the DBP bit in the *PWR control register 1 (PWR_CR1)*.

For more detail on RTC and backup RAM access, refer to *Section 7: Reset and Clock Control (RCC)*.

Backup RAM

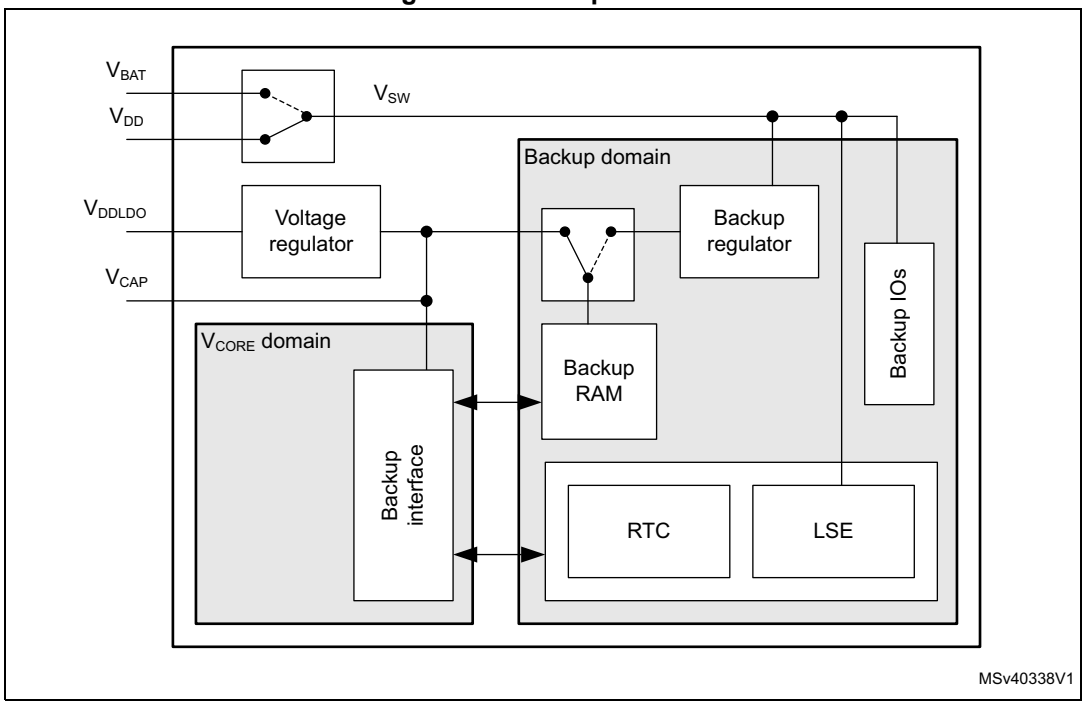
The backup domain includes 4 Kbytes of backup RAM accessible in 32-bit, 16-bit or 8-bit data mode. The backup RAM is supplied from the Backup regulator in the backup domain. When the Backup regulator is enabled through BREN bit in *PWR control register 2 (PWR_CR2)*, the backup RAM content is retained even in Standby and/or V_{BAT} mode (it can be considered as an internal EEPROM if V_{BAT} is always present.)

The Backup regulator can be ON or OFF depending whether the application needs the backup RAM function in Standby or V_{BAT} modes.

The backup RAM is not mass erased by an tamper event, instead it is read protected to prevent confidential data, such as cryptographic private key, from being accessed. To re-gain access to the backup RAM after a tamper event, the memory area needs to be first erased. The backup RAM can be erased:

- through the Flash interface when a protection level change from level 1 to level 0 is requested (refer to the description of Read protection (RDP) in the Flash programming manual).
- After a tamper event, by performing a dummy write with zero as data to the backup RAM.

Figure 21. Backup domain



5.4.5 V_{BAT} battery charging

When V_{DD} is present, the external battery connected to V_{BAT} can be charged through an internal resistance.

V_{BAT} charging can be performed either through a 5 kΩ resistor or through a 1.5 kΩ resistor, depending on the VBRS bit value in *PWR control register 3 (PWR_CR3)*.

The battery charging is enabled by setting the VBE bit in *PWR control register 3 (PWR_CR3)*. It is automatically disabled in V_{BAT} mode.

5.4.6 Analog supply

Separate V_{DDA} analog supply

The analog supply domain is powered by dedicated V_{DDA} and V_{SSA} pads that allow the supply to be filtered and shielded from noise on the PCB, thus improving ADC and DAC conversion accuracy:

- The analog supply voltage input is available on a separate V_{DDA} pin.
- An isolated supply ground connection is provided on V_{SSA} pin.

Analog reference voltage V_{REF+}/V_{REF-}

To achieve better accuracy low-voltage signals, the ADC and DAC also have a separate reference voltage, available on V_{REF+} pin. The user can connect a separate external reference voltage on V_{REF+}.

The V_{REF+} controls the highest voltage, represented by the full scale value, the lower voltage reference (V_{REF-}) being connected to V_{SSA}.

When enabled by ENVR bit in the VREFBUF control and status register (see *Section 26: Voltage reference buffer (VREFBUF)*), V_{REF+} is provided from the internal voltage reference buffer. The internal voltage reference buffer can also deliver a reference voltage to external components through V_{REF+}/V_{REF-} pins.

When the internal voltage reference buffer is disabled by ENVR, V_{REF+} s delivered by an independent external reference supply voltage.

5.4.7 USB regulator

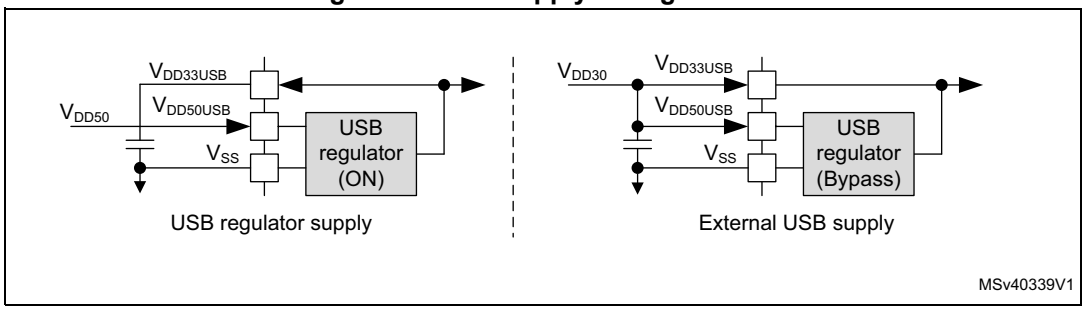
The USB transceivers are supplied from a dedicated $V_{DD33USB}$ supply that can be provided either by the integrated USB regulator, or by an external USB supply.

When enabled by USBREGEN bit in *PWR control register 3 (PWR_CR3)*, the $V_{DD33USB}$ is provided from the USB regulator. Before using $V_{DD33USB}$, check that it is available by monitoring USB33RDY bit in *PWR control register 3 (PWR_CR3)*. The $V_{DD33USB}$ supply level detector shall be enabled through USB33DEN bit in PWR_CR3 register.

When the USB regulator is disabled through USBREGEN bit, $V_{DD33USB}$ can be provided from an external supply. In this case $V_{DD33USB}$ and $V_{DD50USB}$ shall be connected together. The $V_{DD33USB}$ supply level detector must be enabled through USB33DEN bit in PWR_CR3 register before using the USB transceivers.

For more information on the USB regulator (see *Section 56: USB on-the-go high-speed (OTG_HS)*).

Figure 22. USB supply configurations



5.5 Power supply supervision

Power supply level monitoring is available on the following supplies:

- V_{DD} via POR/PDR (see *Section 5.5.1*), BOR (see *Section 5.5.2*) and PVD monitor (see *Section 5.5.3*)
- V_{DDA} via AVD monitor (see *Section 5.5.4*)
- V_{BAT} via VBAT threshold (see *Section 5.5.5*)
- V_{SW} via rst_vsw, which keeps V_{SW} domain in Reset mode as long as the level is not OK.
- V_{BKP} via a BRRDY bit in *PWR control register 2 (PWR_CR2)*.
- $V_{DD33USB}$ via USBRDY bit in *PWR control register 3 (PWR_CR3)*.

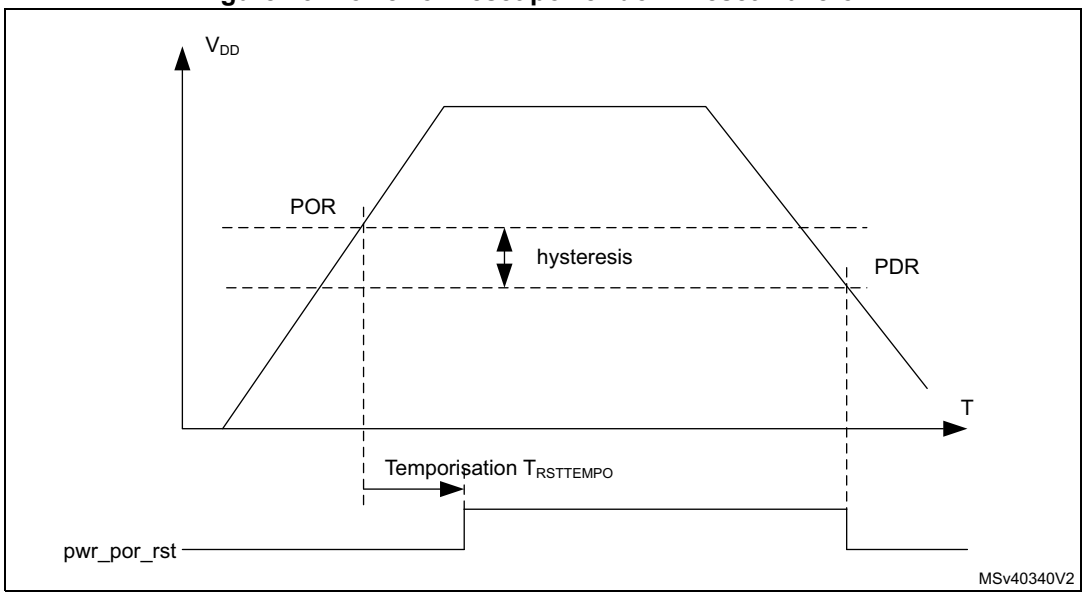
5.5.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in Reset mode when V_{DD} is below a specified V_{POR} threshold, without the need for an external reset circuit. Once the V_{DD} supply level is above the V_{POR} threshold, the system is taken out of reset (see *Figure 23*). For more details concerning the power-on/power-down reset threshold, refer to the electrical characteristics section of the datasheets.

The PDR can be enabled/disabled by the device PDR_ON input pin.

Figure 23. Power-on reset/power-down reset waveform



1. For thresholds and hysteresis values, please refer to the datasheets.

5.5.2 Brownout reset (BOR)

During power-on, the Brownout reset (BOR) keeps the system under reset until the V_{DD} supply voltage reaches the specified V_{BOR} threshold.

The V_{BOR} threshold is configured through system option bytes. By default, BOR is OFF. The following programmable V_{BOR} thresholds can be selected:

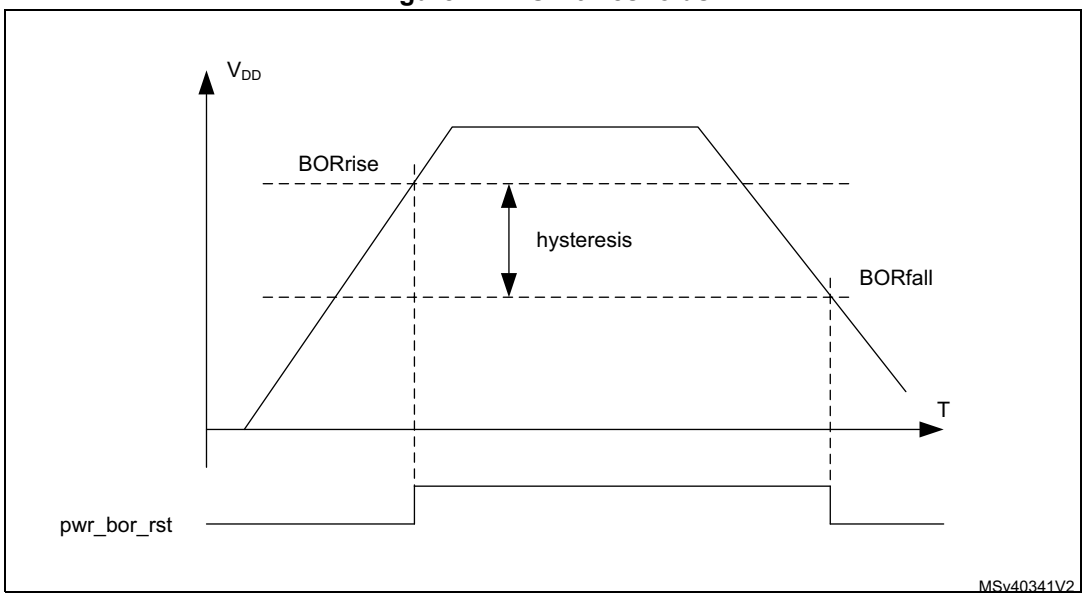
- BOR OFF (V_{BOR0})
- BOR Level 1 (V_{BOR1})
- BOR Level 2 (V_{BOR2})
- BOR Level 3 (V_{BOR3})

For more details on the brown-out reset thresholds, refer to the section “Electrical characteristics” of the product datasheets.

A system reset is generated when the BOR is enabled and V_{DD} supply voltage drops below the selected V_{BOR} threshold.

BOR can be disabled by programming the system option bytes. To disable the BOR function, V_{DD} must have been higher than V_{BOR0} to start the system option byte programming sequence. The power-down is then monitored by the PDR (see *Section 5.5.1*).

Figure 24. BOR thresholds



1. For thresholds and hysteresis values, please refer to the datasheets.

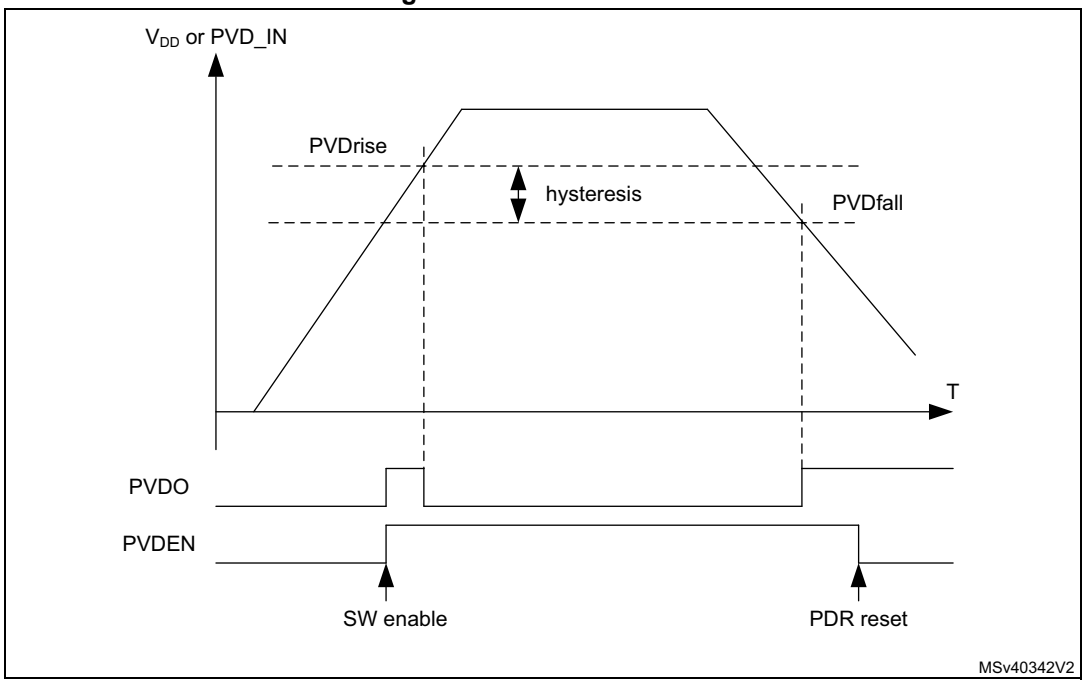
5.5.3 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR control register 1 (PWR_CR1)*. The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case PVD_IN voltage is compared to the internal V_{REFINT} level.

The PVD is enabled by setting the PVDE bit in *PWR control register 1 (PWR_CR1)*.

A PVDO flag is available in the *PWR control status register 1 (PWR_CSR1)* to indicate if V_{DD} or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, assuming it has been enabled through the EXTI registers. The pwr_pvd_wkup output interrupt can be generated when V_{DD} or PVD_IN voltage drops below the PVD threshold and/or when V_{DD} or PVD_IN voltage rises above the PVD threshold depending on EXTI rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

Figure 25. PVD thresholds



1. For thresholds and hysteresis values, please refer to the datasheets.

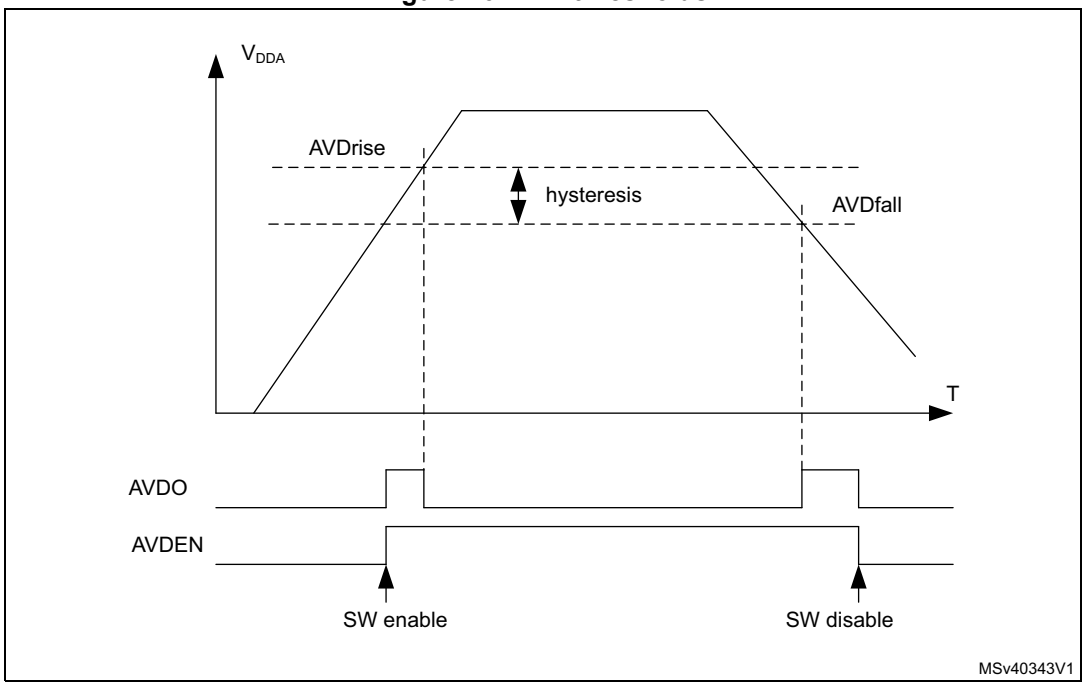
5.5.4 Analog voltage detector (AVD)

The AVD can be used to monitor the V_{DDA} supply by comparing it to a threshold selected by the ALS[1:0] bits in the *PWR control register 1 (PWR_CR1)*.

The AVD is enabled by setting the AVDEN bit in *PWR control register 1 (PWR_CR1)*.

An AVDO flag is available in the *PWR control status register 1 (PWR_CSR1)* to indicate whether V_{DDA} is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The pwr_avd_wkup interrupt can be generated when V_{DDA} drops below the AVD threshold and/or when V_{DDA} rises above the AVD threshold depending on EXTI rising/falling edge configuration. As an example the service routine could indicate when the V_{DDA} supply drops below a minimum level.

Figure 26. AVD thresholds



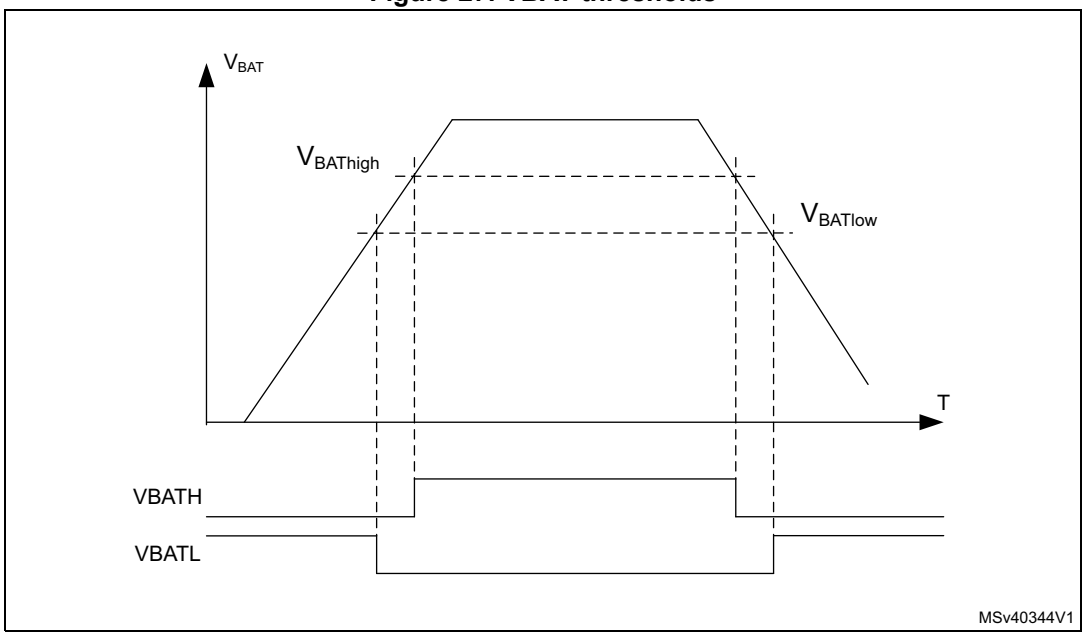
1. For thresholds and hysteresis values, please refer to the datasheets.

5.5.5 Battery voltage thresholds

The V_{BAT} battery voltage supply can be monitored by comparing it with two threshold levels: $V_{BAThigh}$ and V_{BATlow}. VBATH and VBATL flags in the *PWR control register 2 (PWR_CR2)*, indicate if V_{BAT} is higher or lower than the threshold. The V_{BAT} supply monitoring can be enabled/disabled via MONEN bit in *PWR control register 2 (PWR_CR2)*. When it is enabled, the battery voltage thresholds increase power consumption. As an example the levels could be used to trigger a routine to perform emergency saving tasks.

VBATH and VBATL wakeup interrupts are available on the RTC tamper signals (see *Section 45: Real-time clock (RTC)*)

Figure 27. VBAT thresholds



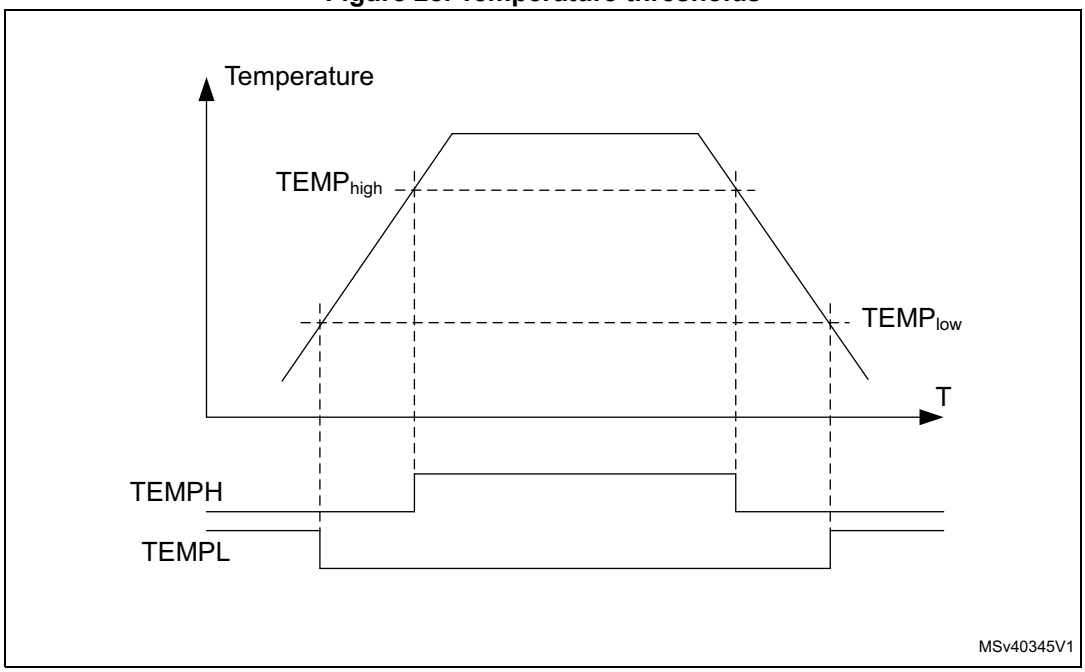
1. For thresholds and hysteresis values, please refer to the datasheets.

5.5.6 Temperature thresholds

The junction temperature can be monitored by comparing it with two threshold levels, $TEMP_{high}$ and $TEMP_{low}$. TEMPH and TEMPL flags, in the *PWR control register 2 (PWR_CR2)*, indicate whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via MONEN bit in *PWR control register 2 (PWR_CR2)*. When enabled, the temperature thresholds increase power consumption. As an example the levels could be used to trigger a routine to perform temperature control tasks.

TEMPH and TEMPL wakeup interrupts are available on the RTC tamper signals (see *Section 45: Real-time clock (RTC)*).

Figure 28. Temperature thresholds



1. For thresholds and hysteresis values, please refer to the datasheets.

5.6 Power management

The power management block controls the V_{CORE} supply in accordance with the system operation modes (see *Section 5.6.1*).

The V_{CORE} domain is split into the following power domains.

- D1 domain containing some peripherals and the Cortex®-M7 Core (CPU).
- D2 domain containing a large part of the peripherals.
- D3 domain containing some peripherals and the system control.

The D1, D2 and system D3 power domains can operate in one of the following operating modes:

- DRun/Run/Run* (power ON, clock ON)
- DStop/Stop (power ON, clock OFF)
- DStandby/Standby (Power OFF, clock OFF).

The operating modes for D1 domain and D2 domain are independent. However system D3 domain power modes depend on D1 and D2 domain modes:

- For system D3 domain to operate in Stop mode, both D1 and D2 domains must be in DStop or DStandby mode.
- For system D3 domain to operate in Standby mode, both D1 and D2 domains must be in DStandby too.

D1, D2 and system D3 domains are supplied from a single regulator at a common V_{CORE} level. The V_{CORE} supply level follows the system operating mode (Run, Stop, Standby). The D1 domain and/or D2 domain supply can be powered down individually when the domains are in DStandby mode.

The following voltage scaling features allow controlling the power with respect to the required system performance (see *Section 5.6.2: Voltage scaling*):

- To obtain a given system performance, the corresponding voltage scaling shall be set in accordance with the system clock frequency. To do this, configure the VOS bits to the Run mode voltage scaling.
- To obtain the best trade-off between power consumption and exit-from-Stop mode latency, configure the SVOS bits to Stop mode voltage scaling.

5.6.1 Operating modes

Several system operating modes are available to tune the system according to the performance required, i.e. when the CPU does not need to execute code and is waiting for an external event. It is up to the user to select the operating mode that gives the best compromise between low power consumption, short startup time and available wakeup sources.

The operating modes allow controlling the clock distribution to the different system blocks and powering them. The system operating mode is driven by the CPU subsystem, D2 domain and system D3 autonomous wakeup. The CPU subsystem can include multiple domains depending on its peripheral allocation (see *Section 7.5.11: Peripheral clock gating control*).

The following operating modes are available for the different system blocks (see *Table 31*):

- CPU subsystem modes:
  - **CRun**

    CPU and CPU subsystem peripheral(s) allocated via RCC PERxEN bits are clocked.
  - **CSleep**:

    The CPU clocks is stalled and the CPU subsystem allocated peripheral(s) clock operate according to RCC PERxLPEN.
  - **CStop**:

    CPU and CPU subsystem peripheral(s) clocks are stalled.
- D1 domain mode:
  - DRun

    The domain bus matrix is clocked. The CPU subsystem operates in CRun or CSleep mode.
  - DStop

    The domain bus matrix clock is stalled.

    The CPU subsystem operates in CStop mode and the PDDS_D1[(a)] bit selects DStop mode.
  - DStandby

    The domain is powered down.

    The CPU subsystem operates in CStop mode and the PDDS_D1 bit selects DStandby mode.
- D2 domain mode:
  - DRun

    The domain bus matrix is clocked.

    The CPU subsystem has an allocated peripheral in the D2 domain and the CPU subsystem operates in CRun or CSleep mode.

a. The PDDS_Dn bits belong to *PWR CPU control register (PWR_CPUCR)*.

– DStop

The domain bus matrix clock is stalled.

The CPU subsystem has no peripherals allocated in the D2 domain and PDDS_D2(a) bit selects DStop mode,

or

the CPU subsystem has an allocated peripheral in D2 domain, the CPU subsystem operates in CStop mode and PDDS_D2 bit selects DStop mode.

– DStandby

The domain is powered down.

The CPU subsystem has no peripherals allocated in the D2 domain and PDDS_D2 bit selects DStandby mode,

or

the CPU subsystem has an allocated peripheral in the D2 domain, the CPU subsystem operates in CStop mode and PDDS_D2 bit selects DStandby mode.

- System /D3 domain modes
  - **Run/Run***

    The system clock and D3 domain bus matrix clock are running:
    - The CPU subsystem is in CRun or CSleep mode
    or
    - A wakeup signal is active. (i.e. System D3 autonomous mode)

    The Run* mode is entered after a POR reset and a wakeup from Standby. In Run* mode, the performance is limited and the system supply configuration shall be programmed in *PWR control register 3 (PWR_CR3)*. The system enters Run mode only when the ACTVOSRDY bit in *PWR control status register 1 (PWR_CSR1)* is set to 1.
  - **Stop**

    The system clock and D3 domain bus matrix clock is stalled:
    - The CPU subsystem is in CStop mode.
    and
    - all wakeup signals are inactive.
    and
    - At least one PDDS_Dn(a) bit for any domain select Stop mode.
  - **Standby**

    The system is powered down:
    - The CPU subsystem is in CStop mode
    and
    - all wakeup signals are inactive.
    and
    - All PDDS_Dn(a) bits for all domains select Standby mode.

In Run mode, power consumption can be reduced by one of the following means:

- Lowering the system performance by slowing down the system clocks and reducing the V_{CORE} supply level through VOS voltage scaling bits.
- Gating the clocks to the APBx and AHBx peripherals when they are not used, through PERxEN bits.

Table 31. Low-power mode summary

System	Domain	CPU	Entry	Wakeup	Sys-oscillator	System clk	Domain bus matrix clk	Peripheral clk	CPU clk	Voltage regulator	Domain supply
Run	DRun(1)	CRun	-	-	ON	ON	ON	ON	ON	ON	ON
		CSleep	WFI or return from ISR or WFE	Any interrupt or event				ON/OFF(2)	OFF		
	DStop(3)	CStop	SLEEPDEEP bit + WFI or return from ISR or WFE	Any EXTI interrupt or event			OFF	ON/OFF(4)			OFF
	DStandby(3)										
Stop(5)	DStop(3)		SLEEPDEEP bit + WFI or return from ISR or WFE or Wakeup source cleared(6)		ON/OFF(7)	OFF		OFF			ON
	DStandby(3)										OFF
Standby(8)	DStandby(3)		All PDDS_Dn bit + SLEEPDEEP bit + WFI or return from ISR or WFE or Wakeup source cleared(6)	WKUP pins rising or falling edge, RTC alarm (Alarm A or Alarm B), RTC Wakeup event, RTC tamper events, RTC time stamp event, external reset in NRST pin, IWDG reset	OFF					OFF	

1. The CPU subsystem has an allocated peripheral in the D2 domain and operates in CRun or CSleep mode.
2. The CPU subsystem peripherals that have a PERxLPEN bit will operate accordingly.
3. If the CPU subsystem has an allocated peripheral in the D2 domain, it must operate in CStop mode.
4. The CPU subsystem peripherals that have a PERxAMEN bit will operate accordingly.
5. All domains need to be in DStop Or DStandby.
6. When the CPU is in CStop and D3 domain in autonomous mode, the last EXTI Wakeup source is cleared.
7. When the system oscillator HSI or CSI is used, the state is controlled by HSIKERON and CSIKERON, otherwise the system oscillator is OFF.
8. All domains are in DStandby mode.

5.6.2 Voltage scaling

The D1, D2, and D3 domains are supplied from a single voltage regulator supporting voltage scaling with the following features:

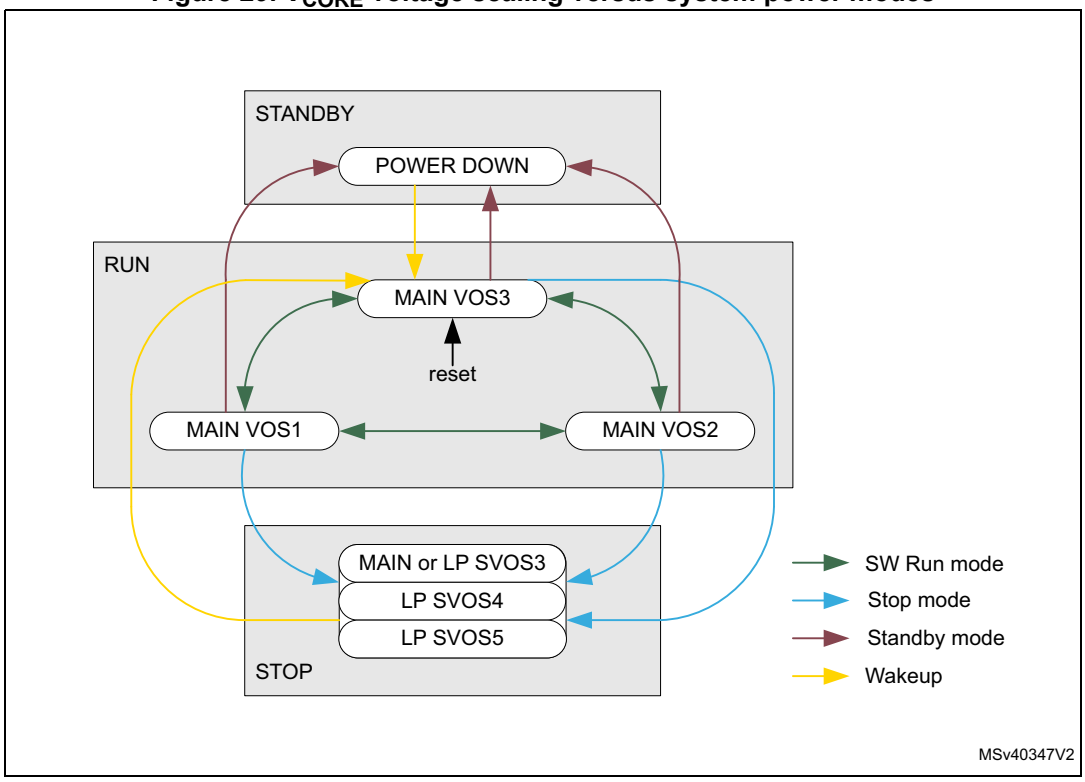
- Run mode voltage scaling
  - VOS1: Scale 1
  - VOS2: Scale 2
  - VOS3: Scale 3
- Stop mode voltage scaling
  - SVOS3: Scale 3
  - LP-SVOS4: Scale 4
  - LP-SVOS5: Scale 5

For more details on voltage scaling values, refer to the product datasheets.

After reset, the system starts on the lowest Run mode voltage scaling (VOS3). The voltage scaling can then be changed on-the-fly by software by programming VOS bits in *PWR D3 domain control register (PWR_D3CR)* according to the required system performance. When exiting from Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS3 value.

Before entering Stop mode, the software can preselect the SVOS level in *PWR control register 1 (PWR_CR1)*. The Stop mode voltage scaling for SVOS4 and SVOS5 also sets the voltage regulator in Low-power (LP) mode to further reduce power consumption. When preselecting SVOS3, the use of the voltage regulator low-power mode (LP) can be selected by LPDS register bit.

Figure 29. V_{CORE} voltage scaling versus system power modes



5.6.3 Power control modes

The power control block handles the V_{CORE} supply for system Run, Stop and Standby modes.

The system operating mode depends on the CPU subsystem modes (CRun, CSleep, CStop), on the domain modes (DRun, DStop, DStandby), and on the system D3 autonomous wakeup:

- In Run mode, V_{CORE} is defined by the VOS voltage scaling.
  The CPU subsystem is in CRun or CSleep or an EXTI wakeup is active.
- In Stop mode, V_{CORE} is defined by the SVOS voltage scaling.
  The CPU subsystem is in CStop mode and all EXTI wakeups are inactive. The D1 domain and D2 domain are either in DStop or DStandby mode.
- In Standby mode, V_{CORE} supply is switched off.
  The CPU subsystem is in CStop mode and all EXTI wakeups are inactive. The D1 domain and D2 domain are both in DStandby mode.

The domain operating mode can depend on the CPU subsystem when peripherals are allocated in the corresponding domain. The domain mode selection between DStop and DStandby is configured via domain dedicated PDDS_Dn bits in *PWR CPU control register (PWR_CPUCR)*. The CPU can choose to keep a domain in DStop, or allow a domain to enter DStandby mode.

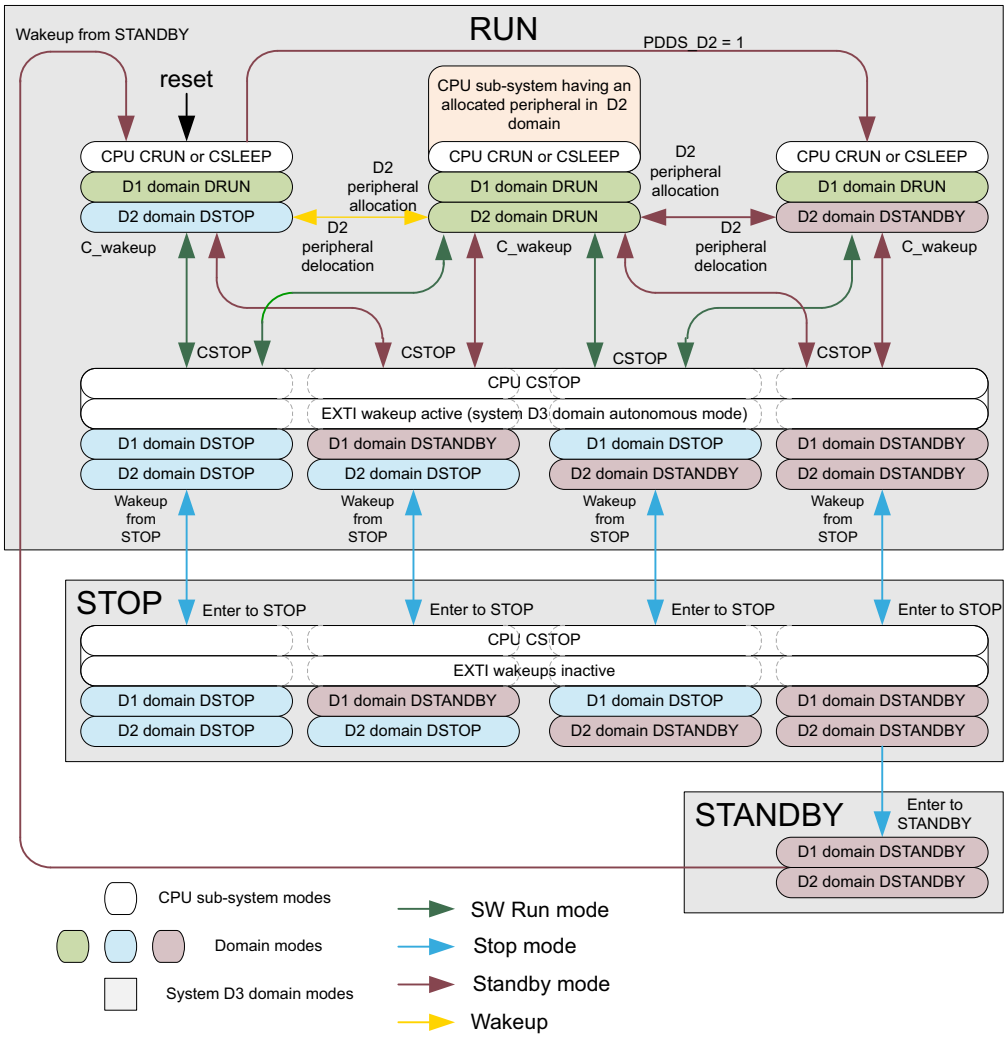
If a domain is in DStandby mode, the corresponding power is switched off.

All the domains can be configured for the system mode (Stop or Standby) through PDDS_Dn bits in *PWR CPU control register (PWR_CPUCR)*. The system enters Standby only when all PDDS_Dn bits for all domains have allowed it.

Table 32. PDDS_Dn low-power mode control

PWR_CPUCR			D1 mode	D2 mode	D3 mode
PDDS_D1	PDDS_D2	PDDS_D3			
0	x	x	DStop	any	Run or Stop
1			DStandby	any	any
x	0		any	DStop	Run or Stop
	1		any	DStandby	any
at least one = 0			DStop or DStandby	DStop or DStandby	Stop
1	1	1	DStandby	DStandby	Standby

Figure 30. Power control modes detailed state diagram



After a system reset, the CPU is in CRun mode.

Power control state transitions are initiated by the following events:

- CPU going to CStop mode (state transitions in Run mode are marked in green and red)
  - Green transitions: CPU wakes up as from CSleep.
  - Red transitions: CPU wakes up with domain reset. The SBF_Dn is set.
- Allocating or de-locating a peripheral in a domain (state transitions in Run mode are marked in orange and red)
  - Orange transitions: the domain wakes up from DStop
  - Red transitions: the domain wakes up from DStandby. The SBF_Dn is set.
- The system enters or exits from Stop mode (state transitions marked in blue)
  - Blue transitions the system wakes up from Stop mode. The STOPF is set.
- The system enters or exits from Standby mode (state transitions in Stop and Standby mode are marked in red).
  - When exiting from Standby mode, the SBF is set.

When a domain exits from DStandby, the domain peripherals are reset, while the domain SBF_Dn bit is set (state transitions causing a domain reset are marked in red).

Table 33 shows the flags that indicate from which mode the domain/system exits. The CPU features a set of flags which can be read from *PWR CPU control register (PWR_CPUCR)*.

Table 33. Low-power exit mode flags

System mode	D1 domain mode	D2 domain mode	SBF_D1	SBF_D2	SBF	STOPF	Comment
Run	DRun or DStop	DRun or DStop	0	0	0	0	D1, D2 and system contents retained
Run	DStandby	DStop	1	0	0	0	D1 contents lost, D2 and system contents retained
Run	DRun or DStop	DStandby	0	1	0	0	D2 contents lost, D1 and system contents retained
Run	DStandby	DStandby	1	1	0	0	D1 and D2 contents lost, system contents retained
Stop	DStop	DStop	0	0	0	1	D1, D2 and system contents retained, clock system reset.
Stop	DStandby	DStop	1	0	0	1	D1 contents lost, D2 and system contents retained, clock system reset
Stop	DStop	DStandby	0	1	0	1	D2 contents lost, D1 and system contents retained, clock system reset
Stop	DStandby	DStandby	1	1	0	1	D1 and D2 contents lost, system contents retained, clock system reset
Standby	DStandby	DStandby	0(1)	0(1)	1	0	D1, D2 and system contents lost

1. When returning from Standby, the SBF_D1 and SBF_D2 reflect the reset value.

5.6.4 Power management examples

- *Figure 31* shows V_{CORE} voltage scaling behavior in Run mode.
- *Figure 32* shows V_{CORE} voltage scaling behavior in Stop mode.
- *Figure 33* shows V_{CORE} voltage regulator and voltage scaling behavior in Standby mode.
- *Figure 34* shows V_{CORE} voltage scaling behavior in Run mode with D1 and D2 domains are in DStandby mode

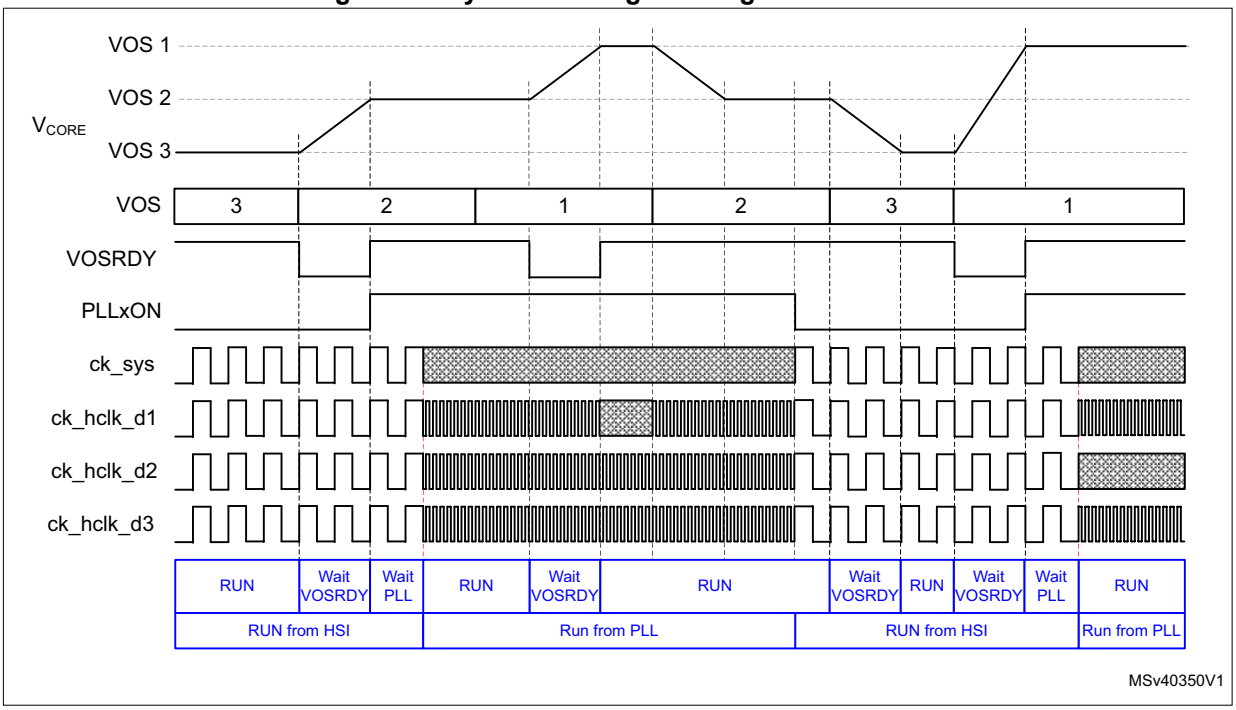
Example of V_{CORE} voltage scaling behavior in Run mode

Figure 31 illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS3.
2. The system performance is first increased to a medium-speed clock from the PLL with voltage scaling VOS2. To do this:
   a) Program the voltage scaling to VOS2.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The system performance is then increased to high-speed clock from the PLL with voltage scaling VOS1. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency.
4. The system performance is then reduced to a medium-speed clock with voltage scaling VOS2. To do this:
   a) First decrease the system frequency.
   b) Then decrease the voltage scaling to VOS2.
5. The next step is to reduce the system performance to HSI clock with voltage scaling VOS3. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS3.
6. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS shall be set accordingly. otherwise the system might be unreliable.

Figure 31. Dynamic voltage scaling in Run mode



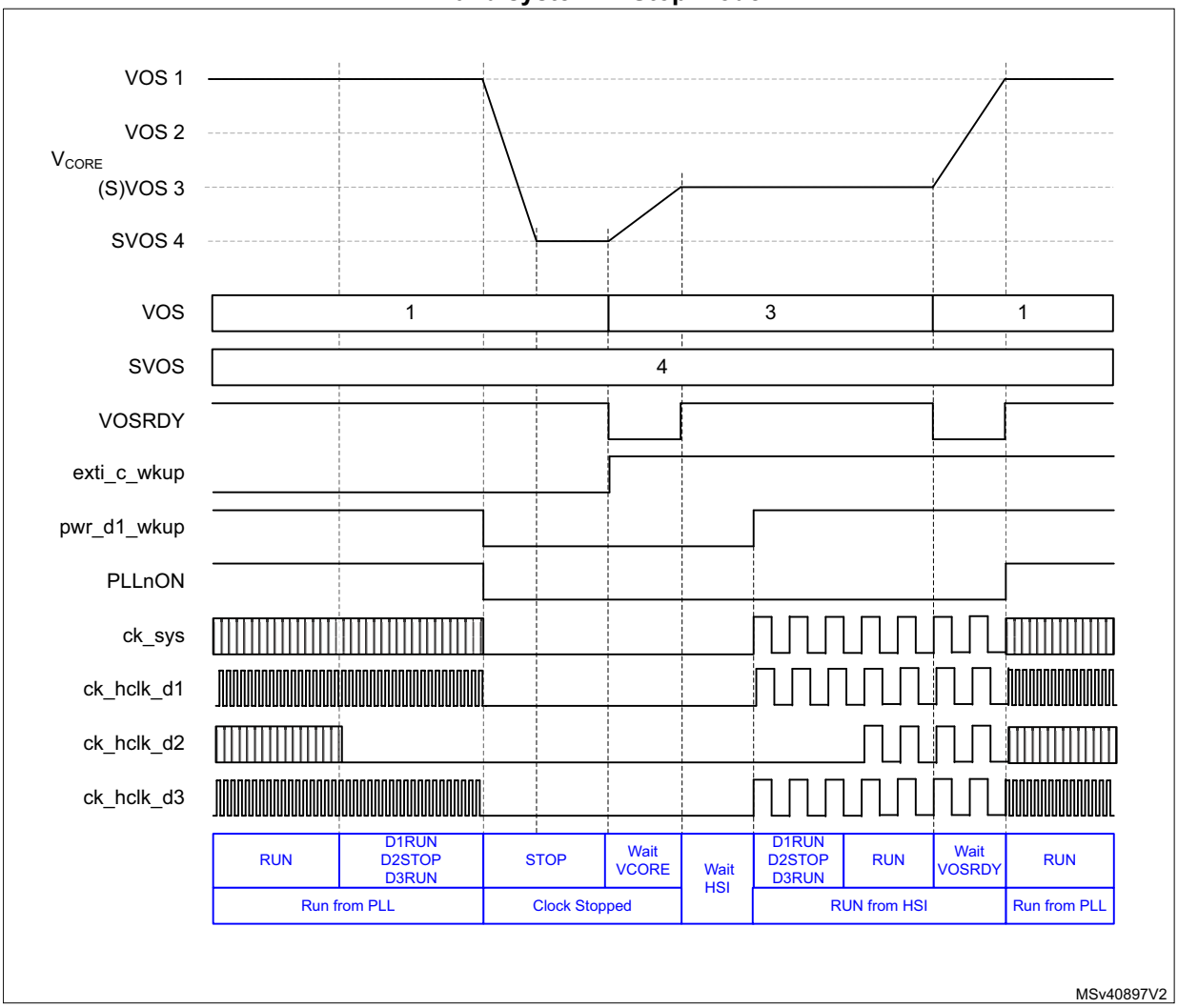
1. The status of the register bits at each step is shown in blue.

Example of V_{CORE} voltage scaling behavior in Stop mode

Figure 32 illustrates the following system operation sequence example:

1. The system is running from the PLL in high-performance mode (VOS1 voltage scaling).
2. The CPU subsystem deallocates all the peripheral in the D2 domain that will first enter DStop mode. D2 system clock is stopped. The system still provides the high-performance system clock, hence the voltage scaling shall stay at VOS1 level.
3. In a second step, the CPU subsystem enters CStop mode, D1 domain enters DStop mode and the system enters Stop mode. The system clock is stopped and the hardware lowers the voltage scaling to the software preselected SVOS4 level.
4. The CPU subsystem is then woken up. The system exits from Stop mode, the D1 domain exits from DStop mode and the CPU subsystem exits from CStop mode. The hardware then sets the voltage scaling to VOS3 level and waits for the requested supply level to be reached before enabling the HSI clock. Once the HSI clock is stable, the system clock and the D1 system clock are enabled.
5. The CPU subsystem allocates a peripheral in the D2 domain. The D2 system clock is enabled.
6. The system performance is then increased. To do this:
   a) The software first sets the voltage scaling to VOS1.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, the clock frequency can be increased by enabling the PLL.
   c) Once the PLL is locked, the system clock can be switched.

Figure 32. Dynamic voltage scaling behavior with D1, D2 and system in Stop mode



1. The status of the register bits at each step is shown in blue.

Example of V_{CORE} voltage regulator and voltage scaling behavior in Standby mode

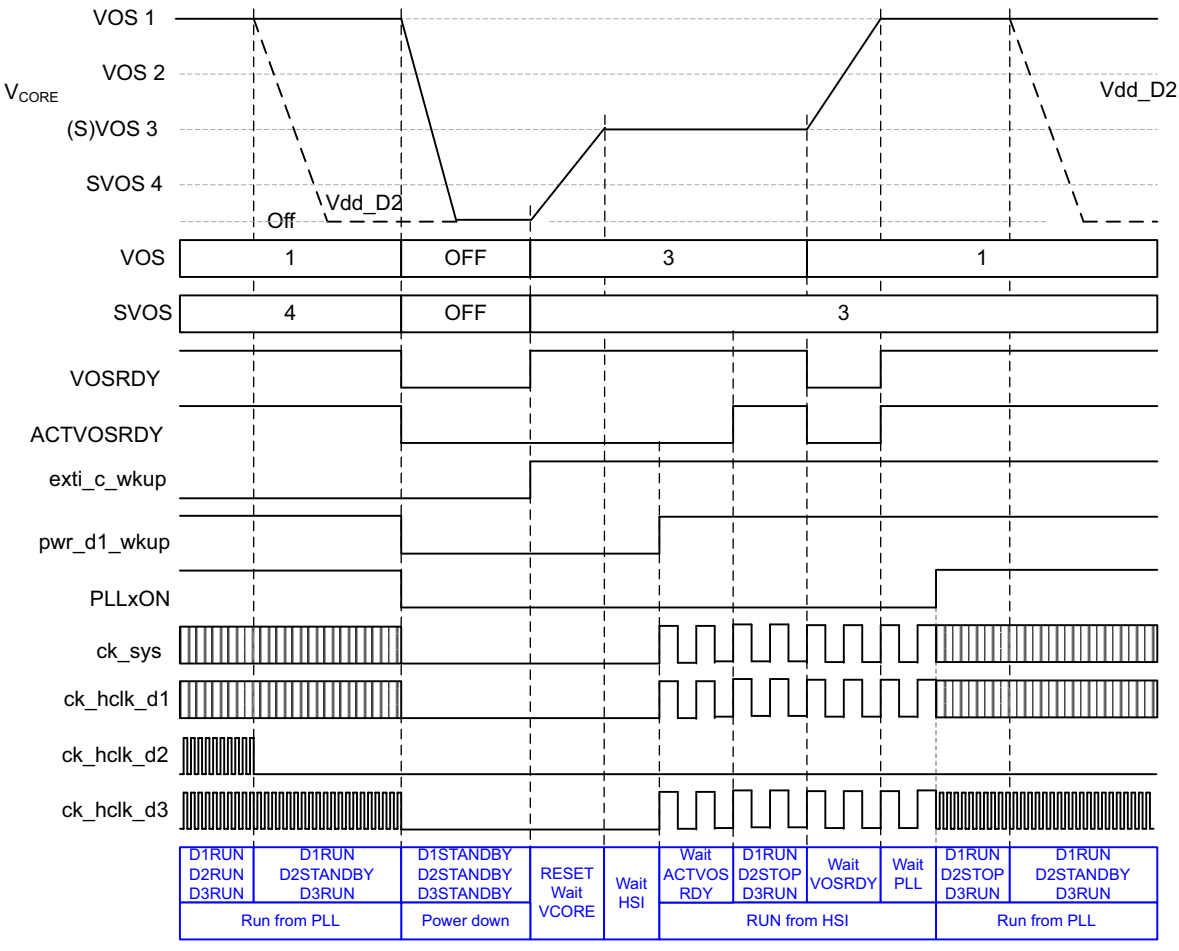
Figure 33 illustrates the following system operation sequence example:

1. The system is running from the PLL in high-performance mode (VOS1 voltage scaling).
2. The CPU subsystem deallocates all the peripherals in the D2 domain that will first enter DStandby mode.The D2 domain bus matrix clock is stopped and the power is switched off. The system performance is unchanged hence the voltage scaling does not change.
3. The CPU subsystem then enters to CStop mode, D1 domain enters DStandby mode and the system enters Standby mode. The system clock is stopped and the voltage regulator switched off.
4. The system is then woken up by a wakeup source. The system exits from Standby mode. The hardware sets the voltage scaling to the default VOS3 level and waits for the requested supply level to be reached before enabling the default HSI oscillator. Once the HSI clock is stable, the system clock and D1 subsystem clock are enabled.

Since there are no allocated peripherals in the D2 domain, this domain remains in DStop mode. The software shall then check the ACTVOSRDY is valid before changing the system performance.

5. In a next step, increase the system performance. To do this:
   a) The software first increases the voltage scaling to VOS1 level
   b) Before enabling the PLL, it waits for the requested supply level to be reached by monitoring VOSRDY bit.
   c) Once the PLL is locked, the system clock can be switched.
6. The CPU subsystem puts the D2 domain in DStandby mode.

Figure 33. Dynamic Voltage Scaling D1, D2, system Standby mode



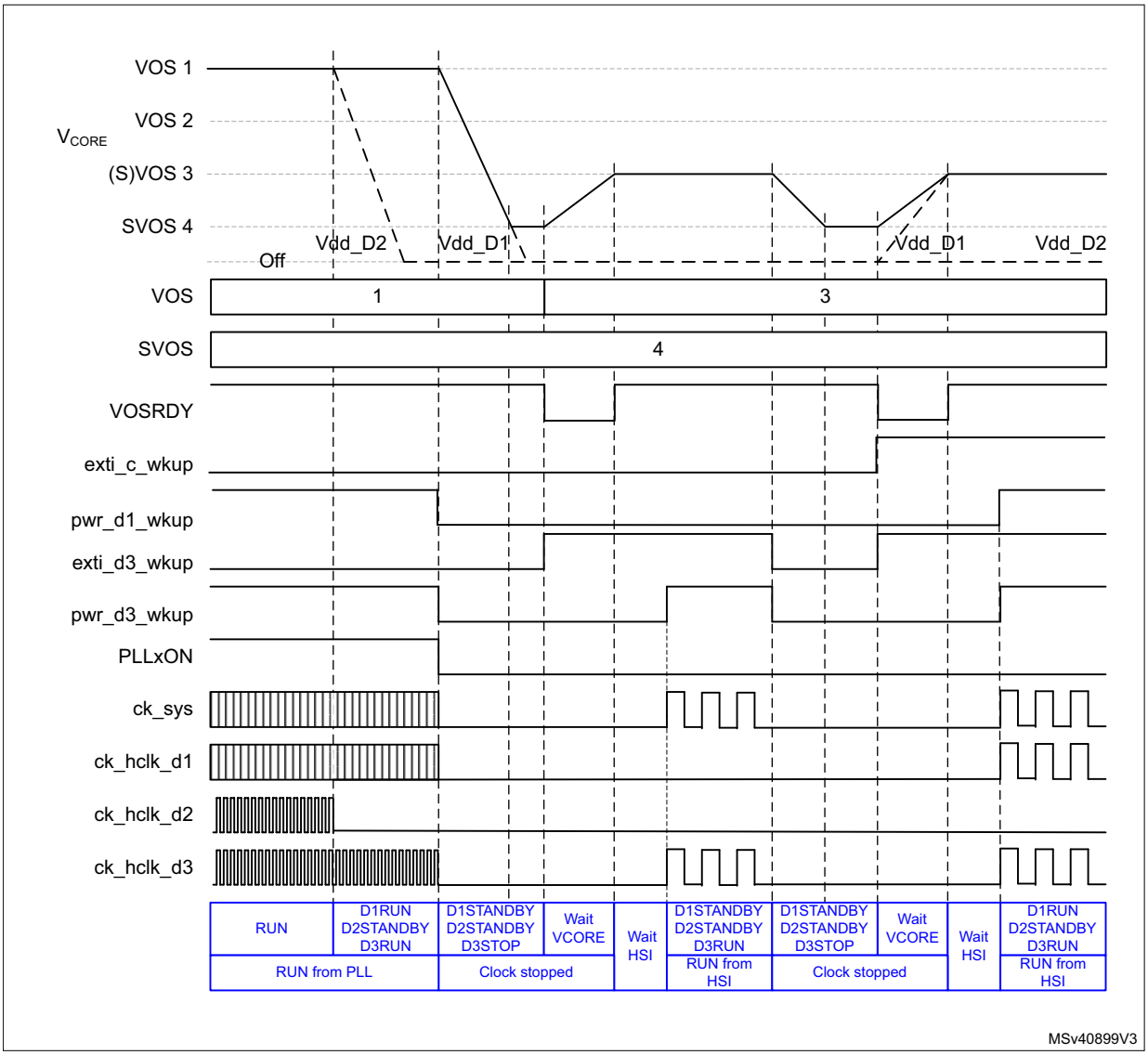
1. The status of the register bits at each step is shown in blue.

Example of V_{CORE} voltage scaling behavior in Run mode with D1 and D2 domains in DStandby mode

Figure 34 illustrates the following system operation sequence example:

1. The system is running from the PLL with system in high performance mode (VOS1 voltage scaling).
2. The CPU subsystem deallocates all the peripherals in the D2 domain that will first enter DStandby mode. The D2 domain bus matrix clock is stopped and its power switched off. The system performance is unchanged hence the voltage scaling does not change.
3. The CPU subsystem then enters CStop mode and the D1 domain enters DStandby mode. The D1 domain bus matrix clock is stopped and its power switched off. At the same time the system enters Stop mode. The system clock is stopped and the hardware lowers the voltage scaling to the software preselected SVOS4 level.
4. The system is then woken up by a D3 autonomous mode wakeup event. The system exits from Stop mode. The hardware sets the voltage scaling to the default VOS3 level and waits for the requested supply level to be reached before enabling the HSI clock. Once the HSI clock is stable, the system clock is enabled. The system is running in D3 autonomous mode.
5. The D3 autonomous mode wakeup source is then cleared, causing the system to enter Stop mode. The system clock is stopped and the voltage scaling is lowered to the software preselected SVOS4 level.
6. The CPU subsystem is then woken up. The system exits from Stop mode, the D1 domain exits from DStandby mode and the CPU subsystem exits from CStop mode. The hardware sets the voltage scaling to the default VOS3 level and waits for the requested supply level to be reached before enabling the default HSI oscillator. Once the HSI clock is stable, the system clock and the D1 subsystem clock are enabled. The D2 domain remains in DStandby mode.

Figure 34. Dynamic voltage scaling behavior with D1 and D2 in DStandby mode and D3 in autonomous mode



1. The status of the register bits at each step is shown in blue.

5.7 Low-power modes

Several low-power modes are available to save power when the CPU does not need to execute code (i.e. when waiting for an external event). It is up to the user application to select the mode that gives the best compromise between low power consumption, short startup time and available wakeup sources:

- Slowing down system clocks (see *Section 7.5.6: System clock (sys_ck)*)
- Controlling individual peripheral clocks (see *Section 7.5.11: Peripheral clock gating control*)
- Low-power modes
  - CSleep (CPU clock stopped)
  - CStop (CPU subsystem clock stopped)
  - DStop (Domain bus matrix clock stopped)
  - Stop (System clock stopped)
  - DStandby (Domain powered down)
  - Standby (System powered down)

5.7.1 Slowing down system clocks

In Run mode, the speed of the system clock ck_sys can be reduced. For more details refer to *Section 7.5.6: System clock (sys_ck)*.

5.7.2 Controlling peripheral clocks

In Run mode, the HCLKx and PCLKx for individual peripherals can be stopped by configuring at any time PERxEN bit in RCC_C1_xxxxENR or RCC_DnxxxxENR to reduce power consumption.

To reduce power consumption in CSleep mode, the individual peripheral clocks can be disabled by configuring PERxLPEN bit in RCC_C1_xxxxLPENR or RCC_DnxxxxLPENR. For the peripherals still receiving a clock in CSleep mode, their clock can be slowed down before entering CSleep mode.

5.7.3 Entering low-power modes

CPU subsystem CSleep and CStop low-power modes are entered by the MCU when executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M System Control register is set on Return from ISR.

A domain can enter DStop or DStandby low-power mode when the CPU subsystem has an allocated peripheral in the domain and enters CStop mode, or when all D2 domain peripherals are deallocated.

The system can enter Stop or Standby low-power mode when all EXTI wakeup sources are cleared and the other domains are in DStop or DStandby mode.

5.7.4 Exiting from low-power modes

The CPU subsystem exits from CSleep mode through any interrupt or event depending on how the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter to low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the system.
- If the WFE instruction is used to enter to low-power mode, the CPU exits from low-power mode as soon as an event occurs. The wakeup event can be generated either by:
  - An NVIC IRQ interrupt.

    When SEVONPEND = 0 in the Cortex®-M7 System Control register, the interrupt must be enabled in the peripheral control register and in the NVIC.

    When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit in the NVIC interrupt clear pending register have to be cleared. Only NVIC interrupts with sufficient priority will wakeup and interrupt the MCU.

    When SEVONPEND = 1 in the Cortex®-M7 System Control register, the interrupt must be enabled in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and, when enabled, the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

    All NVIC interrupts will wakeup the MCU, even the disabled ones.

    Only enabled NVIC interrupts with sufficient priority will wakeup and interrupt the MCU.
  - An event

    An EXTI line must be configured in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It might be necessary to clear the interrupt flag in the peripheral.

The CPU subsystem exits from CStop, DStop and Stop modes by enabling an EXTI interrupt or event depending on how the low-power mode was entered (see above).

The system can wakeup from Stop mode by enabling an EXTI wakeup, without waking up a CPU subsystem. In this case the system will operate in D3 autonomous mode.

The CPU subsystem exits from DStandby mode by enabling an EXTI interrupt or event, regardless on how DStandby mode was entered. Program execution restarts from CPU local reset (such as a reset vector fetched from System configuration block (SYSCFG).

The D2 domain can exit from DStop or DStandby mode when the CPU allocates a first peripheral in the domain.

The CPU subsystem exits from Standby mode by enabling an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC event. Program execution restarts in the same way as after a system reset (such as boot pin sampling, option bytes loading or reset vector fetched).

5.7.5 CSleep mode

The CSleep mode applies only to the CPU subsystem. In CSleep mode, the CPU clock is stopped. The CPU subsystem peripheral clocks operate according to the values of PERxLPEN bits in RCC_C1_xxxxENR or RCC_DnxxxxENR.

Entering CSleep mode

The CSleep mode is entered according to *Section 5.7.3: Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M System Control register is cleared.

Refer to *Table 34* for details on how to enter to CSleep mode.

Exiting from CSleep mode

The CSleep mode is exited according to *Section 5.7.4: Exiting from low-power modes*.

Refer to *Table 34* for more details on how to exit from CSleep mode.

Table 34. CSleep mode

CSleep mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 (Refer to the Cortex®-M System Control register.) – CPU NVIC interrupts and events cleared.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 (refer to the Cortex®-M System Control register.) – CPU NVIC interrupts and events cleared.
Mode exit	If WFI or return from ISR was used for entry: – Any Interrupt enabled in NVIC: Refer to *Table 138: NVIC* If WFE was used for entry and SEVONPEND = 0: – Any event: Refer to *Section 19.5.3: EXTI CPU wakeup procedure* If WFE was used for entry and SEVONPEND = 1: – Any Interrupt even when disabled in NVIC: refer to *Table 138: NVIC* or any event: refer to *Section 19.5.3: EXTI CPU wakeup procedure*
Wakeup latency	None

5.7.6 CStop mode

The CStop mode applies only to the CPU subsystem. In CStop mode, the CPU clock is stopped. Most CPU subsystem peripheral clocks are stopped too and only the CPU subsystem peripherals having a PERxAMEN bit operate accordingly.

In CStop mode, the CPU subsystem peripherals that have a kernel clock request can still request their kernel clock. For the peripheral that have a PERxAMEN bit, this bit shall be set to be able to request the kernel clock.

Entering CStop mode

The CStop mode is entered according to *Section 5.7.3: Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M System Control register is set.

Refer to *Table 35* for details on how to enter to CStop mode.

Exiting from CStop mode

The CStop mode is exited according to *Section 5.7.4: Exiting from low-power modes*.

Refer to *Table 35* for more details on how to exit from CStop mode.

Table 35. CStop mode

CStop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 1 (Refer to the Cortex®-M System Control register.) – CPU NVIC interrupts and events cleared. – All CPU EXTI Wakeup sources are cleared.
	On return from ISR while: – SLEEPDEEP = 1 and – SLEEPONEXIT = 1 (Refer to the Cortex®-M System Control register.) – CPU NVIC interrupts and events cleared. – All CPU EXTI Wakeup sources are cleared.
Mode exit	If WFI or return from ISR was used for entry: – EXTI Interrupt enabled in NVIC: Refer to *Table 138: NVIC*, for peripheral which are not stopped or powered down. If WFE was used for entry and SEVONPEND = 0: – EXTI event: Refer to *Section 19.5.3: EXTI CPU wakeup procedure*, for peripheral which are not stopped or powered down. If WFE was used for entry and SEVONPEND = 1: – EXTI Interrupt even when disabled in NVIC: refer to *Table 138: NVIC* or EXTI event: refer to *Section 19.5.3: EXTI CPU wakeup procedure*, for peripheral which are not stopped or powered down.
Wakeup latency	EXTI and RCC wakeup synchronization (see *Section 7.4.7: Power-on and wakeup sequences*)

5.7.7 DStop mode

D1 domain and/or D2 domain enters DStop mode only when the CPU subsystem is in CStop mode and has allocated peripheral in the domain (see *Table 36*). In DStop mode the domain bus matrix clock is stopped.

The Flash memory can enter low-power Stop mode when it is enabled through FLPS in PWR_CR1 register. This allows a trade-off between domain DStop restart time and low power consumption.

Table 36. DStop mode overview

Peripheral allocation	CPU	D1 domain	D2 domain	Comment
No peripheral allocated in D2 domain	CRun or CSleep	DRun	DStop	
	CStop	DStop	DStop	
Peripheral allocated in D2 domain	CRun or CSleep	DRun	DRun	CPU subsystem, keep D2 domain active.
	CStop	DStop	DStop	

In DStop mode domain peripherals using the LSI or LSE clock and peripherals having a kernel clock request are still able to operate.

Entering DStop mode

The DStop mode is entered according to *Section 5.7.3: Entering low-power modes*, when at least one PDDS_Dn bit in *PWR CPU control register (PWR_CPUCR)* for the domain select Stop.

Refer to *Table 37* for details on how to enter DStop mode.

If Flash memory programming is ongoing, the DStop mode entry is delayed until the memory access is finished.

If an access to the domain bus matrix is ongoing, the DStop mode entry is delayed until the domain bus matrix access is complete.

Exiting from DStop mode

The DStop mode is exited according to *Section 5.7.4: Exiting from low-power modes*.

Refer to *Table 37* for more details on how to exit from DStop mode.

When exiting from DStop mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling depend on the system mode.

- When the system did not enter Stop mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling values are the same as when entering DStop mode.
- When the system has entered Stop mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling are reset.

Table 37. DStop mode

DStop mode	Description
Mode entry	– The domain CPU subsystem enters CStop. – The CPU subsystem has an allocated peripheral in the D2 domain and enters CStop. – The CPU subsystem deallocated its last peripheral in the D2 domain. – The PDDS_Dn bit for the domain selects Stop mode.
Mode exit	– The domain CPU subsystem exits from CStop mode (see *Table 35*) – The CPU subsystem has an allocated peripheral in the D2 domain and exits from CStop mode (see *Table 35*) – The CPU subsystem allocates a first peripheral in the D2 domain.
Wakeup latency	EXTI and RCC wakeup synchronization (see *Section 7.4.7: Power-on and wakeup sequences*).

5.7.8 Stop mode

The system D3 domain enters Stop mode only when the CPU subsystem is in CStop mode, the EXTI wakeup sources are inactive and at least one PDDS_Dn bit in *PWR CPU control register (PWR_CPUCR)* for any domain request Stop. In Stop mode, the system clock including a PLL and the D3 domain bus matrix clocks are stopped. When HSI or CSI is selected, the system oscillator operates according to the HSIKERON and CSIKERON bits in RCC_CR register. Other system oscillator sources are stopped.

In system D3 domain Stop mode, D1 domain and D2 domain are either in DStop and/or DStandby mode.

In Stop mode, the domain peripherals that use the LSI or LSE clock, and the peripherals that have a kernel clock request to select HSI or CSI as source, are still able to operate.

In system Stop mode, the following features can be selected to remain active by programming individual control bits:

- Independent watchdog (IWDG)

  The IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset (see *Section 44.3* in *Section 44: Independent watchdog (IWDG)*.
- Real-time clock (RTC)

  This is configured via the RTCEN bit in the *RCC Backup Domain Control Register (RCC_BDCR)*.
- Internal RC oscillator (LSI RC)

  This is configured via the LSION bit in the *RCC Clock Control and Status Register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE OSC)

  This is configured via the LSEON bit in the *RCC Backup Domain Control Register (RCC_BDCR)*.

- Peripherals capable of running on the LSI or LSE clock.
- Peripherals having a kernel clock request.
- Internal RC oscillators (HSI and CSI)
- This is configured via the HSIKERON and CSIKERON bits in the *RCC Clock Control and Status Register (RCC_CSR)*.
- The ADC or DAC can also consume power during Stop mode, unless they are disabled before entering this mode. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register must both be written to 0.

The selected SVOS4 and SVOS5 levels add an additional startup delay when exiting from system Stop mode (see *Table 38*).

Table 38. Stop mode operation

SVOS	LPDS	Stop mode Voltage regulator operation	Wake-up Latency
SVOS3	0	**Main**	No additional wakeup time.
	1	**LP**	Voltage Regulator wakeup time from LP mode.
SVOS4 or SVOS5	x	**LP**	Voltage Regulator wakeup time from LP mode + voltage level wakeup time for SVOS4 or SVOS5 level to VOS3 level

Entering Stop mode

The Stop mode is entered according to *Section 5.7.3: Entering low-power modes*, when at least one PDDS_Dn bit n *PWR CPU control register (PWR_CPUCR)* for any domain request Stop.

Refer to *Table 39* for details on how to enter Stop mode.

If Flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to a bus matrix (AXI, AHB or APB) is ongoing, the Stop mode entry is delayed until the bus matrix access is finished.

Note: Use a DSB instruction to ensure that outstanding memory transactions complete before entering stop mode.

To allow peripherals having a kernel clock request to operate in Stop mode, the system must use SVOS3 level.

Exiting from Stop mode

The Stop mode is exited according to *Section 5.7.4: Exiting from low-power modes*.

Refer to *Table 39* for more details on how to exit from Stop mode.

When exiting from Stop mode, the system clock, D3 domain bus matrix clocks and voltage scaling are reset.

STOPF status flag in *PWR CPU control register (PWR_CPUCR)* indicates that the system has exited from Stop mode (see *Table 33*).

Table 39. Stop mode

Stop mode	Description
Mode entry	– When the CPU is in CStop mode and there is no active EXTI Wakeup source and Run_D3 = 0. – At least one PDDS_Dn bit for any domain select Stop.
Mode exit	– On a EXTI Wakeup.
Wakeup latency	System oscillator startup (when disabled). + EXTI and RCC wakeup synchronization. + Voltage Scaling refer to *Table 38* (see *Section 5.6.2: Voltage scaling*)

I/O states in Stop mode

I/O pin configuration remain unchanged in Stop mode.

5.7.9 DStandby mode

Like DStop mode, DStandby mode is based on the CPU subsystem CStop mode. However the domain V_{CORE} supply is powered off. A domain enters DStandby mode only when the CPU subsystem is in CStop mode if peripherals are allocated in the domain

A domain enters DStandby mode only when the CPU subsystem is in CStop mode if peripherals are allocated in the domain and the PDDS_Dn bit in *PWR CPU control register (PWR_CPUCR)* for the domain is configured accordingly. In DStandby mode, the domain is powered down and the domain RAM and register contents are lost.

Entering DStandby mode

The DStandby mode is entered according to *Section 5.7.3: Entering low-power modes*, when the PDDS_Dn bit in *PWR CPU control register (PWR_CPUCR)* for the Dn domain selects Standby mode.

Refer to *Table 40* for details on how to enter DStandby mode.

If Flash memory programming is ongoing, the DStandby mode entry is delayed until the memory access is finished.

If an access to the domain bus matrix is ongoing, the DStandby mode entry is delayed until the domain bus matrix access is finished.

Note: When the CPU sets the PDDS_D2 bit to select Standby mode, the D2 domain enters DStandby mode (the CPU has no allocated peripherals in the D2 domain).

Exiting from DStandby mode

The DStandby mode is exited according to *Section 5.7.4: Exiting from low-power modes*.

Refer to *Table 40* for more details on how to exit from DStandby mode.

Note: When the D2 domain is in DStandby mode and the CPU sets the domain PDDS_D2 bit to select Stop mode, the D2 domain remains in DStandby mode. The D2 domain will only exit DStandby when the CPU allocates a peripheral in the D2 domain.

When exiting from DStandby mode, the domain CPU and peripherals are reset. However the state of the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling depends on the system mode:

- When the system did not enter Stop mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling are the same as when entering DStandby mode.
- When the system has entered Stop or Standby mode, the CPU subsystem clocks, domain(s) bus matrix clocks and voltage scaling are reset.

When the D2 domain exits from DStandby mode due to the CPU subsystem (i.e when allocating a first peripheral or when peripherals are allocated in the D2 domain and the CPU subsystem exits from CStop mode), the CPU shall verify that the domain has exited from DStandby mode. To ensure correct operation, it is recommended to follow the sequence below:

1. First check that the domain bus matrix clock is available. The domain bus matrix clock state can be checked in RCC_CR register:
   – When RCC DnCKRDY = 0, the domain bus matrix clock is stalled.
   – If RCC DnCKRDY = 1, the domain bus matrix clock is enabled.
2. Then wait for the domain has exited from DStandby mode. To do this, check the SBF_Dn flag in *PWR CPU control register (PWR_CPUCR)*. The domain is powered and can be accessed only when SBF_Dn is cleared. Below an example of code:

```
Loop
write PWR SBF_Dn = 0 ; try to clear bit.
read PWR SBF_Dn
While 1 ==> loop
```

Table 40. DStandby mode

DStandby mode	Description
Mode entry	– The domain CPU subsystem enters CStop. – The CPU subsystem has an allocated peripheral in D2 domain and enters CStop. – The CPU subsystem deallocated its last peripheral in the D2 domain. – The PDDS_Dn bits for the domain select Standby mode. – All WKUPF bits in Power Control/Status register (PWR_WKUPFR) are cleared.
Mode exit	– The CPU subsystem exits from CStop mode (see *Table 35*) – The CPU subsystem has an allocated peripheral in the D2 domain and exits from CStop mode (see *Table 35*) – The CPU subsystem allocates a first peripheral in the D2 domain.
Wakeup latency	EXTI and RCC wakeup synchronization. + Domain power up and reset. (see *Section 7.4.7: Power-on and wakeup sequences*)

5.7.10 Standby mode

The Standby mode allows achieving the lowest power consumption. Like Stop mode, it is based on CPU subsystem CStop mode. However the V_{CORE} supply regulator is powered off.

The system D3 domain enters Standby mode only when the D1 and D2 domain are in DStandby. When the system D3 domain enters Standby mode, the voltage regulator is disabled. The complete V_{CORE} domain is consequently powered off. The PLLs, HSI oscillator, CSI oscillator, HSI48 and the HSE oscillator are also switched off. SRAM and register contents are lost except for backup domain registers (RTC registers, RTC backup register and backup RAM), and Standby circuitry (see *Section 5.4.4: Backup domain*).

In system Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG)

  The IWDG is started by programming its Key register or by hardware option. Once started, it cannot be stopped except by a reset (see *Section 44.3* in *Section 44: Independent watchdog (IWDG)*.
- Real-time clock (RTC)

  This is configured via the RTCEN bit in the backup domain control register (RCC_BDCR).
- Internal RC oscillator (LSI RC)

  This is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE OSC)

  This is configured by the LSEON bit in the backup domain control register (RCC_BDCR).

Entering Standby mode

The Standby mode is entered according to *Section 5.7.3: Entering low-power modes*, when all PDDS_Dn bits in *PWR CPU control register (PWR_CPUCR)* for all domains request Standby.

Refer to *Table 42* for more details on how to enter to Standby mode.

Exiting from Standby mode

The Standby mode is exited according to *Section 5.7.4: Exiting from low-power modes*.

Refer to *Table 42* for more details on how to exit from Standby mode.

The system exits from Standby mode when an external Reset (NRST pin), an IWDG Reset, a WKUP pin event, a RTC alarm, a tamper event, or a time stamp event is detected. All registers are reset after waking up from Standby except for power control and status registers (*PWR control register 2 (PWR_CR2)*, *PWR control register 3 (PWR_CR3)*), SBF bit in *PWR CPU control register (PWR_CPUCR)*, *PWR wakeup flag register (PWR_WKUPFR)*, and *PWR wakeup enable and polarity register (PWR_WKUPEPR)*.

After waking up from Standby mode, the program execution restarts in the same way as after a system reset (boot option sampling, boot vector reset fetched, etc.). The SBF status flags in *PWR CPU control register (PWR_CPUCR)* registers indicate from which mode the system has exited (see *Table 41*).

Table 41. Standby and Stop flags

SBF_D2	SBF_D1	SBF	STOPF	Description
0	1	0	0	D1 domain exits from DStandby while system stayed in Run
0	1	0	1	D1 domain exits from DStandby, while system has been in or exits from Stop
1	0	0	0	D2 domain exits from DStandby while system stayed in Run
1	0	0	1	D2 domain exits from DStandby while system has been in or exits from Stop
1	1	0	0	D1 and D2 domain exit from DStandby while the system remains in Run mode
1	1	0	1	D1 and D2 domain exit from DStandby while the system is in Stop mode or is exiting this mode.
0	0	0	1	System has been in or exits from Stop
0(1)	0(1)	1	0	System exits from Standby

1. When exiting from Standby the SBF_D1 and SBF_D2 reflect the reset value

Table 42. Standby mode

Standby mode	Description
Mode entry	– The CPU subsystem is in CStop mode, and there is no active EXTI Wakeup source and RUN_D3 = 0. – All PDDS_Dn bits for all domains select Standby. – All WKUPF bits in Power Control/Status register (PWR_WKUPFR) are cleared.
Mode exit	– WKUP pins rising or falling edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time stamp event, external reset in NRST pin, IWDG reset.
Wakeup latency	System reset phase (see *Section 7.4.2: System reset*)

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance without pull, except for:

- Reset pad (still available)
- RTC_AF1 pin if configured for tamper, time stamp, RTC Alarm out, or RTC clock calibration out
- WKUP pins (if enabled). The WKUP pin pull configuration can be defined through WKUPPUPD register bits in *PWR wakeup enable and polarity register (PWR_WKUPEPR)*.

5.8 PWR register description

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

5.8.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0xF000 C000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS		AVDEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVOS		Res.	Res.	Res.	Res.	FLPS	DBP	PLS			PVDE	Res.	Res.	Res.	LPDS
rw	rw					rw	rw	rw	rw	rw	rw				rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **ALS:** Analog voltage detector level selection
These bits select the voltage threshold detected by the AVD.
00: 1.7 V
01: 2.1 V
10: 2.5 V
11: 2.8 V

Bit 16 **AVDEN:** Peripheral voltage monitor on V_{DDA} enable
0: Peripheral voltage monitor on V_{DDA} disabled.
1: Peripheral voltage monitor on V_{DDA} enabled

Bits 15:14 **SVOS:** System Stop mode voltage scaling selection
These bits control the V_{CORE} voltage level in system Stop mode, to obtain the best trade-off between power consumption and performance.
00: Reserved
01: SVOS5 Scale 5
10: SVOS4 Scale 4
11: SVOS3 Scale 3 (default)

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **FLPS:** Flash low-power mode in DStop mode
This bit allows to obtain the best trade-off between low-power consumption and restart time when exiting from DStop mode.
When it is set, the Flash memory enters low-power mode when D1 domain is in DStop mode.
0: Flash memory remains in normal mode when D1 domain enters DStop (quick restart time).
1: Flash memory enters low-power mode when D1 domain enters DStop mode (low-power consumption).

Bit 8 **DBP:** Disable backup domain write protection

In reset state, the RCC_BDCR register, the RTC registers (including the backup registers), BREN and MOEN bits in PWR_CR2 register, are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Access to RTC, RTC Backup registers and backup SRAM disabled

1: Access to RTC, RTC Backup registers and backup SRAM enabled

Bits 7:5 **PLS:** Programmable voltage detector level selection

These bits select the voltage threshold detected by the PVD.

000: 1.95 V

001: 2.1 V

010: 2.25 V

011: 2.4 V

100: 2.55 V

101: 2.7 V

110: 2.85 V

111: External voltage level on PVD_IN pin, compared to internal V_{REFINT} level.

Note: Refer to Section "Electrical characteristics" of the product datasheet for more details.

Bit 4 **PVDE:** Programmable voltage detector enable

0: Programmable voltage detector disabled.

1: Programmable voltage detector enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **LPDS:** Low-power Deepsleep with SVOS3 (SVOS4 and SVOS5 always use low-power, regardless of the setting of this bit)

0: Voltage regulator in Main mode (MR) when SVOS3 is selected for Stop mode

1: Voltage regulator in Low-power mode (LPR) when SVOS3 is selected for Stop mode

5.8.2 PWR control status register 1 (PWR_CSR1)

Address offset: 0x004

Reset value: 0x0000 4000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVDO
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTVOS		ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.
r		r									r				

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AVDO**: Analog voltage detector output on V_{DDA}

This bit is set and cleared by hardware. It is valid only if AVD on V_{DDA} is enabled by the AVDEN bit.

0: V_{DDA} is equal or higher than the AVD threshold selected with the ALS[2:0] bits.

1: V_{DDA} is lower than the AVD threshold selected with the ALS[2:0] bits

Note: Since the AVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the AVDEN bit is set.

Bits 15:14 **ACTVOS**: VOS currently applied for V_{CORE} voltage scaling selection.

These bits reflect the last VOS value applied to the voltage regulator.

Bit 13 **ACTVOSRDY:** Voltage levels ready bit for currently used VOS

This bit is set to 1 by hardware when the voltage regulator is disabled and Bypass mode is selected in PWR control register 3 (PWR_CR3).

0: Voltage level invalid, above or below current VOS selected level.

1: Voltage level valid, at current VOS selected level.

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **PVDO**: Programmable voltage detect output

This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.

0: V_{DD} or PVD_IN voltage is equal or higher than the PVD threshold selected through the PLS[2:0] bits.

1: V_{DD} or PVD_IN voltage is lower than the PVD threshold selected through the PLS[2:0] bits.

Note: since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bits 3:0 Reserved, must be kept at reset value.

5.8.3 PWR control register 2 (PWR_CR2)

Address offset: 0x008

Reset value: 0x0000 0000

This register is not reset by wakeup from Standby mode, RESET signal and V_{DD} POR. It is only reset by V_{SW} POR and VSWRST reset.

This register shall not be accessed when VSWRST bit in RCC_BDCR register resets the V_{SW} domain.

After reset, PWR_CR2 register is write-protected. Prior to modifying its content, the DBP bit in PWR_CR1 register must be set to disable the write protection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VBATH	VBATL	Res.	Res.	Res.	BRRDY
								r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.	Res.	Res.	BREN
											rw				rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TEMPH**: Temperature level monitoring versus high threshold
0: Temperature below high threshold level.
1: Temperature equal or above high threshold level.

Bit 22 **TEMPL**: Temperature level monitoring versus low threshold
0: Temperature above low threshold level.
1: Temperature equal or below low threshold level.

Bit 21 **VBATH**: V_{BAT} level monitoring versus high threshold
0: V_{BAT} level below high threshold level.
1: V_{BAT} level equal or above high threshold level.

Bit 20 **VBATL**: V_{BAT} level monitoring versus low threshold
0: V_{BAT} level above low threshold level.
1: V_{BAT} level equal or below low threshold level.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BRRDY**: Backup regulator ready
This bit is set by hardware to indicate that the Backup regulator is ready.
0: Backup regulator not ready.
1: Backup regulator ready.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **MONEN:** V_{BAT} and temperature monitoring enable
When set, the V_{BAT} supply and temperature monitoring is enabled.
0: V_{BAT} and temperature monitoring disabled.
1: V_{BAT} and temperature monitoring enabled.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **BREN:** Backup regulator enable
When set, the Backup regulator (used to maintain the backup RAM content in Standby and V_{BAT} modes) is enabled.
If BREN is reset, the backup regulator is switched off. The backup RAM can still be used in Run and Stop modes. However, its content will be lost in Standby and V_{BAT} modes.
If BREN is set, the application must wait till the Backup Regulator Ready flag (BRRDY) is set to indicate that the data written into the SRAM will be maintained in Standby and V_{BAT} modes.
0: Backup regulator disabled.
1: Backup regulator enabled.

5.8.4 PWR control register 3 (PWR_CR3)

Address offset: 0x00C

Reset value: 0x0000 0006 (reset only by POR only, not reset by wakeup from Standby mode and RESET pad).

The lower byte of this register is written once after POR and shall be written before changing VOS level or ck_sys clock frequency. No limitation applies to the upper bytes.

Programming data corresponding to an invalid combination of LDOEN and BYPASS bits (see *Table 30*) will be ignored: data will not be written, the written-once mechanism will lock the register and any further write access will be ignored. The default supply configuration will be kept and the ACTVOSRDY bit in *PWR control status register 1 (PWR_CSR1)* will go on indicating invalid voltage levels. The system shall be power cycled before writing a new value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	USB33RDY	USBREGEN	USB33DEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					r	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	SCUEN	LDOEN	BYPASS
						rw	rw						rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **USB33RDY**: USB supply ready.
0: USB33 supply not ready.
1: USB33 supply ready.

Bit 25 **USBREGEN**: USB regulator enable.
0: USB regulator disabled.
1: USB regulator enabled.

Bit 24 **USB33DEN**: $V_{DD33USB}$ voltage level detector enable.
0: $V_{DD33USB}$ voltage level detector disabled.
1: $V_{DD33USB}$ voltage level detector enabled.

Bits 23:10 Reserved, must be kept at reset value.

Bit 9 **VBRS**: V_{BAT} charging resistor selection
0: Charge V_{BAT} through a 5 kΩ resistor.
1: Charge V_{BAT} through a 1.5 kΩ resistor.

Bit 8 **VBE**: V_{BAT} charging enable
0: V_{BAT} battery charging disabled.
1: V_{BAT} battery charging enabled.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **SCUEN**: Supply configuration update enable
This bit is read-only:
0: Supply configuration update locked.
1: Single write enabled to Supply configuration (LDOEN and BYPASS)

Bit 1 **LDOEN**: Low drop-out regulator enable
0: Low drop-out regulator disabled.
1: Low drop-out regulator enabled (default)

Bit 0 **BYPASS**: Power management unit bypass
0: Power management unit normal operation.
1: Power management unit bypassed, voltage monitoring still active.

5.8.5 PWR CPU control register (PWR_CPUCR)

This register allows controlling CPU power.

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RUN_D3	Res.	CSSF	SBF_D2	SBF_D1	SBF	SOPFF	Res.	Res.	PDDS_D3	PDDS_D2	PDDS_D1
				rw		rw	r	r	r	r			rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RUN_D3:** Keep system D3 domain in Run mode regardless of the CPU subsystem modes
0: D3 domain follows CPU subsystem modes.
1: D3 domain remains in Run mode regardless of CPU subsystem modes.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CSSF:** Clear Standby and Stop flags (always read as 0)
This bit is cleared to 0 by hardware.
0: No effect.
1: STOPF, SBF, SBF_D1, and SBF_D2 flags are cleared.

Bit 8 **SBF_D2:** D2 domain DStandby flag
This bit is set by hardware and cleared by any system reset or by setting the CSSF bit. Once set, this bit can be cleared only when the D2 domain is no longer in DStandby mode.
0: D2 domain has not been in DStandby mode
1: D2 domain has been in DStandby mode.

Bit 7 **SBF_D1:** D1 domain DStandby flag
This bit is set by hardware and cleared by any system reset or by setting the CSSF bit. Once set, this bit can be cleared only when the D1 domain is no longer in DStandby mode.
0: D1 domain has not been in DStandby mode
1: D1 domain has been in DStandby mode.

Bit 6 **SBF:** System Standby flag
This bit is set by hardware and cleared only by a POR (Power-on Reset) or by setting the CSSF bit
0: System has not been in Standby mode
1: System has been in Standby mode

Bit 5 **STOPF:** STOP flag
This bit is set by hardware and cleared only by any reset or by setting the CSSF bit.
0: System has not been in Stop mode
1: System has been in Stop mode

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **PDDS_D3:** System D3 domain Power Down Deepsleep.
This bit allows defining the Deepsleep mode for System D3 domain.
0: Keep Stop mode when D3 domain enters Deepsleep.
1: Allow Standby mode when D3 domain enters Deepsleep.

Bit 1 **PDDS_D2:** D2 domain Power Down Deepsleep.
This bit allows defining the Deepsleep mode for D2 domain.
0: Keep DStop mode when D2 domain enters Deepsleep.
1: Allow DStandby mode when D2 domain enters Deepsleep.

Bit 0 **PDDS_D1:** D1 domain Power Down Deepsleep selection.
This bit allows defining the Deepsleep mode for D1 domain.
0: Keep DStop mode when D1 domain enters Deepsleep.
1: Allow DStandby mode when D1 domain enters Deepsleep.

5.8.6 PWR D3 domain control register (PWR_D3CR)

This register allows controlling D3 domain power.

Address offset: 0x018

Reset value: 0x0000 4000 (Following reset VOSRDY will be read 1 by software).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VOS		VOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		r													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **VOS:** Voltage scaling selection according to performance
These bits control the V_{CORE} voltage level and allow to obtains the best trade-off between power consumption and performance:
– When increasing the performance, the voltage scaling shall be changed before increasing the system frequency.
– When decreasing performance, the system frequency shall first be decreased before changing the voltage scaling.
00: Reserved (Scale 3 selected).
01: Scale 3 (default)
10: Scale 2
11: Scale 1

Bit 13 **VOSRDY:** VOS Ready bit for V_{CORE} voltage scaling output selection.
This bit is set to 1 by hardware when Bypass mode is selected in PWR control register 3 (PWR_CR3).
0: Not ready, voltage level below VOS selected level.
1: Ready, voltage level at or above VOS selected level.

Bits 12:0 Reserved, must be kept at reset value.

5.8.7 PWR wakeup clear register (PWR_WKUPCR)

Address offset: 0x020

Reset value: 0x0000 0000 (reset only by system reset, not reset by wakeup from Standby mode)

5 wait states are required when writing this register (when clearing a WKUPF bit in PWR_WKUPFR, the AHB write access will complete after the WKUPF has been cleared).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPC6	WKUPC5	WKUPC4	WKUPC3	WKUPC2	WKUPC1
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:6 Reserved, always read as 0.

Bits 5:0 **WKUPCn+1:** Clear Wakeup pin flag for WKUPn+1.

These bits are always read as 0.

0: No effect

1: Writing 1 clears the WKUPFn+1 Wakeup pin flag (bit is cleared to 0 by hardware)

5.8.8 PWR wakeup flag register (PWR_WKUPFR)

Address offset: 0x024

Reset value: 0x0000 0000 (reset only by system reset, not reset by wakeup from Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF6	WKUPF5	WKUPF4	WKUPF3	WKUPF2	WKUPF1
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **WKUPn+1:** Wakeup pin WKUPn+1 flag.

This bit is set by hardware and cleared only by a Reset pin or by setting the WKUPCn+1 bit in the *PWR wakeup clear register (PWR_WKUPCR)*.

0: No wakeup event occurred

1: A wakeup event was received from WKUPn+1 pin

5.8.9 PWR wakeup enable and polarity register (PWR_WKUPEPR)

Address offset: 0x028

Reset value: 0x0000 0000 (reset only by system reset, not reset by wakeup from Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WKUPPUPD6		WKUPPUPD5		WKUPPUPD4		WKUPPUPD3		WKUPPUPD2		WKUPPUPD1	
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WKUPP6	WKUPP5	WKUPP4	WKUPP3	WKUPP2	WKUPP1	Res.	Res.	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WKUPPUPD[truncate(n/2)-7)**: Wakeup pin pull configuration for WKUP(truncate(n/2)-7)

These bits define the I/O pad pull configuration used when WKUPEN(truncate(n/2)-7) = 1. The associated GPIO port pull configuration shall be set to the same value or to ‘00’.

The Wakeup pin pull configuration is kept in Standby mode.

00: No pull-up

01: Pull-up

10: Pull-down

11: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **WKUPPn-7**: Wakeup pin polarity bit for WKUPn-7

These bits define the polarity used for event detection on WKUPn-7 external wakeup pin.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **WKUPENn+1**: Enable Wakeup Pin WKUPn+1

Each bit is set and cleared by software.

0: An event on WKUPn+1 pin does not wakeup the system from Standby mode.

1: A rising or falling edge on WKUPn+1 pin wakes-up the system from Standby mode.

Note: An additional wakeup event is detected if WKUPn+1 pin is enabled (by setting the WKUPENn+1 bit) when WKUPn+1 pin level is already high when WKUPPn+1 selects rising edge, or low when WKUPPn+1 selects falling edge.

5.8.10 PWR register map

Table 43. Power control register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS		AVDEN	SVOS						FLPS	DBP	PLS			PVDE				LPDS
	Reset value														0	0	0	1	1					0	0	0	0	0	0				0
0x004	**PWR_CSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVDO	ACTVOS		ACTVOSRDY									PVDO				
	Reset value																0	0	1	0									0				
0x008	**PWR_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VBATH	VBATL	Res.	Res.	Res.	BRRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.	Res.	Res.	BREN
	Reset value									0	0	0	0				0												0				0
0x00C	**PWR_CR3**	Res.	Res.	Res.	Res.	Res.	USB33RDY	USBREGEN	USB33DEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	SCUEN	LDOEN	BYPASS
	Reset value						0	0	0															0	0						1	1	0
0x010	**PWR_CPUCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RUN_D3	Res.	CSSF	SBF_D2	SBF_D1	SBF	STOPF	Res.	Res.	PDDS_D3	PDDS_D2	PDDS_D1
	Reset value																					0		0	0	0	0	0			0	0	0
0x014	**Reserved**	Reserved																															
	Reset value																																
0x018	**PWR_D3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS		VOSRDY													
	Reset value																	0	1	0													
0x020	**PWR_WKUPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPC6	WKUPC5	WKUPC4	WKUPC3	WKUPC2	WKUPC1
	Reset value																											0	0	0	0	0	0
0x024	**PWR_WKUPFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF6	WKUPF5	WKUPF4	WKUPF3	WKUPF2	WKUPF1
	Reset value																											0	0	0	0	0	0
0x028	**PWR_WKUPEPR**	Res.	Res.	Res.	Res.	WKUPPUPD6		WKUPPUPD5		WKUPPUPD4		WKUPPUPD3		WKUPPUPD2		WKUPPUPD1		Res.	Res.	WKUPP6	WKUPP5	WKUPP4	WKUPP3	WKUPP2	WKUPP1	Res.	Res.	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x030	Reserved																																

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

6 Low-power D3 domain

This section describes, through an example, how to use the D3 domain to implement low-power applications.

6.1 Introduction

The first part of the description explains how the EXTI, RCC and PWR blocks interact with each other and with the other system blocks. A detailed explanation on how the DMAMUX2 can be used to free the CPU is also provided.

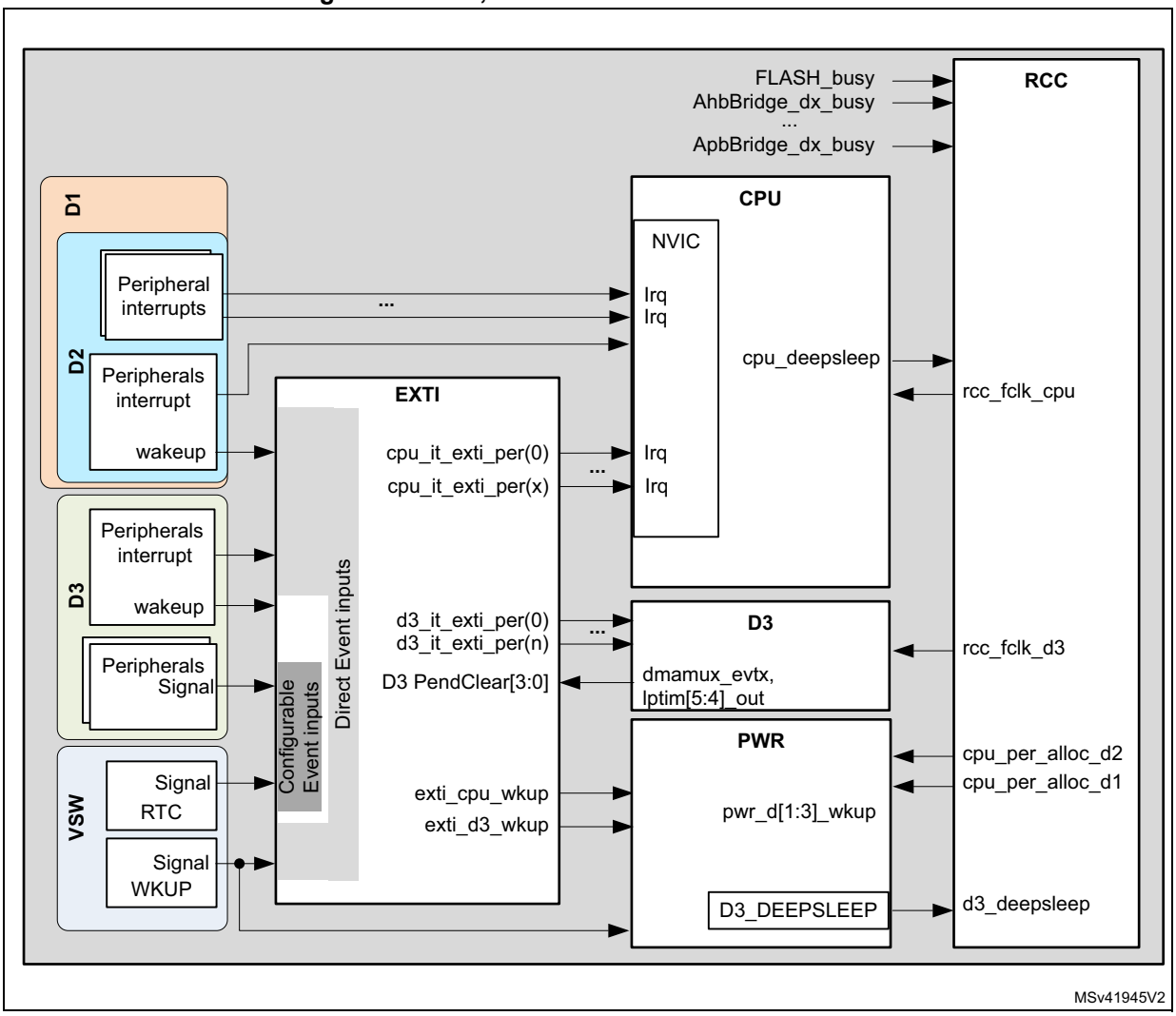
The second part explains how to use the Autonomous mode to perform simple data transfers through an example of LINUART1 transmission.

Register programming is detailed only for the blocks related to the Autonomous mode.

6.2 EXTI, RCC and PWR interconnections

Figure 35 shows the main EXTI, RCC and PWR interconnections.

Figure 35. EXTI, RCC and PWR interconnections



6.2.1 Interrupts and wakeup

Three kinds of signals are exchanged between the peripherals. They can be used to wake up the system from Stop mode:

- **Wakeup events** (or asynchronous interrupts)

  Some peripherals can generate interrupt events, even if their bus interface clock is not present. These interrupt events are called wakeup events (or asynchronous interrupts).

  Example: i2c1_wkup, usart1_wkup and lptim1_wkup.

- **Signals**

  Some peripherals generate a pulse instead of an interrupt signal. These pulses are called signals.

  Examples: lptim2_out and lptim3_out.

- **Interrupts**

  Contrary to signals, the interrupts should be cleared by a CPU or any other bus master, either by clearing the corresponding event bit in the peripheral register or by updating the FIFO interrupt level.

  All the interrupts associated to system peripherals are directly connected to the NVIC, except for the peripherals which are able to wake up the system from Stop mode or the CPU from CStop. In this latter case, the interrupts, signals or wakeup events are connected to the NVIC via the EXTI.

  Example: spi1_it, tim1_brk_it and tim1_upd_it.

The interrupt and wakeup sources that require to be cleared in the peripheral itself are connected to EXTI Direct Event inputs. The EXTI does not manage any CPU status pending bit.

The peripherals signals are connected to EXTI Configurable Event inputs. These EXTI inputs provide a CPU status pending bit which needs to be cleared by the application.

6.2.2 Block interactions

Interaction between EXTI and PWR blocks

The EXTI delivers wakeup requests signals (**exti_c_wkup**, **exti_d3_wkup**) to the PWR controller. These signals are activated according to the state of the interrupts, signals or wakeup events connected to the EXTI. These wakeup requests are used by the PWR controller to supply the domain who needs to handle the activated wakeup event generated by the peripherals.

Interaction between PWR and RCC blocks

The PWR block controls the V_{CORE} supply according to the system operating mode (CRun, CSleep or CStop). The PWR block also controls the power switches (ePODs) that delivers V_{CORE} supply to D1 and D2 domains.

The RCC block controls the clock generation in accordance with the system operating mode. It is also responsible for reset generation.

To synchronize the system mode transitions, the RCC block is tightly coupled with the PWR controller:

- The RCC informs the PWR controller when peripherals located in Dx domain are allocated by the CPU (**c_per_alloc_d2**, **c_per_alloc_d1**).
- The RCC also warns the PWR block when a domain clock is activated/deactivated. These signals are used in case of domain transition from DRun to DStop or DStandby. In this case, the PWR controller waits until the domain clock has been gated, before switching down this domain.
- Similarly, the PWR controller informs the RCC about the V_{CORE} supply status of each domain (**pwr_d[1:3]_wkup**). This information is used by the RCC when a domain transition from DStop or DStandby to DRun occurs.

Interaction between EXTI and D3 domain

All the wakeup event inputs received by the EXTI from the peripherals located in D3 domain are forwarded back to the D3 domain after system clock re-synchronization. These events are used by the D3 domain to perform autonomous operations without activating the CPU.

The EXTI **D3_PenClear[3:0]** inputs received from the D3 domain are used to acknowledge the ongoing wakeup requests generated by peripherals located in the D3 domain. The **D3_PenClear[3:0]** inputs allow switching the system D3 domain from Run to Stop mode.

6.2.3 Role of D3 domain DMAMUX2

The DMAMUX2 implemented in the D3 domain allows chaining BDMA transfers. BDMA requests are synchronized thanks to trigger events (**dmamux2_evtx**) which can be generated when the expected amount of data has been transferred.

These events can also trigger DMAMUX2 request generators (REQ_GEN[3:0]), and thus chain several BDMA transfers. In fact REQ_GEN[3:0] can be triggered indirectly by all the wakeup events generated by all D3 domain peripherals.

Like LPTIM5 and LPTIM4 outputs, dmamux2_evt7 and dmamux2_evt6 events are connected to the EXTI. They can be used to switch the D3 domain from DRun to DStop mode when the task requested by the wakeup event is complete.

6.3 Low-power application example based on LPUART1 transmission

This section illustrates, through an example, the benefit of the D3 domain usage on power consumption. To help the user program the device, only the key register settings are given herein.

Refer to *Section 7: Reset and Clock Control (RCC)* and *Section 5: Power control (PWR)*for additional details.

6.3.1 Memory retention

The D3 domain features 64 Kbytes of SRAM (SRAM4), which can be used to retain data while the D1 and D2 domains enter DStandby mode.

This feature can be used in several use-cases:

- to retain the application code in order to recover properly from DStandby
- to retain the data from/to a sensor when the CPU enters CStop with D1 or D2 domain in DStandby) between two consecutive operations.

Note: SRAM4 remains available as long as the system is not in Standby mode.

If the system is in Standby mode, it is still possible to use the BKUP_SRAM. However, its size is limited to 4 Kbytes.

6.3.2 Memory-to-peripheral transfer using LPUART1 interface

Example description

Figure 36 shows the proposed implementation. At a regular time interval given by LPTIM4, the CPU wakes up from CStop mode (which domain is in DStandby). When the CPU is in Run mode, it prepares the data to be transmitted via LPUART1, transfers them to SRAM4, and goes back to CStop. The D3 domain is configured to perform data transfers via LPUART1 and go back to Stop mode when the transfer is complete.

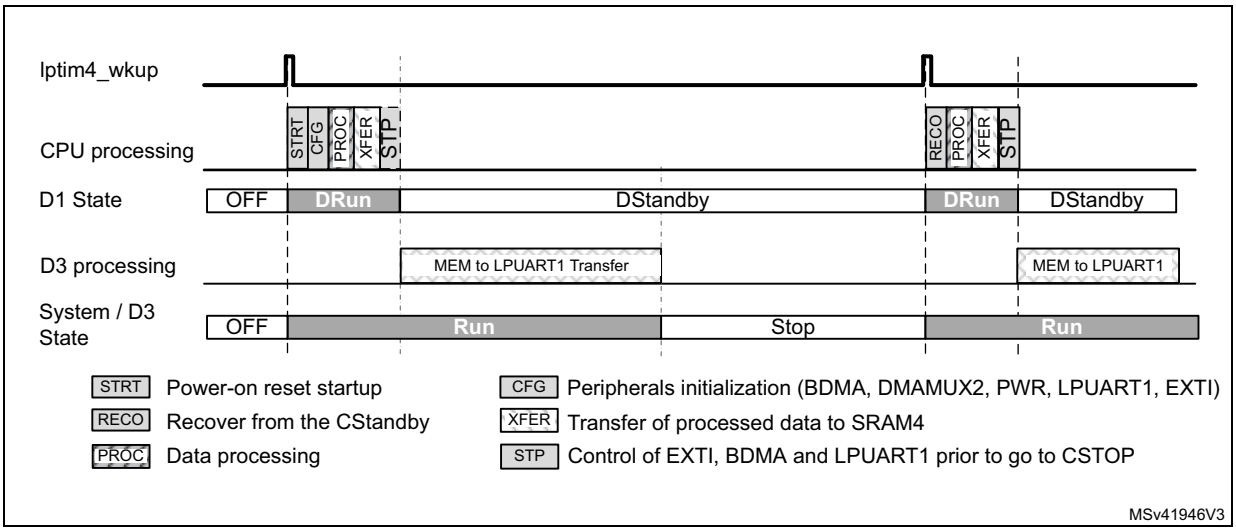
The LPTIM4 interface is used to wake up the system from Standby at regular time intervals. the CPU must then perform the following operations:

1. Recover the application from the system Standby mode (RECO).
2. Process the new data to be sent via LPUART1 (PROC).
3. Transfer the data into SRAM4 (XFER).
4. Configure the DMAMUX2, the BDMA, the LPUART1, and the RCC (CFG).
5. Configure the EXTI (CFG).
6. Configure the PWR block to allow the D1 domain to go to DStandby mode (STP).
7. Set the CPU to Stop mode.

The D3 domain executes the following tasks in Autonomous mode:

1. Transfer the data from SRAM4 to LPUART1, using BDMA.
2. When the LPUART1 interface indicates that the last byte has been transferred, the D3 domain is switched to Stop mode.

Figure 36. Timing diagram of SRAM4-to-LPUART1 transfer with BDMA and D3 domain in Autonomous mode



Note: In the above example described in this section, the D3 domain cannot be kept in Run mode when D1 and D2 domains are in DStop/DStandby by using the RUN_D3 bit of PWR_CPUCR register. RUN_D3 will force the D3 domain to Run mode, but it will not be able to go back to Stop on its own.

If the application needs to toggle the D3 domain between Stop and Run modes, then the Run mode must be triggered by a wakeup event so that the D3 domain can clear this event is needed.

RCC programming

In this example, the CPU sub-system also includes the peripherals of D3 domain that are used for the data transfer, that is BDMA, DMAMUX2, LPUART1 and LPTIM4. These peripherals must be programmed in Autonomous mode, in order to operate even when the CPU is in CSTOP mode.

LPUART1 can use its own APB clock as kernel clock. Since the system will not enter Stop mode before LPUART1 has completed data transfer, PLLx can be used to provide clocks to the peripherals.

PWR programming

In this example, the PWR block must be programmed in order to:

- Prevent system D3 domain to enter Standby mode when the data transfer is complete.
- Allow the D1 domain to enter DStandby.
- Define the working voltage according to system modes.

Note: D3 domain could enter Standby as well, but in this case the LPTIM4 could not be used to wake up the system and the AWU should be used instead. In addition, everything must be reprogrammed when the system wakes up.

EXTI programming

The EXTI block must be configured to provide the following services:

- Keep D3 domain running when D1 domain is in DStandby. This will be done by a software event.
- Set the device to Stop mode when the data transfer via LPUART1 is complete.
- Wake up the product from Stop when LPTIM4 time interval has elapsed.

The EXTI block is configured once before performing the first data transfer. For incoming data transfers, the programmed configuration remains unchanged; only some events need to be triggered or acknowledged.

Note: The CPU uses the event input number 0 to generate a software event. LPTIM4 wakeup signal is connected to event input number 52 (direct event input).

All other event inputs must be disabled: EXTI_RTSRx_TRy = '0' and EXTI_FTSRx_TRy = '0'.

To generate a wakeup event for D3 domain, the CPU must write SWIER0 bit of EXTI_SWIER1 to '1'. The GPIO connected to this event input must not toggle in order to avoid spurious wakeup. To prevent the GPIO to disturb, the following sequence can be done:

BDMA and DMAMUX2 programming

Two BDMA channels are required to execute data transfers via LPUART1.

- A BDMA channel, such as channel 0, is used to transfer data from SRAM4 to LPUART1, using the TXE flag.
- The second BDMA channel role is to switch the D3 domain to Stop mode. For that purpose, DMAMUX2 request generator channel 0 (REQ_GEN0) and DMAMUX2 channel 7 synchronization block (SYNC7) are used in conjunction with BDMA channel 7.

BDMA channel 0 does not use DMAMUX2 trigger capabilities. Refer to *Table 45* for initialization details.

BDMA channel 7 uses REQ_GEN0 to generate BDMA requests. The generation of BDMA requests is triggered by the LPUART1 transmit interrupt (**lpuart1_tx_it**). The LPUART1 interface generates lpuart1_tx_it interrupt when the transmit complete event is detected. The BDMA then clears the pending interrupt by performing a write operation to the LPUART1.

The SYNC7 block is programmed in Free-running mode. It generates a pulse on its **dmamux2_evt7** output when the BDMA request generated by the REQ_GEN0 is complete. **dmamux2_evt7** signal is used by the EXTI to switch back the D3 domain to Stop mode.

Figure 44 shows the active signal paths via DMAMUX2. The grayed blocks represent the unused paths.

Table 44. BDMA and DMAMUX2 interconnection

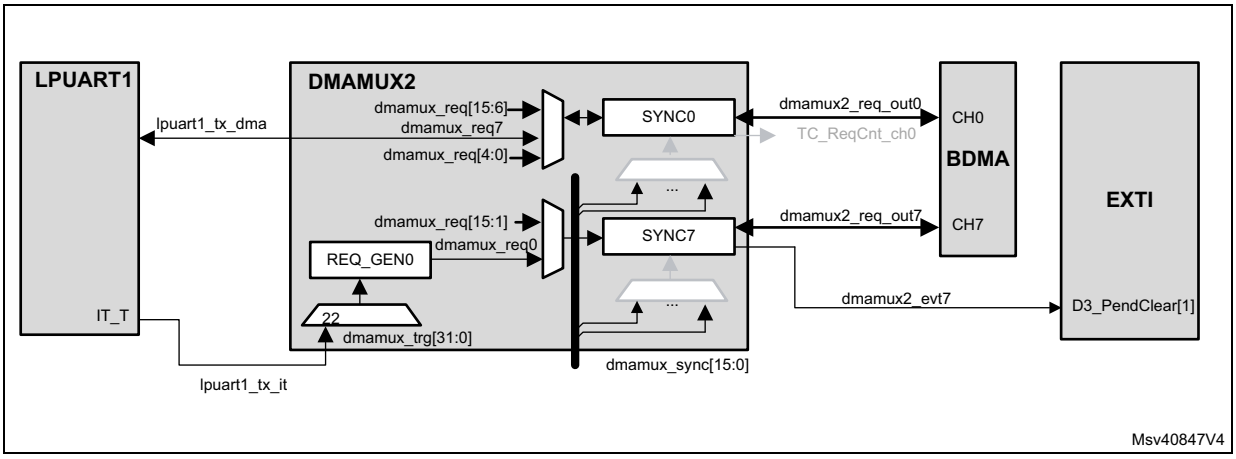


Table 45 explain how to program BDMA and DMAMUX2 key functions. The way errors are handled is not described.

Table 45. BDMA and DMAMUX2 initialization sequence (DMAMUX2_INIT)

Peripherals	Register content	Related actions
DMAMUX2 SYNC0	DMAREQ_ID of DMAMUX2_C0CR = ‘7’ SE of DMAMUX2_C0CR = ‘0’ EGE of DMAMUX2_C0CR = ‘0’ NBREQ of DMAMUX2_C0CR = ‘0’	Selects LPUART_TX BDMA request. Disables block synchronization. No event generation. Generates an event every BDMA transfer (Free-running mode).
DMAMUX2 SYNC7	DMAREQ_ID of DMAMUX2_C7CR = ‘0’ SE of DMAMUX2_C7CR = ‘0’ EGE of DMAMUX2_C7CR = ‘1’ NBREQ of DMAMUX2_C7CR = ‘0’	Selects of REQ_GEN0 as BDMA request. Disables block synchronization. Enables event generation. Generates an event every BDMA transfer (Free-running mode).
DMAMUX2 REQ_GEN0	SIG_ID of DMAMUX2_RG0CR = ‘0d24’ GPOL of DMAMUX2_RG0CR = ‘0b01’ GNBREQ of DMAMUX2_RG0CR = ‘0’ GE of DMAMUX2_RG0CR = ‘1’	Selects LPUART TX interrupt as trigger. Trigger on rising edge of the event. Generates only one BDMA request. Enables generator.

Table 45. BDMA and DMAMUX2 initialization sequence (DMAMUX2_INIT) (continued)

Peripherals	Register content	Related actions
BDMA - CH0	NDT bits of BDMA_CNDTR0 = DatNber	Number of data to transfer.
	PA of BDMA_CPAR0 = &LPUART1_TDR	Address of LPUART1_TDR.
	MA of BDMA_CMAR0 = &DatBuff	Address of memory buffer of SRAM4.
	DIR of BDMA_CCR0 = ‘1’	Read from memory.
	CIRC of BDMA_CCR0 = ‘0’	Circular mode disabled.
	PINC of BDMA_CCR0 = ‘0’	Peripheral increment disabled.
	MINC of BDMA_CCR0 = ‘1’	Memory increment enabled.
	PSIZE of BDMA_CCR0 = ‘0’	Peripheral size = 8 bits.
	MSIZE of BDMA_CCR0 = ‘1’	Memory size = 8 bits.
	MEM2MEM of BDMA_CCR0 = ‘0’	Memory to memory disabled.
BDMA - CH7	NDT bits of BDMA_CNDTR7 = ‘1’	Only one data transferred.
	PA of BDMA_CPAR7 = &LPUART1_ICR	Address of LPUART1_ICR (Interrupt Flag Clear Reg.).
	MA of BDMA_CMAR7 = &DatClrTC	Address of a variable located into SRAM4. This variable must contain 0x0040 in order to clear the TC flag.
	DIR of BDMA_CCR7 = ‘1’	Read from memory.
	CIRC of BDMA_CCR7 = ‘0’	Circular mode disabled.
	PINC of BDMA_CCR7 = ‘0’	Peripheral increment disabled.
	MINC of BDMA_CCR7 = ‘1’	Memory increment disabled.
	PSIZE of BDMA_CCR7 = 2	Peripheral size = 32 bits.
	MSIZE of BDMA_CCR7 = 2	Memory size = 32 bits.
	MEM2MEM of BDMA_CCR7 = ‘0’	Memory to memory disabled.

LPTIM4 programming

When LPTIM4 wakeup event occurs, the CPU reboots and D3 domain mode is also set to Run mode.

An interrupt issued by LPTIM4 is pending on the CPU NVIC. LPTIM4 interrupt handler must acknowledge this LPTIM4 interrupt by writing ARRMCF bit in LPTIM4_ICR register to ‘1’ (LPTIM4_Ack).

LPUART programming

In the use-case described herein, the capability of the LPUART1 to request the kernel clock according to some events is not used.

LPUART1 is programmed so that is generates a BDMA request when its TX-FIFO is not full.

LPUART1 also generates an interrupt when the TX-FIFO and its transmit shift register are empty. This interrupt is used to switch the D3 domain to Stop mode.

Table 46 gives the key settings concerning the handling of Stop mode for LPUART1.

Table 46. LPUART1 Initial programming (LPUART1_INIT)

Register content	Related actions
FIFOEN of LPUART1_CR1 = ‘1’	Enables FIFO. BDMA will then use TXFNF (TXFIFO Not Full) flag for generating the BDMA requests.
TCIE of LPUART1_CR1 = ‘0’	Disables interrupt when the transmit buffer is empty.
UE of LPUART1_CR1 = ‘1’	Enables BDMA.
TE of LPUART1_CR1 = ‘1’	Enables the LPUART1.
TXE of LPUART1_CR1 = ‘1’	Enables transmission.
DMAT of LPUART1_CR3 = ‘1’	Enables the BDMA mode for transmission.

Respect the sequence described in *Table 47* to enable LPUART1.

Table 47. LPUART1 Initial programming (LPUART1_Start)

Register content	Related actions
TCCF of LPUART1_ICR = ‘1’	Clears the TC flag, to avoid immediate interrupt generation, which would clear the D3_PendClear[1] in EXTI.
TCIE of LPUART1_CR1 = ‘1’	Enables interrupt when the transmit buffer is empty.

6.3.3 Overall description of the low-power application example based on LPUART1 transmission

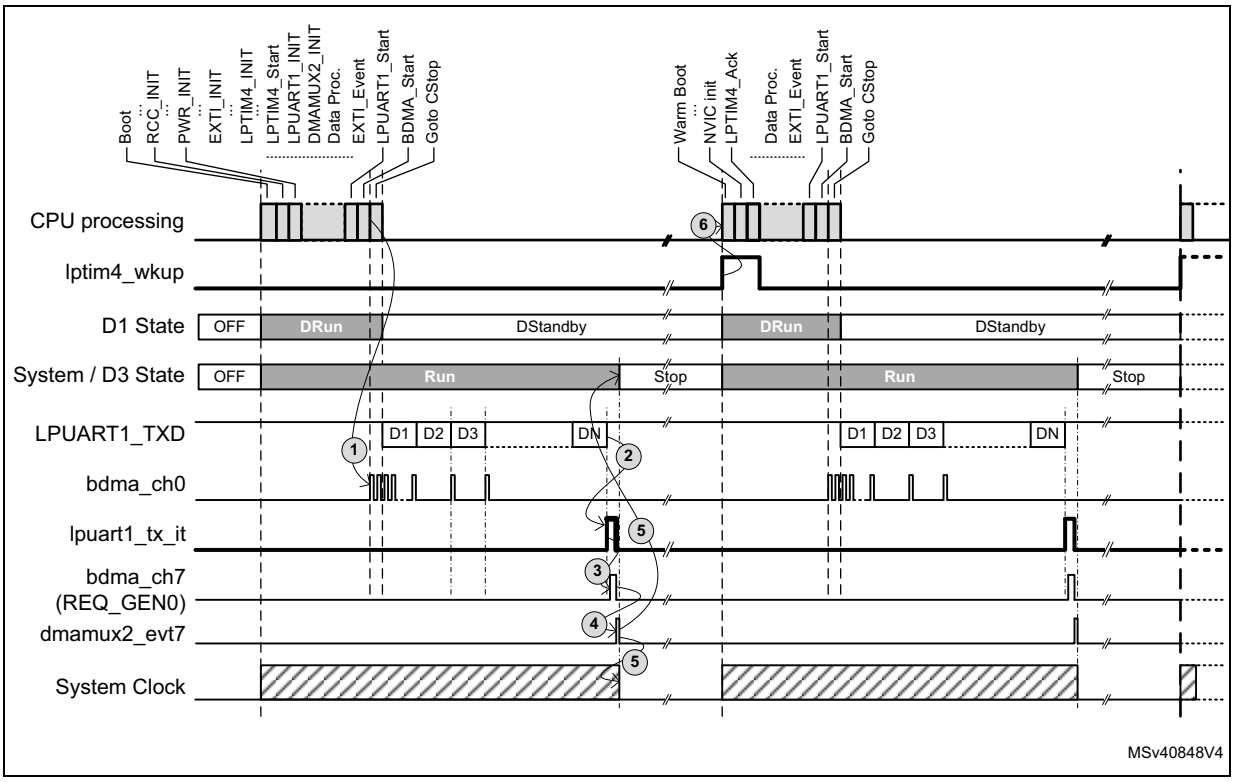
After a Power-on reset, the CPU perform the following operations:

1. Boot sequence (not described here).
2. Full initialization of RCC, PWR, EXTI, LPUART1, GPIOs, LPTIM4, DMAMUX2, BDMA and NVIC.
   Only the relevant steps of RCC, EXTI, PWR, LPUART1, BDMA and DMAMUX2 initialization related to the Autonomous mode are described herein. Refer to the previous sections for additional details.
3. The CPU processes the data to be transferred and copies them to SRAM4.
4. The CPU generates a wakeup event (EXTI_Event) to maintain D3 in Run mode when D1 enters DStandby.
5. The CPU enables the BDMA to start LPUART transmission and goes to Stop mode. As it is allowed to do so, D1 domain enters DStandby while D3 remains in Run mode. The data stored in SRAM4 are retained while the D1 domain is in DStandby mode.
6. As soon as the BDMA is enabled, it serves the request from LPUART1 in order to fill its TX-FIFO. In parallel, serial data transmission can start.
7. When the expected amount of data has been transmitted (NDT bits of BDMA_CNDTR0 set to 0), the BDMA no longer provides data to the LPUART1. The LPUART1 generates an interrupt when the TX-FIFO and the transmit buffer are empty.
8. This interrupt triggers DMAMUX2 REQ_GEN0, thus activating a data transfer via BDMA channel 7 (BDMA_Ch7). This transfer clears LPUART1 TC flag, and the **lpuart1_tx_it** is reset to ‘0’.
9. The end of this transfer triggers a dmamux2_evt7 signal which is used to clear the wakeup request generated by the CPU.
10. As a consequence, the D3 domain (i.e. the system) enters Stop mode and the system clock is gated. LPTIM4 still operates since it uses **ck_lsi** clock.

11. LPTIM4 **lptim4_wkup** interrupt wakes up the system. The device exits from Stop mode with the HSI clock. The CPU must restore the proper clock configuration during the warm re-boot sequence and perform the following tasks:
    d) Acknowledge LPTIM4 wakeup interrupt,
    e) Process the next data block and transfers them to SRAM4,
    f) Generate again a wakeup event for D3 domain,
    g) Start the BDMA.
    h) Go back to CStop mode.

Note: The CPU does not need to initialize BDMA, DMAMUX2 and LPUART1 again.

Figure 37. Timing diagram of LPUART1 transmission with D3 domain in Autonomous mode



6.3.4 Alternate implementations

More power efficient implementations are also possible. As an example the system clock can be stopped once the data have been transferred to LPUART1 TX-FIFO, instead of remaining activated during the whole transmission as in the example presented above. In this case, the LPUART1 must use ck_hsi or ck_csi as kernel clock when the system switches from Run to Stop mode. LPUART1 must be programmed to wake up D3 domain when its TX-FIFO in almost empty. This asynchronous interrupt can be used as trigger by the REQ_GENx of the DMAMUX2, which will perform a given number (e.g. 14) of data transfers to LPUART1_TDR and then switch back the D3 domain to Stop mode. This implementation is possible because the LPUART1 can request the kernel clock as long as the TX-FIFO and transmit buffer are not empty.

6.4 Other low-power applications

Other peripherals located in D3 domain, such as I2C4, SPI6, SAI4 or ADC3, can be used to implement low-power applications.

7 Reset and Clock Control (RCC)

The RCC block manages the clock and reset generation for the whole microcontroller.

The RCC block is located in the D3 domain (refer to *Section 5: Power control (PWR)* for a detailed description).

The operating modes this section refers to are defined in *Section 5.6.1: Operating modes* of the PWR block.

7.1 RCC main features

Reset block

- Generation of local and system reset
- Bidirectional pin reset allowing to reset the microcontroller or external devices
- Hold Boot function
- WWDG reset supported

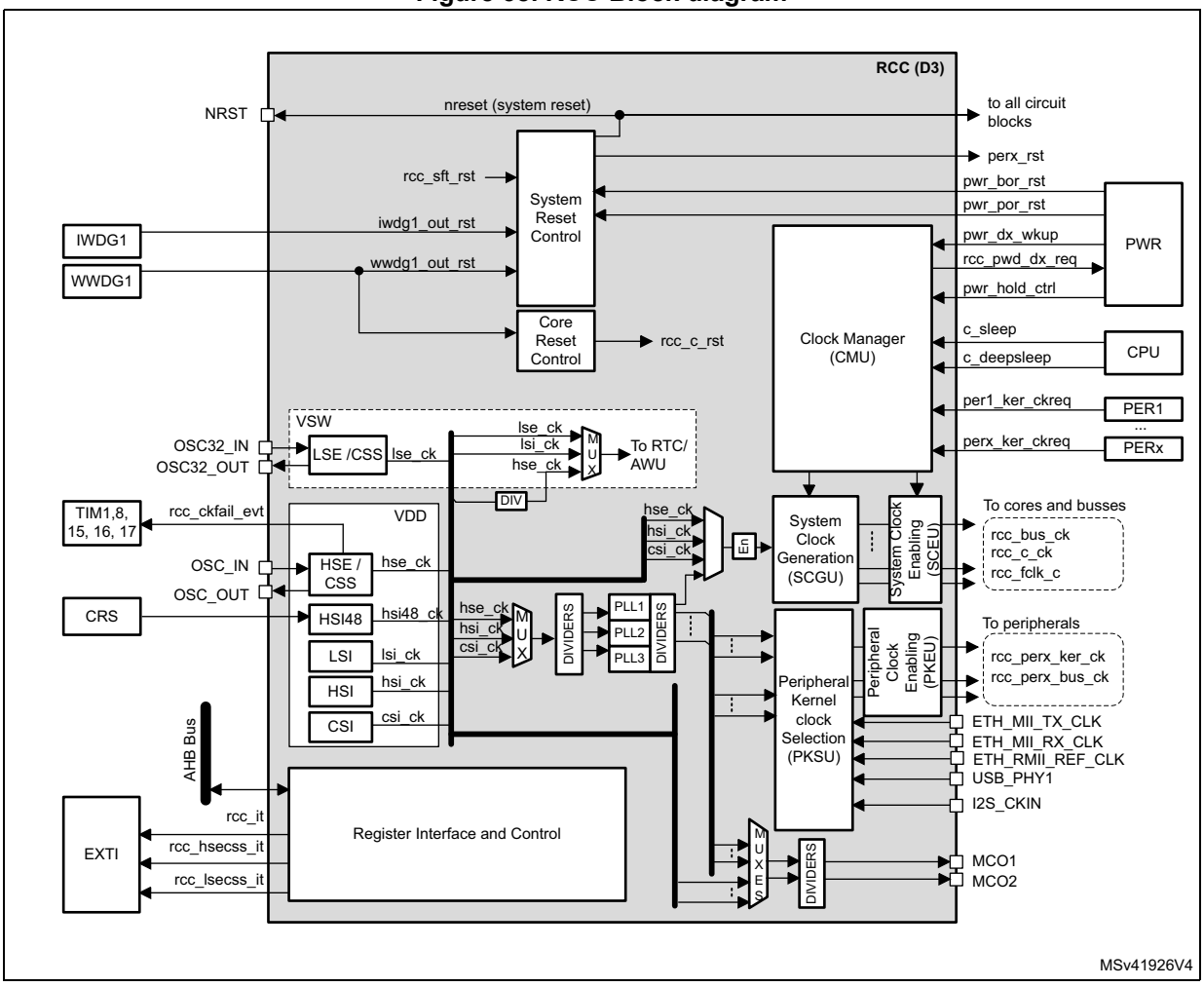
Clock generation block

- Generation and dispatching of clocks for the complete device
- 3 separate PLLs using integer or fractional ratios
- Possibility to change the PLL fractional ratios on-the-fly
- Smart clock gating to reduce power dissipation
- 2 external oscillators:
  - High-speed external oscillator (HSE) supporting a wide range of crystals from 4 to 48 MHz frequency
  - Low-speed external oscillator (LSE) for the 32 kHz crystals
- 4 internal oscillators
  - High-speed internal oscillator (HSI)
  - 48 MHz RC oscillator (HSI48)
  - Low-power Internal oscillator (CSI)
  - Low-speed internal oscillator (LSI)
- Buffered clock outputs for external devices
- Generation of two types of interrupts lines:
  - Dedicated interrupt lines for clock security management
  - One general interrupt line for other events
- Clock generation handling in Stop and Standby mode
- D3 domain Autonomous mode

7.2 RCC block diagram

Figure 38 shows the RCC block diagram.

Figure 38. RCC Block diagram



7.3 RCC pins and internal signals

Table 48 lists the RCC inputs and output signals connected to package pins or balls.

Table 48. RCC input/output signals connected to package pins or balls

Signal name	Signal type	Description
NRST	I/O	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input

Table 48. RCC input/output signals connected to package pins or balls (continued)

Signal name	Signal type	Description
OSC_OUT	O	System oscillator output
MCO1	O	Output clock 1 for external devices
MCO2	O	Output clock 2 for external devices
I2S_CKIN	I	External kernel clock input for digital audio interfaces: SPI/I2S, SAI, and DFSDM
ETH_MII_TX_CLK	I	External TX clock provided by the Ethernet MII interface
ETH_MII_RX_CLK	I	External RX clock provided by the Ethernet MII interface
ETH_RMII_REF_CLK	I	External reference clock provided by the Ethernet RMII interface
USB_PHY1	I	USB clock input provided by the external USB PHY (OTG_HS_ULPI_CK)

The RCC exchanges a lot of internal signals with all components of the product, for that reason, the *Table 48* only shows the most significant internal signals.

Table 49. RCC internal input/output signals

New Signal name	Signal type	Description
rcc_it	O	General interrupt request line
rcc_hsecss_it	O	HSE clock security failure interrupt
rcc_lsecss_it	O	LSE clock security failure interrupt
rcc_ckfail_evt	O	Event indicating that a HSE clock security failure is detected. This signal is connected to TIMERS
nreset	I/O	System reset
iwdg1_out_rst	I	Reset line driven by the IWDG1, indicating that a timeout occurred.
wwdg1_out_rst	I	Reset line driven by the WWDG1, indicating that a timeout occurred.
pwr_bor_rst	I	Brownout reset generated by the PWR block
pwr_por_rst	I	Power-on reset generated by the PWR block
pwr_vsw_rst	I	Power-on reset of the VSW domain generated by the PWR block
rcc_perx_rst	O	Reset generated by the RCC for the peripherals.
pwr_d[3:1]_wkup	I	Wake-up domain request generated by the PWR. Generally used to restore the clocks a domain when this domain exits from DStop
rcc_pwd_d[3:1]_req	O	Low-Power request generated by the RCC. Generally used to ask to the PWR to set a domain into low-power mode, when a domain is in DStop.
pwr_hold_ctrl	I	Signals generated by the PWR, in order to set the processor into CStop when exiting from system Stop mode.
c_sleep	I	Signal generated by the CPU, indicating if the CPU is in CRun, CSleep or CStop.
c_deepsleep	I	

Table 49. RCC internal input/output signals (continued)

New Signal name	Signal type	Description
perx_ker_ckreq	I	Signal generated by some peripherals in order to request the activation of their kernel clock.
rcc_perx_ker_ck	O	Kernel clock signals generated by the RCC, for some peripherals.
rcc_perx_bus_ck	O	Bus interface clock signals generated by the RCC for peripherals.
rcc_bus_ck	O	Clocks for APB (rcc_apb_ck), AHB (rcc_ahb_ck) and AXI (rcc_axi_ck) bridges generated by the RCC.
rcc_c_ck	O	Clock for the CPU, generated by the RCC.
rcc_fclk_c	O	

7.4 RCC reset block functional description

Several sources can generate a reset:

- An external device via NRST pin
- A failure on the supply voltage applied to V_{DD}
- A watchdog timeout
- A software command

The reset scope depends on the source that generates the reset. Three reset categories exist:

- Power-on/off reset
- System reset
- Local resets

7.4.1 Power-on/off reset

The power-on/off reset (**pwr_por_rst**) is generated by the power controller block (PWR). It is activated when the input voltage (V_{DD}) is below a threshold level. This is the most complete reset since it resets the whole circuit, except the backup domain. The power-on/off reset function can be disabled through PDR_ON pin (see *Section 5.5: Power supply supervision*).

Refer to *Table 50: Reset distribution summary* for details.

7.4.2 System reset

A system reset (**nreset)** resets all registers to their default values except for the reset status flags in the RCC_RSR (or RCC_C1_RSR) register, the debug features, the Flash memory and the Backup domain registers.

A system reset can be generated from one of the following sources:

- A reset from NRST pin (external reset)
- A reset from the power-on/off reset block (**pwr_por_rst**)
- A reset from the brownout reset block (**pwr_bor_rst**)

  Refer to *Section 5.5.2: Brownout reset (BOR)* for a detailed description of the BOR function.
- A reset from the independent watchdogs (**iwdg1_out_rst**)
- A software reset from the Cortex®-M7 core

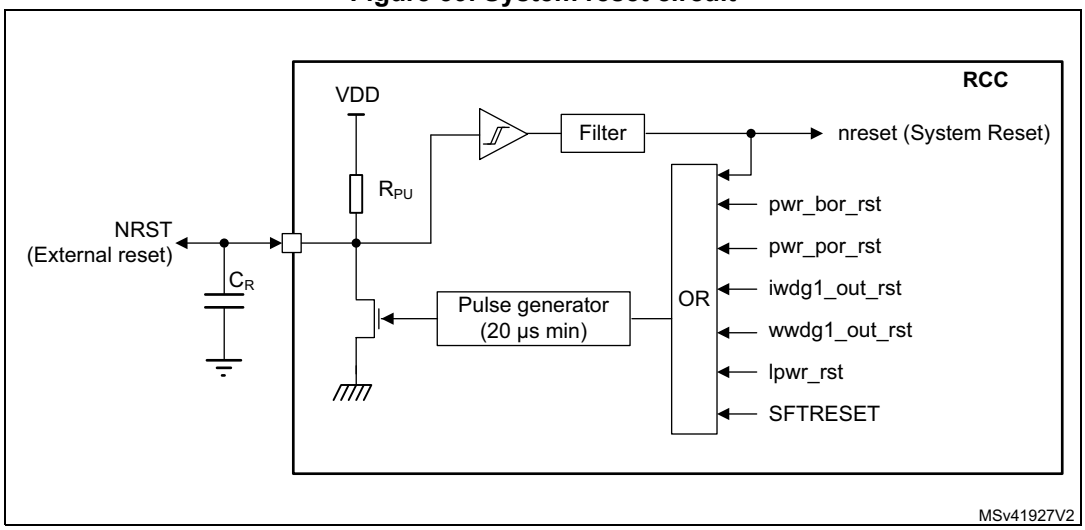
  It is generated via the SYSRESETREQ signal issued by the Cortex®-M7 core. This signal is also named SFTRESET in this document.
- A reset from the window watchdogs depending on WWDG configuration (**wwdg1_out_rst**)
- A reset from the low-power mode security reset, depending on option byte configuration (**lpwr[2:1]_rst**)

Note: The SYSRESETREQ bit in Cortex®-M7 Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the Cortex®-M7 with FPU technical reference manual for more details (see http://infocenter.arm.com*).*

As shown in *Figure 39*, some internal sources (such as **pwr_por_rst**, **pwr_bor_rst**, **iwdg1_out_rst**) perform a system reset of the circuit, which is also propagated to the NRST pin to reset the connected external devices. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted Low.

Note: It is not recommended to let the NRST pin unconnected. When it is not used, connect this pin to ground via a 10 to 100 nF *capacitor (C_R in Figure 39*).

Figure 39. System reset circuit



7.4.3 Local resets

CPU reset

The CPU can reset itself by means of the CPURST bit in *RCC AHB3 Reset Register (RCC_AHB3RSTR)*.

Domain reset

Some resets also dependent on the domain status. For example, when D1 domain exits from DStandby, it is reset (**d1_rst**). The same mechanism applies to D2.

When the system exits from Standby mode, a **stby_rst** reset is applied. The **stby_rst** signal generates a reset of the complete V_{CORE} domain as long the V_{CORE} voltage provided by the internal regulator is not valid.

Table 50 gives a detailed overview of reset sources and scopes.

Table 50. Reset distribution summary

Reset source	Reset name	D1 CPU	D1 Interconnect	D1 Peripherals	D1 Debug	WWDG1	D2 Interconnect	D2 Peripherals	D3 Peripherals	IWDG1	FLASH	RTC domain	Backup RAM	System Supply	NRST pin	Comments
Pin	NRST	x	x	x	-	x	x	x	x	x	-	-	-	-	x	– Resets D1 and D2 domains, and all their peripherals – Resets D3 domain peripherals – Resets V_{DD} domain: IWDG1, LDO... – Debug features, Flash memory, RTC and backup RAM are not reset
PWR	pwr_bor_rst	x	x	x	-	x	x	x	x	x	-	-	-	-	x	– Same as pin reset. The pin is asserted as well.
	pwr_por_rst	x	x	x	x	x	x	x	x	x	x	-	-	x	x	– Same as **pwr_bor_rst** reset, plus: Reset of the Flash memory digital block (including the option byte loading). Reset of the debug block
	lpwr_rst	x	x	x	-	x	x	x	x	x	-	-	-	-	x	– The low-power mode security reset has the same scope than **pwr_por_rst.** Refer to *Section 7.4.5: Low-power mode security reset (lpwr_rst)* for additional information.

Table 50. Reset distribution summary (continued)

Reset source	Reset name	D1 CPU	D1 Interconnect	D1 Peripherals	D1 Debug	WWDG1	D2 Interconnect	D2 Peripherals	D3 Peripherals	IWDG1	FLASH	RTC domain	Backup RAM	System Supply	NRST pin	Comments
RCC	BDRST	-	-	-	-	-	-	-	-	-	-	x	-	-	-	– The backup domain reset can be triggered by software. Refer to *Section 7.4.6: Backup domain reset* for additional information
	d1_rst	x	x	x	x	x	-	-	-	-	-	-	-	-	-	– Resets D1 domain, and all its peripherals, when the domain exits DStandby mode.
	d2_rst	-	-	-	-	-	x	x	-	-	-	-	-	-	-	– Resets D2 domain, and all its peripherals, when the domain exits DStandby mode.
	stby_rst	x	x	x	x	x	x	x	x	-	-	-	-	-	-	– When the device exits Standby mode, a reset of the complete V_{CORE} domain is performed as long the V_{CORE} voltage is not valid. The V_{CORE} is supplied by the internal regulator. NRST signal is not asserted.
	CPURST	x	-	-	-	x	-	-	-	-	-	-	-	-	-	– This reset is generated by software through the bit located into *RCC AHB3 Reset Register (RCC_AHB3RSTR)*. – Resets the CPU, and the WWDG1 block
CPU	SFTRESET	x	x	x	-	x	x	x	x	x	-	-	-	-	x	– This reset is generated by software when writing SYSRESETREQ bit located into AIRCR register of the Cortex®-M7 core. – Same scope as **pwr_bor_rst** reset.
Backup domain	pwr_vsw_rst	-	-	-	-	-	-	-	-	-	-	x	-	-	-	– This reset is generated by the backup domain when the V_{SW} supply voltage is outside the operating range.
IWDG1	iwdg1_out_rst	x	x	x	-	x	x	x	x	x	-	-	-	-	x	– Same as **pwr_bor_rst** reset.
WWDG1	wwdg1_out_rst	x	x	x	-	x	x	x	x	x	-	-	-	-	x	– Same as **pwr_bor_rst** reset.

7.4.4 Reset source identification

The CPU can identify the reset source by checking the reset flags in the RCC_RSR (or RCC_C1_RSR) register.

The CPU can reset the flags by setting RMVF bit.

Table 51 shows how the status bits of RCC_RSR (or RCC_C1_RSR) register behaves, according to the situation that generated the reset. For example when an IWDG1 timeout occurs (line #10), if the CPU is reading the RCC_RSR (or RCC_C1_RSR) register during the boot phase, both PINRSTF and IWDG1RSTF bits are set, indicating that the IWDG1 also generated a pin reset.

Table 51. Reset source identification (RCC_RSR)(1)

#	Situations Generating a Reset	LPWRRSTF	WWDG1RSTF	IWDG1RSTF	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	D2RSTF	D1RSTF	CPURSTF
1	Power-on reset (pwr_por_rst)	0	0	0	0	1	1	1	1	1	1
2	Pin reset (NRST)	0	0	0	0	0	1	0	0	0	1
3	Brownout reset (pwr_bor_rst)	0	0	0	0	0	1	1	0	0	1
4	System reset generated by CPU (SFTRESET)	0	0	0	1	0	1	0	0	0	1
5	CPU reset (CPURST)	0	0	0	0	0	0	0	0	0	1
6	WWDG1 reset (wwdg1_out_rst)	0	1	0	0	0	**1**	0	0	0	**1**
8	IWDG1 reset (iwdg1_out_rst)	0	0	1	0	0	1	0	0	0	1
10	D1 exits DStandby mode	0	0	0	0	0	0	0	0	1	0
11	D2 exits DStandby mode	0	0	0	0	0	0	0	1	0	0
12	D1 erroneously enters DStandby mode or CPU erroneously enters CStop mode	1	0	0	0	0	1	0	0	0	1

1. Grayed cells highlight the register bits that are set.

7.4.5 Low-power mode security reset (lpwr_rst)

To prevent critical applications from mistakenly enter a low-power mode, two low-power mode security resets are available. When enabled through nRST_STOP_D1 option bytes, a system reset is generated if the following conditions are met:

- CPU accidentally enters CStop mode

  This type of reset is enabled by resetting nRST_STOP_D1 user option byte. In this case, whenever the CPU CStop mode entry sequence is successfully executed, a system reset is generated.
- D1 domain accidentally enters DStandby mode

  This type of reset is enabled by resetting nRST_STDBY_D1 user option byte. In this case, whenever a D1 domain DStandby mode entry sequence is successfully executed, a system reset is generated.

LPWRRSTF bits in *RCC Reset Status Register (RCC_RSR)* indicates that a low-power mode security reset occurred (see line #12 in *Table 51*).

lpwr_rst is activated when a low-power mode security reset due to D1 or CPU occurred.

Refer to *Section 3.4: FLASH option bytes* for additional information.

7.4.6 Backup domain reset

A backup domain reset is generated when one of the following events occurs:

- A software reset, triggered by setting BDRST bit in the *RCC Backup Domain Control Register (RCC_BDCR)*. All RTC registers and the RCC_BDCR register are reset to their default values. The backup RAM is not affected.
- V_{SW} voltage is outside the operating range. All RTC registers and the RCC_BDCR register are reset to their default values. In this case the content of the backup RAM is no longer valid.

There are two ways to reset the backup RAM:

- through the Flash memory interface by requesting a protection level change from 1 to 0
- when a tamper event occurs.

Refer to *Section 5.4.4: Backup domain* section of PWR block for additional information.

7.4.7 Power-on and wakeup sequences

For detailed diagrams refer to *Section 5.4.1: System supply startup* in the PWR section.

The time interval between the event which exits the product from a low-power and the moment where the CPU is able to execute code, depends on the system state and on its configuration. *Figure 40* shows the most usual examples.

Power-on wakeup sequence

The power-on wakeup sequence shown in *Figure 40* gives the most significant phases of the power-on sequence. It is the longest sequence since the circuit was not powered. Note that this sequence remains unchanged, whatever V_{BAT} was present or not.

Boot from pin reset

When a pin reset occurs, V_{DD} is still present. As a results:

- The regulator settling time is faster since the reference voltage is already stable.
- The HSI restart delay may be needed if the HSI was not enabled when the NRST occurred, otherwise this restart delay phase is skipped.
- The Flash memory power recovery delay can also be skipped if the Flash memory was enabled when the NRST occurred.

Note: The boot sequence is similar for pwr_bor_rst, lpwr_rst, STFxRESET, iwdg1_out_rst and wwdg1_out_rst (if WW1RSC = '1').

Boot from system Standby

When waking up from system Standby, the reference voltage is stable since V_{DD} has not been removed. As a result, the regulator settling time is fast. Since V_{CORE} was not present, the restart delay for the HSI, the Flash memory power recovery and the option byte reloading cannot be skipped.

Restart from system Stop

When restarting from system Stop, V_{DD} is still present. As a result, the sequence is mainly composed of two steps:

1. Regulator settling time to reach VOS3 (default voltage)
2. HSI/CSI restart delay. This step can be skipped if HSIKERON or CSIKERON bit, in *RCC Source Control Register (RCC_CR)* is set to '1'.

Boot from domain DStandby

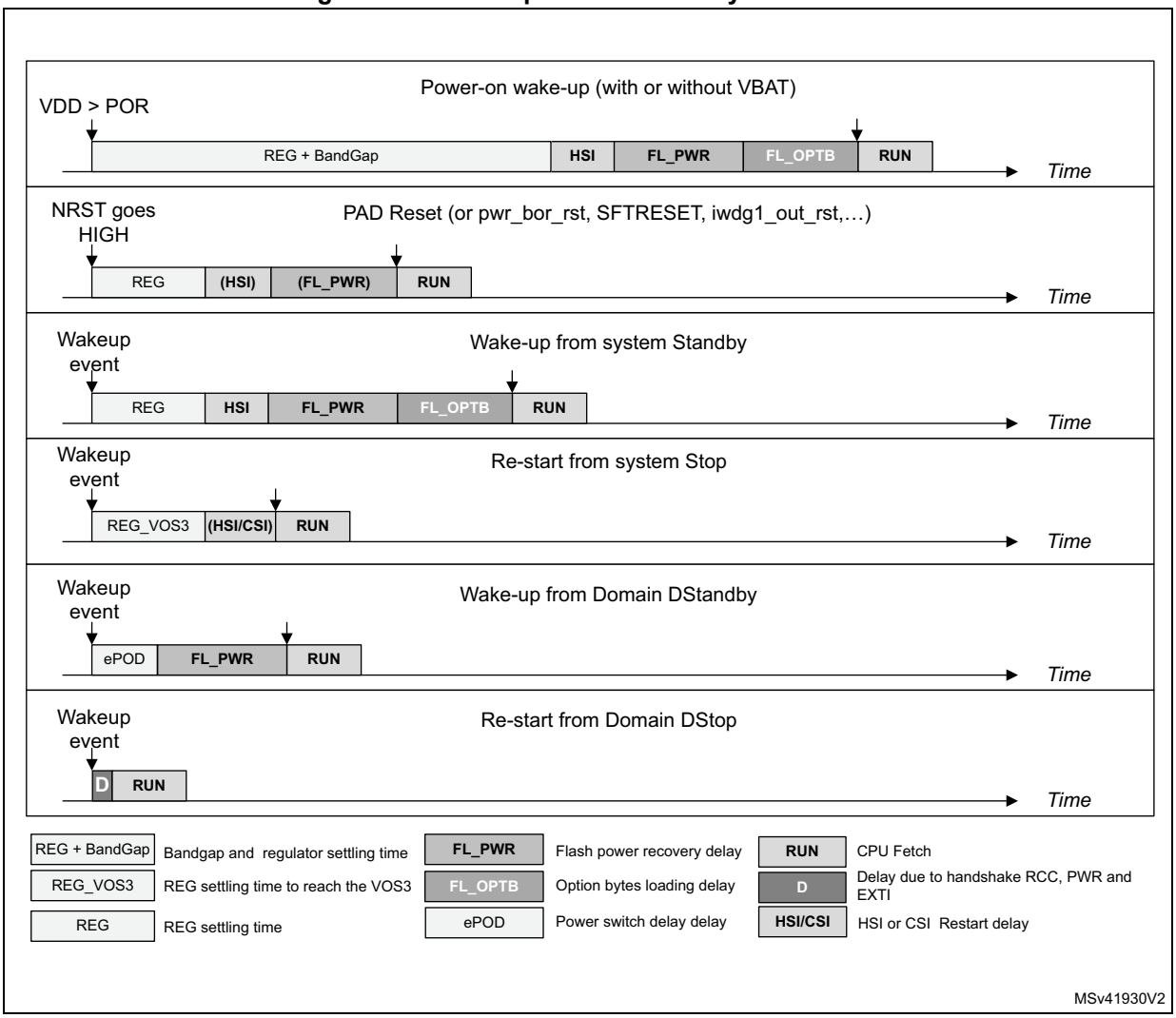
The boot sequence of a domain from domain DStandby is mainly composed of two steps:

1. The power switch settling time (the regulator is already activated).
2. The Flash memory power recovery.

Restart from domain DStop

The restart sequence of a domain from domain DStop is mainly composed of the handshake between the RCC, EXTI and PWR blocks.

Figure 40. Boot sequences versus system states



7.5 RCC clock block functional description

The RCC provides a wide choice of clock generators:

- HSI (High-speed internal oscillator) clock: ~ 8, 16, 32 or 64 MHz
- HSE (High-speed external oscillator) clock: 4 to 48 MHz
- LSE (Low-speed external oscillator) clock: 32 kHz
- LSI (Low-speed internal oscillator) clock: ~ 32 kHz
- CSI (Low-power internal oscillator) clock: ~4 MHz
- HSI48 (High-speed 48 MHz internal oscillator) clock: ~48 MHz

It offers a high flexibility for the application to select the appropriate clock for CPU and peripherals, in particular for peripherals that require a specific clock such as Ethernet, USB OTG-FS and HS, SPI/I2S, SAI and SDMMC.

To optimize the power consumption, each clock source can be switched ON or OFF independently.

The RCC provides up to 3 PLLs; each of them can be configured with integer or fractional ratios.

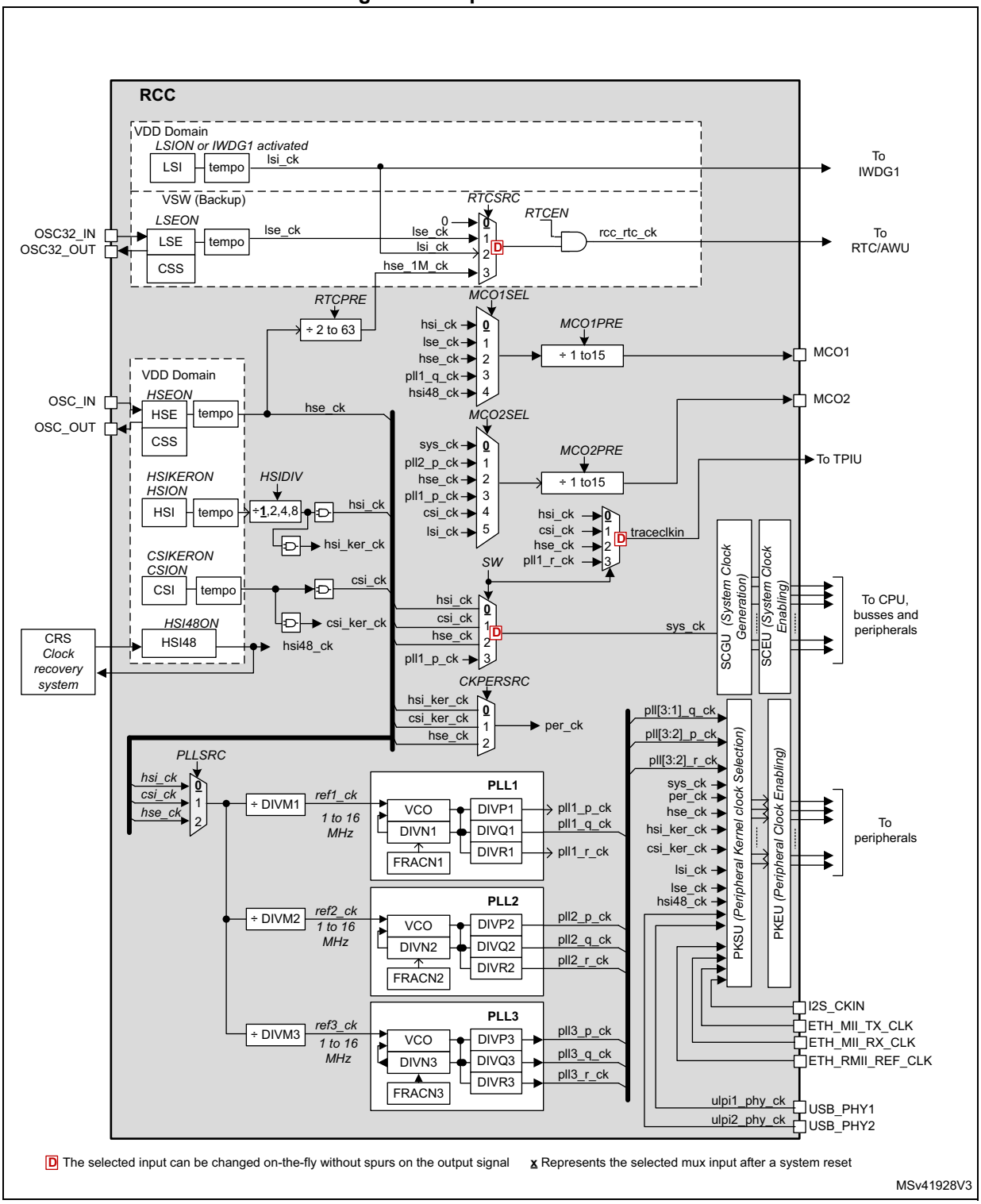
As shown in the *Figure 41*, the RCC offers 2 clock outputs (MCO1 and MCO2), with a great flexibility on the clock selection and frequency adjustment.

The SCGU block (System Clock Generation Unit) contains several prescalers used to configure the CPU and bus matrix clock frequencies.

The PKSU block (Peripheral Kernel clock Selection Unit) provides several dynamic switches allowing a large choice of kernel clock distribution to peripherals.

The PKEU (Peripheral Kernel clock Enable Unit) and SCEU (System Clock Enable Unit) blocks perform the peripheral kernel clock gating, and the bus interface/cores/bus matrix clock gating, respectively.

Figure 41. Top-level clock tree



7.5.1 Clock naming convention

The RCC provides clocks to the complete circuit. To avoid misunderstanding, the following terms are used in this document:

- Peripheral clocks

  The peripheral clocks are the clocks provided by the RCC to the peripherals. Two kinds of clock are available:

  – The bus interface clocks

  – The kernel clocks

  A peripheral receives from the RCC a bus interface clock in order to access its registers, and thus control the peripheral operation. This clock is generally the AHB, APB or AXI clock depending on which bus the peripheral is connected to. Some peripherals only need a bus interface clock (e.g. RNG, TIMx).

  Some peripherals also require a dedicated clock to handle the interface function. This clock is named “kernel clock”. As an example, peripherals such as SAI have to generate specific and accurate master clock frequencies, which require dedicated kernel clock frequencies. Another advantage of decoupling the bus interface clock from the specific interface needs, is that the bus clock can be changed without reprogramming the peripheral.

- CPU clocks

  The CPU clock is the clock provided to the CPU. It is derived from the system clock (**sys_ck**).

- Bus matrix clocks

  The bus matrix clocks are the clocks provided to the different bridges (APB, AHB or AXI). These clocks are derived from the system clock (**sys_ck**).

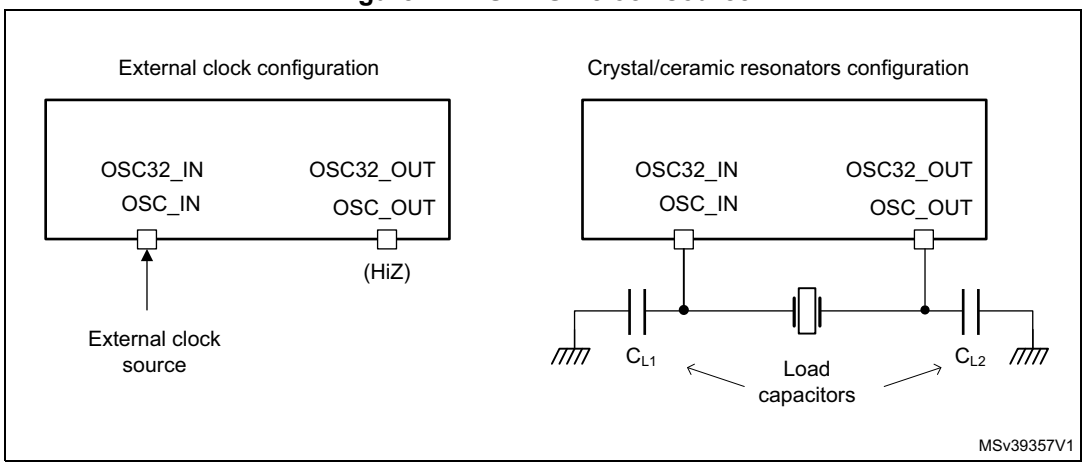
7.5.2 Oscillators description

HSE oscillator

The HSE block can generate a clock from two possible sources:

- External crystal/ceramic resonator
- External clock source

Figure 42. HSE/LSE clock source



External clock source (HSE bypass)

In this mode, an external clock source must be provided to OSC_IN pin. This mode is selected by setting the HSEBYP and HSEON bits of the *RCC Source Control Register (RCC_CR)* to '1'. The external clock source (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left HI-Z (see *Figure 42*).

External crystal/ceramic resonator

The oscillator is enabled by setting the HSEBYP bit to '0' and HSEON bit to '1'.

The HSE can be used when the product requires a very accurate high-speed clock.

The associated hardware configuration is shown in *Figure 42*: the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag of the *RCC Source Control Register (RCC_CR)*, indicates whether the HSE oscillator is stable or not. At startup, the **hse_ck** clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC Clock Source Interrupt Enable Register (RCC_CIER)*.

The HSE can be switched ON and OFF through the HSEON bit. Note that the HSE cannot be switched OFF if one of the two conditions is met:

- The HSE is used directly (via software mux) as system clock
- The HSE is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSEON bit to '0'.

The HSE is automatically disabled by hardware, when the system enters Stop or Standby mode (refer to *Section 7.5.7: Handling clock generators in Stop and Standby mode* for additional information).

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.

LSE oscillator

The LSE block can generate a clock from two possible sources:

- External crystal/ceramic resonator
- External user clock

External clock source (LSE bypass)

In this mode, an external clock source must be provided to OSC32_IN pin. The input clock can have a frequency up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits of *RCC Backup Domain Control Register (RCC_BDCR)* to '1'. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left HI-Z (see *Figure 42*).

External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions.

The LSERDY flag of the *RCC Backup Domain Control Register (RCC_BDCR)* indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC Clock Source Interrupt Enable Register (RCC_CIER)*.

The LSE oscillator is switched ON and OFF using the LSEON bit. The LSE remains enabled when the system enters Stop or Standby mode.

In addition, the LSE clock can be driven to the MCO1 output and used as clock source for other application components.

The LSE also offers a programmable driving capability (LSEDRV[1:0]) that can be used to modulate the amplifier driving capability. The driving capability can be changed dynamically from high drive to medium high drive, and then to medium low drive.

HSI oscillator

The HSI block provides the default clock to the product.

The HSI is a high-speed internal RC oscillator which can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32 or 64 MHz. This predivider is controlled by the HSIDIV.

The HSI advantages are the following:

- Low-cost clock source since no external crystal is required
- Faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The HSI can be switched ON and OFF using the HSION bit. Note that the HSI cannot be switched OFF if one of the two conditions is met:

- The HSI is used directly (via software mux) as system clock
- The HSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSION bit to ‘0’.

Note that the HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to ‘1’). In that case the hardware does not update the HSIDIV with the new value. However it is possible to change the HSIDIV if the HSI is used directly as system clock.

The HSIRDY flag indicates if the HSI is stable or not. At startup, the HSI output clock is not released until this bit is set by hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to *Section : CSS on HSE*). The HSI can be disabled or not when the system enters Stop mode, please refer to *Section 7.5.7: Handling clock generators in Stop and Standby mode* for additional information.

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals, the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency accuracy.

Note: The HSI can remain enabled when the system is in Stop mode (see Section 7.5.7 for additional information).

HSION, HSIRDY and HSIDIV bits are located in the RCC Source Control Register (RCC_CR).

HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to improve accuracy (refer to the product datasheet for more information).

After a power-on reset, the factory calibration value is loaded in the HSICAL[11:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[5:0] bits.

Note: HSICAL[11:0] and HSITRIM[5:0] bits are located in the RCC Internal Clock Source Calibration Register (RCC_ICSCR).

CSI oscillator

The CSI is a low-power RC oscillator which can be used directly as system clock, peripheral clock, or PLL input.

The CSI advantages are the following:

- Low-cost clock source since no external crystal is required
- Faster startup time than HSE (a few microseconds)
- Very low-power consumption,

The CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

CSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The CSI can be switched ON and OFF through the CSION bit. The CSIRDY flag indicates whether the CSI is stable or not. At startup, the CSI output clock is not released until this bit is set by hardware.

The CSI cannot be switched OFF if one of the two conditions is met:

- The CSI is used directly (via software mux) as system clock
- The CSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the CSION bit to ‘0’.

The CSI can be disabled or not when the system enters Stop mode (refer to *Section 7.5.7: Handling clock generators in Stop and Standby mode* for additional information).

In addition, the CSI clock can be driven to the MCO2 output and used as clock source for other application components.

Even if the CSI settling time is faster than the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application has to take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency precision.

Note: CSION and CSIRDY bits are located in the RCC Source Control Register (RCC_CR).

CSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by STMicroelectronics to achieve improve (refer to the product datasheet for more information).

After reset, the factory calibration value is loaded in the CSICAL[7:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the CSI frequency using the CSITRIM[4:0] bits.

Note: CSICAL[7:0] and CSITRIM[4:0] bits are located into the RCC Internal Clock Source Calibration Register (RCC_ICSCR)

HSI48 oscillator

The HSI48 is an RC oscillator that delivers a 48 MHz clock that can be used directly as kernel clock for some peripherals.

The HSI48 oscillator mainly aims at providing a high precision clock to the USB peripheral by means of a special Clock Recovery System (CRS) circuitry, which could use the USB SOF signal, the LSE, or an external signal, to automatically adjust the oscillator frequency on-the-fly, in very small granularity.

The HSI48 oscillator is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, this oscillator is free running and thus subject to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACC_{HSI48} (refer to the product datasheet of the for more information).

For more details on how to configure and use the CRS, please refer to *Section 8: Clock recovery system (CRS)*.

The HSI48RDY flag indicates whether the HSI48 oscillator is stable or not. At startup, the HSI48 output clock is not released until this bit is set by hardware.

The HSI48 can be switched ON and OFF using the HSI48ON bit.

The HSI48 clock can also be driven to the MCO1 multiplexer and used as clock source for other application components.

Note: HSI48ON and HSI48RDY bits are located in the RCC Source Control Register (RCC_CR).

LSI oscillator

The LSI acts as a low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG) and Auto-Wakeup Unit (AWU). The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced ON and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode (refer to *Section 7.5.7: Handling clock generators in Stop and Standby mode* for additional information).

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the *RCC Clock Source Interrupt Enable Register (RCC_CIER)*.

In addition, the LSI clock can be driven to the MCO2 output and used as a clock source for other application components.

Note: Bits LSION and LSIRDY are located into the RCC Clock Control and Status Register (RCC_CSR).

7.5.3 Clock Security System (CSS)

CSS on HSE

The clock security system can be enabled by software via the HSECSSON bit. The HSECSSON bit can be enabled even when the HSEON is set to ‘0’.

The CSS on HSE is enabled by the hardware when the HSE is enabled and ready, and HSECSSON set to ‘1’.

The CSS on HSE is disabled when the HSE is disabled. As a result, this function does not work when the system is in Stop mode.

It is not possible to clear directly the HSECSSON bit by software.

The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode (see *Section 7.4.2: System reset*).

If a failure is detected on the HSE clock, the system automatically switches to the HSI in order to provide a safe clock. The HSE is then automatically disabled, a clock failure event is sent to the break inputs of advanced-control timers (TIM1, TIM8, TIM15, TIM16, and TIM17), and an interrupt is generated to inform the software about the failure (CSS interrupt: ***rcc_hsecss_it***), thus allowing the MCU to perform rescue operations. If the HSE output was used as clock source for PLLs when the failure occurred, the PLLs are also disabled.

If an HSE clock failure occurs when the CSS is enabled, the CSS generates an interrupt which causes the automatic generation of an NMI. The HSECSSF flag in *RCC Clock Source Interrupt Flag Register (RCC_CIFR)* is set to ‘1’ to allow the application to identify the failure source. The NMI routine is executed indefinitely until the HSECSSF bit is cleared. As a consequence, the application has to clear the HSECSSF flag in the NMI ISR by setting the HSECSSC bit in the *RCC Clock Source Interrupt Clear Register (RCC_CICR)*.

CSS on LSE

A clock security system on the LSE oscillator can be enabled by software by programming the LSECSSON bit in the *RCC Backup Domain Control Register (RCC_BDCR)*.

This bit can be disabled only by hardware when the following conditions are met:

- after a pwr_vsw_rst (V_{SW} software reset)
- or after a failure detection on LSE.

LSECSSON bit must be written after the LSE is enabled (LSEON bit set by software) and ready (LSERDY set by hardware), and after the RTC clock has been selected through the RTCSEL bit.

The CSS on LSE works in all modes (Run, Stop and Standby) except VBAT.

If an LSE failure is detected, the LSE clock is no more delivered to the RTC but the value of RTCSEL, LSECSSON and LSEON bits are not changed by the hardware.

A wakeup is generated in Standby mode. In other modes an interrupt (**rcc_lsecss_it**) can be sent to wake up the software. The software must then disable the LSECSSON bit, stop the defective LSE (clear LSEON bit), and can change the RTC clock source (no clock or LSI or HSE) through RTCSEL bits, or take any required action to secure the application.

7.5.4 Clock output generation (MCO1/MCO2)

Two micro-controller clock output (MCO) pins, MCO1 and MCO2, are available. A clock source can be selected for each output.The selected clock can be divided thanks to configurable prescaler (refer to *Figure 41* for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1[2:0], MCO2PRE[3:0] and MCO2[2:0] located in the *RCC Clock Configuration Register (RCC_CFGR)*.

The GPIO port corresponding to each MCO pin, has to be programmed in alternate function mode.

The clock provided to the MCOs outputs must not exceed the maximum pin speed (refer to the product datasheet for information on the supported pin speed).

7.5.5 PLL description

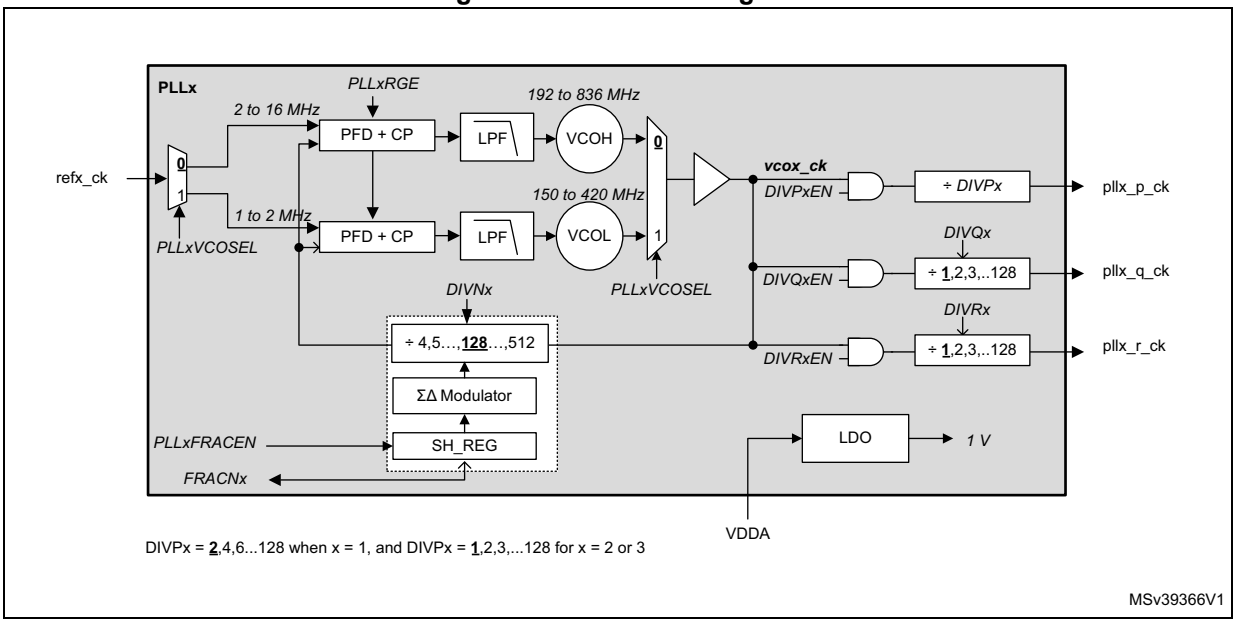
The RCC features three PLLs:

- A main PLL, PLL1, which is generally used to provide clocks to the CPU and to some peripherals.
- Two dedicated PLLs, PLL2 and PLL3, which are used to generate the kernel clock for peripherals.

The PLLs integrated into the RCC are completely independent. They offer the following features:

- Two embedded VCOs:
  - A wide-range VCO (VCOH)
  - A low-frequency VCO (VCOL)
- Input frequency range:
  - 1 to 2 MHz when VCOL is used
  - 2 to 16 MHz when VCOH is used
- Capability to work either in integer or Fractional mode
- 13-bit Sigma-Delta modulator, allowing to fine-tune the VCO frequency by steps of 11 to 0.3 ppm.
- The Sigma-Delta modulator can be updated on-the-fly, without generating frequency overshoots on PLLs outputs.
- Each PLL offer 3 outputs with post-dividers

Figure 43. PLL block diagram



The PLLs are controlled via RCC_PLLxDIVR, RCC_PLLxFRACR, RCC_PLLCFGR and RCC_CR registers.

The frequency of the reference clock provided to the PLLs (**refx_ck**) must range from 1 to 16 MHz. The user application has to program properly the DIVMx dividers of the *RCC PLLs Clock Source Selection Register (RCC_PLLCKSELR)* in order to match this condition. In addition, the PLLxRGE of the *RCC PLLs Configuration Register (RCC_PLLCFGR)* field must be set according to the reference input frequency to guarantee an optimal performance of the PLL.

The user application can then configure the proper VCO: if the frequency of the reference clock is lower or equal to 2 MHz, then VCOL must be selected, otherwise VCOH must be chosen. To reduce the power consumption, it is recommended to configure the VCO output to the lowest frequency.

DIVNx loop divider has to be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range must be respected.

The PLLs operate in integer mode when the value of SH_REG (FRACNx shadow register) is set to '0'. The SH_REG is updated with the FRACNx value when PLLxFRACEN bit goes from '0' to '1'. The Sigma-Delta modulator is designed in order to minimize the jitter impact while allowing very small frequency steps.

The PLLs can be enabled by setting PLLxON to '1'. The bits PLLxRDY indicate that the PLL is ready (i.e. locked).

*Note: Before enabling the PLLs, make sure that the reference frequency (**refx_ck**) provided to the PLL is stable, so the hardware does not allow changing DIVMx when the PLLx is ON and it is also not possible to change PLLSRC when one of the PLL is ON.*

The hardware prevents writing PLL1ON to '0' if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to sections HSE oscillator, HSI oscillator and CSI oscillator).
The following PLL parameters cannot be changed once the PLL is enabled: DIVNx, PLLxRGE, PLLxVCOSEL, DIVPx, DIVQx, DIVRx, DIVPxEN, DIVQxEN and DIVRxEN.

To insure an optimal behavior of the PLL when one of the post-divider (DIVP, DIVQ or DIVR) is not used, the application shall set the enable bit (DIVyEN) as well as the corresponding post-divider bits (DIVP, DIVQ or DIVR) to '0'.

If the above rules are not respected, the PLL output frequency is not guaranteed.

Output frequency computation

When the PLL is configured in integer mode (SH_REG = '0'), the VCO frequency (F_{VCO}) is given by the following expression:

$$F_{VCO} = F_{REF_CK} \times DIVN$$

$$F_{PLL_y_CK} = (F_{VCO} / (DIVy + 1)) \text{ with y = P, Q or R}$$

When the PLL is configured in fractional mode (SH_REG different from '0'), the DIVN divider must be initialized before enabling the PLLs. However, it is possible to change the value of FRACNx on-the-fly without disturbing the PLL output.

This feature can be used either to generate a specific frequency from any crystal value with a good accuracy, or to fine-tune the frequency on-the-fly.

For each PLL, the VCO frequency is given by the following formula:

$$F_{VCO} = F_{ref_ck} \times \left(DIVN + \frac{FRACN}{2^{(13)}} \right)$$

Note: For PLL1, DIVP can only take odd values.

The PLLs are disabled by hardware when:

- The system enters Stop or Standby mode.
- An HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock.

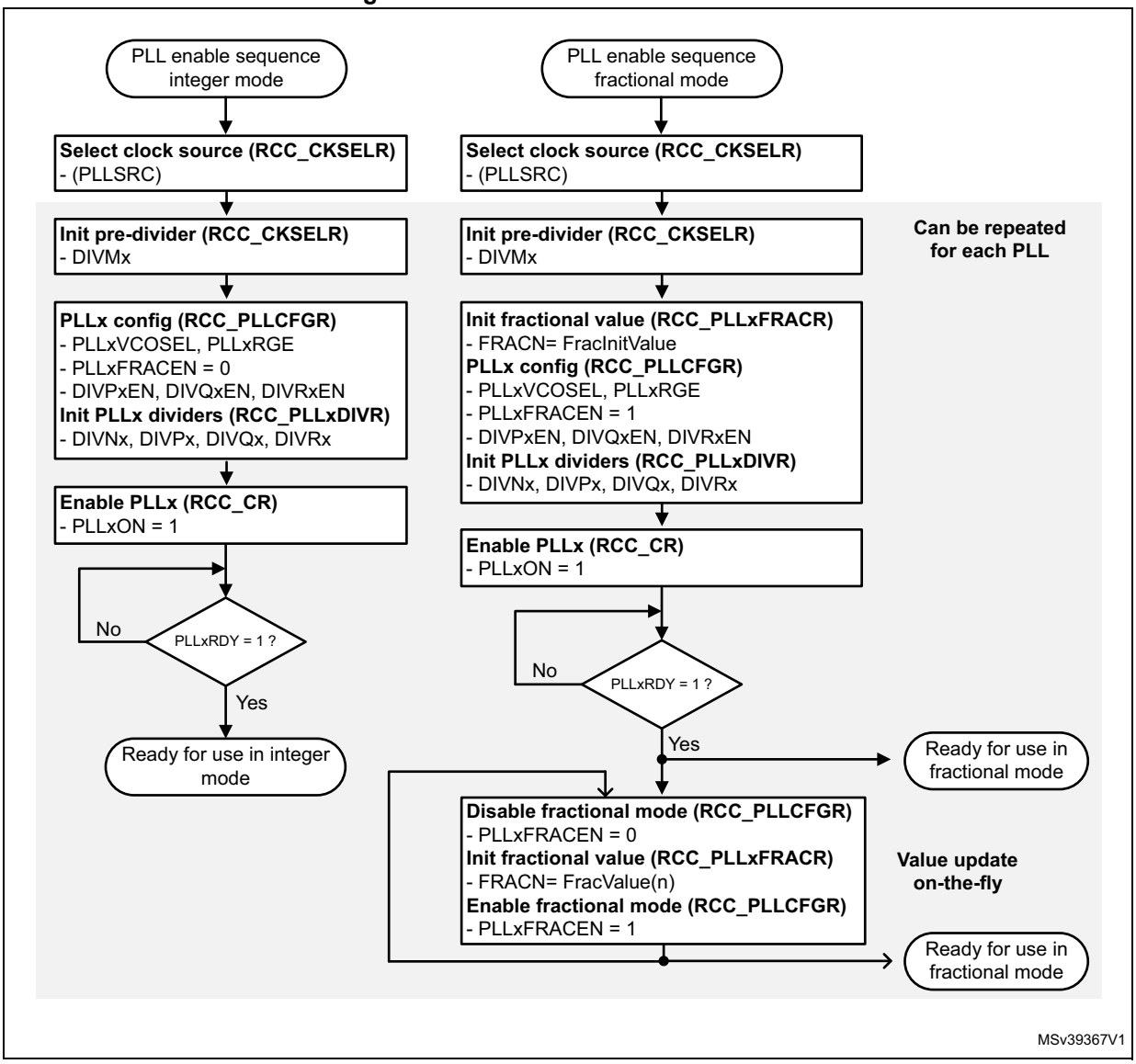
PLL initialization phase

Figure 44 shows the recommended PLL initialization sequence in integer and fractional mode. The PLLx are supposed to be disabled at the start of the initialization sequence:

1. Initialize the PLLs registers according to the required frequency.
   - Set PLLxFRACEN of *RCC PLLs Configuration Register (RCC_PLLCFGR)* to '0' for integer mode.
   - For fractional mode, set FRACN to the required initial value (FracInitValue) and then set PLLxFRACEN to '1'.
2. Once the PLLxON bit is set to '1', the user application has to wait until PLLxRDY bit is set to '1'. If the PLLx is in fractional mode, the PLLxFRACEN bit must not be set back to '0' as long as PLLxRDY = '0'.
3. Once the PLLxRDY bit is set to '1', the PLLx is ready to be used.
4. If the application intends to tune the PLLx frequency on-the-fly (possible only in fractional mode), then:
   a) PLLxFRACEN must be set to '0',
   When PLLxFRACEN = '0', the Sigma-Delta modulator is still operating with the value latched into SH_REG.
   b) A new value must be uploaded into PLLxFRACR (FracValue(n)).
   c) PLLxFRACEN must be set to '1', in order to latch the content of PLLxFRACR into its shadow register.

Note: When the PLLxRDY goes to '1', it means that the difference between the PLLx output frequency and the target value is lower than ±2%.

Figure 44. PLLs Initialization Flowchart



7.5.6 System clock (sys_ck)

System clock selection

After a system reset, the HSI is selected as system clock and all PLLs are switched OFF. When a clock source is used for the system clock, it is not possible for the software to disable the selected source via the xxxON bits.

Of course, the system clock can be stopped by the hardware when the System enters Stop or Standby mode.

When the system is running, the user application can select the system clock (**sys_ck**) among the 4 following sources:

- HSE
- HSI
- CSI
- or pll1_p_ck

This function is controlled by programming the *RCC Clock Configuration Register (RCC_CFGR)*. A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready.
The SWS status bits in the *RCC Clock Configuration Register (RCC_CFGR)* indicate which clock is currently used as system clock. The other status bits in the RCC_CR register indicate which clock(s) is (are) ready.

System clock generation

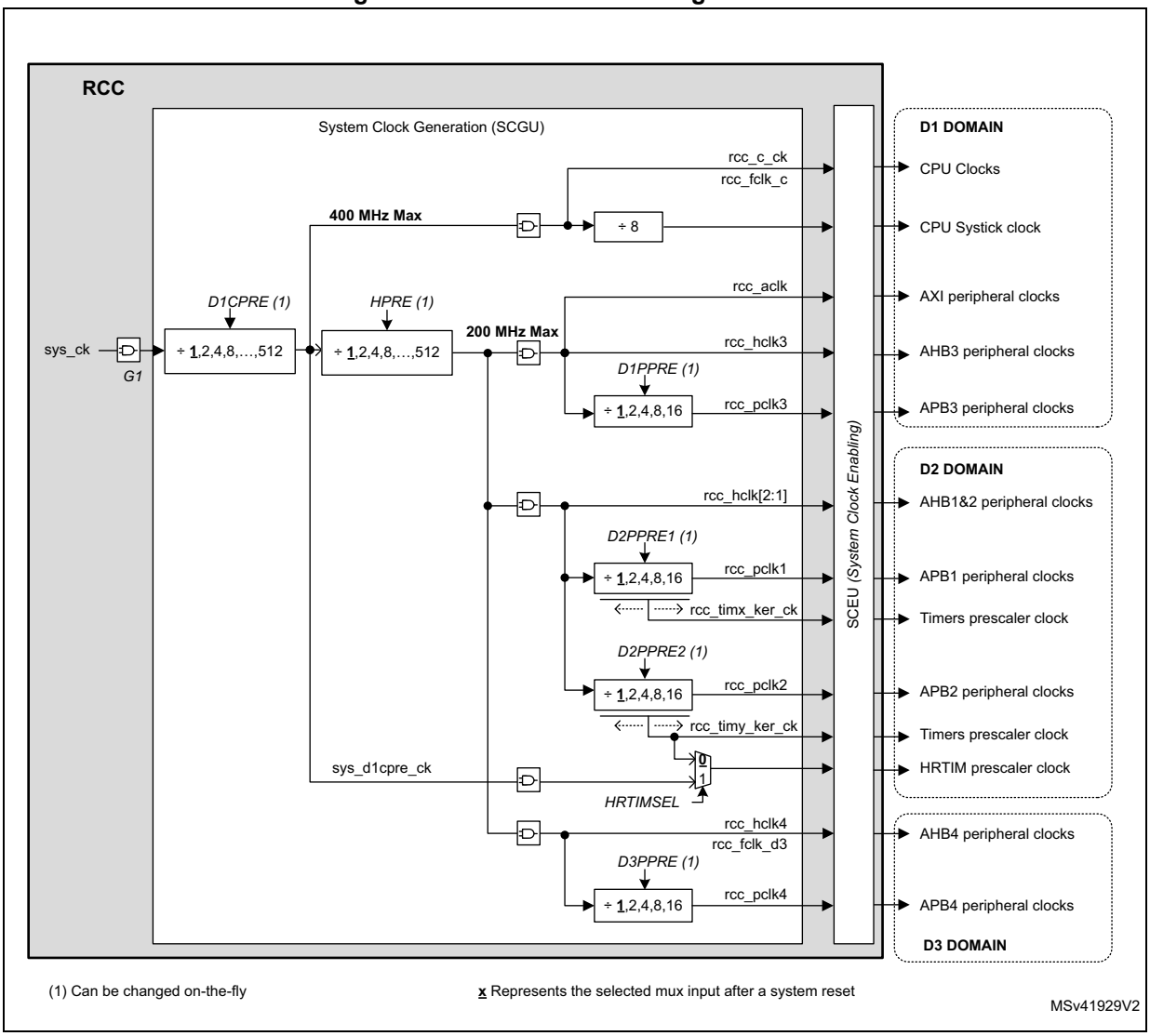
Figure 45 shows a simplified view of the clock distribution for the CPU and busses. All the dividers shown in the block diagram can be changed on-the-fly without generating timing violations. This feature is a very simply solution to adapt the busses frequencies to the application needs, thus optimizing the power consumption.

The D1CPRE divider can be used to adjust the CPU clock. However this also impacts the clock frequency of all bus matrix and HRTIM.

In the same way, HPRE divider can be used to adjust the clock for D1 domain bus matrix, but this also impacts the clock frequency of bus matrix of D2 and D3 domains.

Most of the prescalers are controlled via RCC_D1CFGR, RCC_D2CFGR and RCC_D3CFGR registers.

Figure 45. Core and bus clock generation



This block also provides the clock for the timers (**rcc_timx_ker_ck and rcc_timy_ker_ck**). The frequency of the timers clock depends on the APB prescaler corresponding to the bus to which the timer is connected, and on TIMPRE bit. *Table 52* shows how to select the timer clock frequency.

Table 52. Ratio between clock timer and pclk

D2PPRE1 (1) D2PPRE2	TIMPRE (2)		$F_{rcc_timx_ker_ck}$ $F_{rcc_timy_ker_ck}$	F_{rcc_pclk1} F_{rcc_pclk2}	Comments
0xx	0	→	F_{rcc_hclk1}	F_{rcc_hclk1}	The timer clock is equal to the bus clock.
100	0	→	F_{rcc_hclk1}	F_{rcc_hclk1} / 2	The timer clock is twice as fast as the bus clock.
101	0	→	F_{rcc_hclk1} / 2	F_{rcc_hclk1} / 4	
110	0	→	F_{rcc_hclk1} / 4	F_{rcc_hclk1} / 8	
111	0	→	F_{rcc_hclk1} / 8	F_{rcc_hclk1} / 16	

Table 52. Ratio between clock timer and pclk (continued)

D2PPRE1 (1) D2PPRE2	TIMPRE (2)		$F_{rcc_timx_ker_ck}$ $F_{rcc_timy_ker_ck}$	F_{rcc_pclk1} F_{rcc_pclk2}	Comments
0xx	1	→	F_{rcc_hclk1}	F_{rcc_hclk1}	The timer clock is equal to the bus clock.
100	1	→	F_{rcc_hclk1}	F_{rcc_hclk1} / 2	The timer clock is twice as fast as the bus clock.
101	1	→	F_{rcc_hclk1}	F_{rcc_hclk1} / 4	The timer clock is 4 times faster than the bus clock.
110	1	→	F_{rcc_hclk1} / 2	F_{rcc_hclk1} / 8	
111	1	→	F_{rcc_hclk1} / 4	F_{rcc_hclk1} / 16	

1. D2PPRE1 and D2PPRE2 belong to *RCC Domain 2 Clock Configuration Register (RCC_D2CFGR)*.
2. TIMPRE belongs to *RCC Clock Configuration Register (RCC_CFGR)*.

7.5.7 Handling clock generators in Stop and Standby mode

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped as well as the following clock sources:

- CSI, HSI (depending on HSIKERON, and CSIKERON bits)
- HSE
- PLL1, PLL2 and PLL3
- HSI48

The content of the RCC registers is not altered except for PLL1ON, PLL2ON, PLL3ON HSEON and HSI48ON which are set to ‘0’.

Exiting Stop mode

When the microcontroller exits system Stop mode via a wake-up event, the application can select which oscillator (HSI and/or CSI) will be used to restart. The STOPWUCK bit selects the oscillator used as system clock. The STOPKERWUCK bit selects the oscillator used as kernel clock for peripherals. The STOPKERWUCK bit is useful if after a system Stop a peripheral needs a kernel clock generated by an oscillator different from the one used for the system clock.

All these bits belong to the *RCC Clock Configuration Register (RCC_CFGR)*. *Table 53* gives a detailed description of their behavior.

Table 53. STOPWUCK and STOPKERWUCK description

STOPWUCK	STOPKERWUCK		Activated oscillator when the system exits Stop mode	Distributed clocks when System exits Stop mode	
				System Clock	Kernel Clock
0	0	→	HSI	HSI	HSI
	1	→	HSI and CSI		HSI and/or CSI
1	0	→		CSI	
	1	→	CSI		CSI

During Stop mode

There are two specific cases where the HSI or CSI can be enabled during system Stop mode:

- When a dedicated peripheral requests the kernel clock:

  In this case the peripheral will receive the HSI or CSI according to the kernel clock source selected for this peripheral (via PERxSRC).
- When the bits HSIKERON or CSIKERON are set:

  In this case the HSI and CSI are kept running during Stop mode but the outputs are gated. In that way, the clock will be available immediately when the system exits Stop mode or when a peripheral requests the kernel clock (see *Table 54* for details).

HSIKERON and CSIKERON bits belong to *RCC Source Control Register (RCC_CR)*. *Table 54* gives a detailed description of their behavior.

Table 54. HSIKERON and CSIKERON behavior

HSIKERON (CSIKERON)		HSI (CSI) state during Stop mode	HSI (CSI) Setting time
0	→	OFF	$t_{su(HSI)}$ ($t_{su(CSI)}$) (1)
1	→	Running and Gated	Immediate

1. $t_{su(HSI)}$ and $t_{su(CSI)}$ are the startup times of the HSI and CSI oscillators (see refer to the product datasheet for the values of these parameters).

When the microcontroller exists system Standby mode, the HSI is selected as system and kernel clock, the RCC registers are reset to their initial values except for the RCC_RSR (or RCC_C1_RSR) and RCC_BDCR registers.

Note as well that the HSI and CSI outputs provide two clock paths (see *Figure 41*):

- one path for the system clock (**hsi_ck** or **csi_ck**)
- one path for the peripheral kernel clock (**hsi_ker_ck** or **csi_ker_ck**).

When a peripheral requests the kernel clock in system Stop mode, only the path providing the **hsi_ker_ck** or **csi_ker_ck** is activated.

Caution: It is not guaranteed that the CPU will get automatically the same clock frequencies when leaving CStop mode: this mainly depends on the System state. For example If the CPU goes to CStop, while the D3 domain is kept in CRun, when the CPU exits from CStop, the clock settings remain unchanged. If the D3 domain goes to CStop while the CPU is also in CStop, then when the CPU exits from CStop, the CPU will operate with HSI or CSI when it left the CStop mode.

7.5.8 Kernel clock selection

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- a clock domain synchronous with the register and bus interface (**ckg_bus_perx** clock)
- and a clock domain generally synchronous with the peripheral (kernel clock).

The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral.

As a consequence, the user application can change the bus frequency without reprogramming the peripherals. As an example an on-going transfer with UART will not be disturbed if its APB clock is changed on-the-fly.

Table 55 shows the kernel clock that the RCC can deliver to the peripherals. Each row of *Table 55* represents a MUX and the peripherals connected to its output. The columns starting from number 4 represents the clock sources. Column 3 gives the maximum allowed frequency at each MUX output. It is up to the user to respect these requirements.

Table 55. Kernel clock distribution overview

Peripherals	Clock mux control bits	Max. allowed freq. [MHz]	Domain	Clock Sources																		
				pll1_q_ck	pll2_p_ck	pll2_q_ck	pll2_r_ck	pll3_p_ck	pll3_q_ck	pll3_r_ck	sys_ck	bus clocks (1)	hse_ck	hsi_ker_ck	csi_ker_ck	hsi48_ck	lse_ck	lsi_ck	per_ck (2)	I2S_CKIN	USB_PHY1	Disabled
LTDC	-	66	D1							x												
FMC	FMCSEL	200		1			2					0							3			
QUADSPI	QSPISEL	200		1			2					0							3			
SDMMC1	SDMMCSEL	200 (4)		0			1															
SDMMC2		200	D2																			
DFSDM1 Aclk	SAI1SEL	133		0	1			2											4	3		
DFSDM1 clk	DFSDM1SEL	133									1	0										
FDCAN	FDCANSEL	100		1		2							0									
HDMI-CEC	CECSEL	66													2 (3)		0	1				
I2C1,2,3	I2C123SEL	100								1		0		2	3							
LPTIM1	LPTIM1SEL	100			1					2		0					3	4	5			
TIM[8:1], TIM[17:12]	-	200										x										
HRTIM	-	400										x										
RNG	RNGSEL	66		1												0	2	3				
SAI1	SAI1SEL	133		0	1			2											4	3		
SAI2	SAI23SEL	133		0	1			2											4	3		
SAI3		133																				
SPDIFRX	SPDIFSEL	200		0			1			2				3								
SPI(I2S)1,2,3	SPI123SEL	200		0	1			2											4	3		
SPI4,5	SPI45SEL	100				1			2			0	5	3	4							
SWPMI	SWPSEL	100										0		1								
USART1,6	USART16SEL	100				1			2			0		3	4		5					
USART2,3 UART4,5,7,8	USART234578 SEL	100				1			2			0		3	4		5					
USB1OTG USB2OTG	USBSEL	66		1					2							3						0
USB1ULPI	-	100																			x	
ADC1,2	ADCSEL	36 (4)			0					1									2			
ADC3			D3																			

Table 55. Kernel clock distribution overview (continued)

Peripherals	Clock mux control bits	Max. allowed freq. [MHz]	Domain	Clock Sources																		
				pll1_q_ck	pll2_p_ck	pll2_q_ck	pll2_r_ck	pll3_p_ck	pll3_q_ck	pll3_r_ck	sys_ck	bus clocks (1)	hse_ck	hsi_ker_ck	csi_ker_ck	hsi48_ck	lse_ck	lsi_ck	per_ck (2)	I2S_CKIN	USB_PHY1	Disabled
I2C4	I2C4SEL	100	D3							1		0		2	3							
LPTIM2	LPTIM2SEL	100			1					2		0					3	4	5			
LPTIM3,4,5	LPTIM345SEL	100			1					2		0					3	4	5			
LPUART1	LPUART1SEL	100				1			2			0		3	4		5					
SAI4_A	SAI4ASEL	133		0	1			2											4	3		
SAI4_B	SAI4BSEL	133		0	1			2											4	3		
SPI6	SPI6SEL	100				1			2			0	5	3	4							
RTC/AWU	RTCSEL	1	VSW										3 (5)				1	2				0

1. The bus clocks are the bus interface clocks to which the peripherals are connected, it can be APB, AHB or AXI clocks.
2. The per_ck clock could be hse_ck, hsi_ker_ck or csi_ker_ck according to CKPERSEL selection.
3. Clock CSI divided by 122.
4. With a duty cycle close to 50%, meaning that DIV[P/Q/R]x values shall be even. For SDMMCx, the duty cycle shall be 50% when supporting DDR.
5. Clock HSE divided by RTCPRE.

Figure 46 to *Figure 55* provide a more detailed description of kernel clock distribution. To simplify the drawings, the bus interface clocks (pclk, hclk) are not represented, even if they are gated with enable signals. Refer to *Section 7.5.11: Peripheral clock gating control* for more details.

To reduce the amount of switches, some peripherals share the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

Peripherals dedicated to audio applications

The audio peripherals generally need specific accurate frequencies, except for SPDIFRX. As shown in *Figure 46*, the kernel clock of the SAIs or SPI(I2S)s can be generated by:

- The PLL1 when the amount of active PLLs has to be reduced
- The PLL2 or 3 for optimal flexibility in frequency generation
- HSE, HSI or CSI for use-cases where the current consumption is critical
- I2S_CKIN when an external clock reference need to be used.

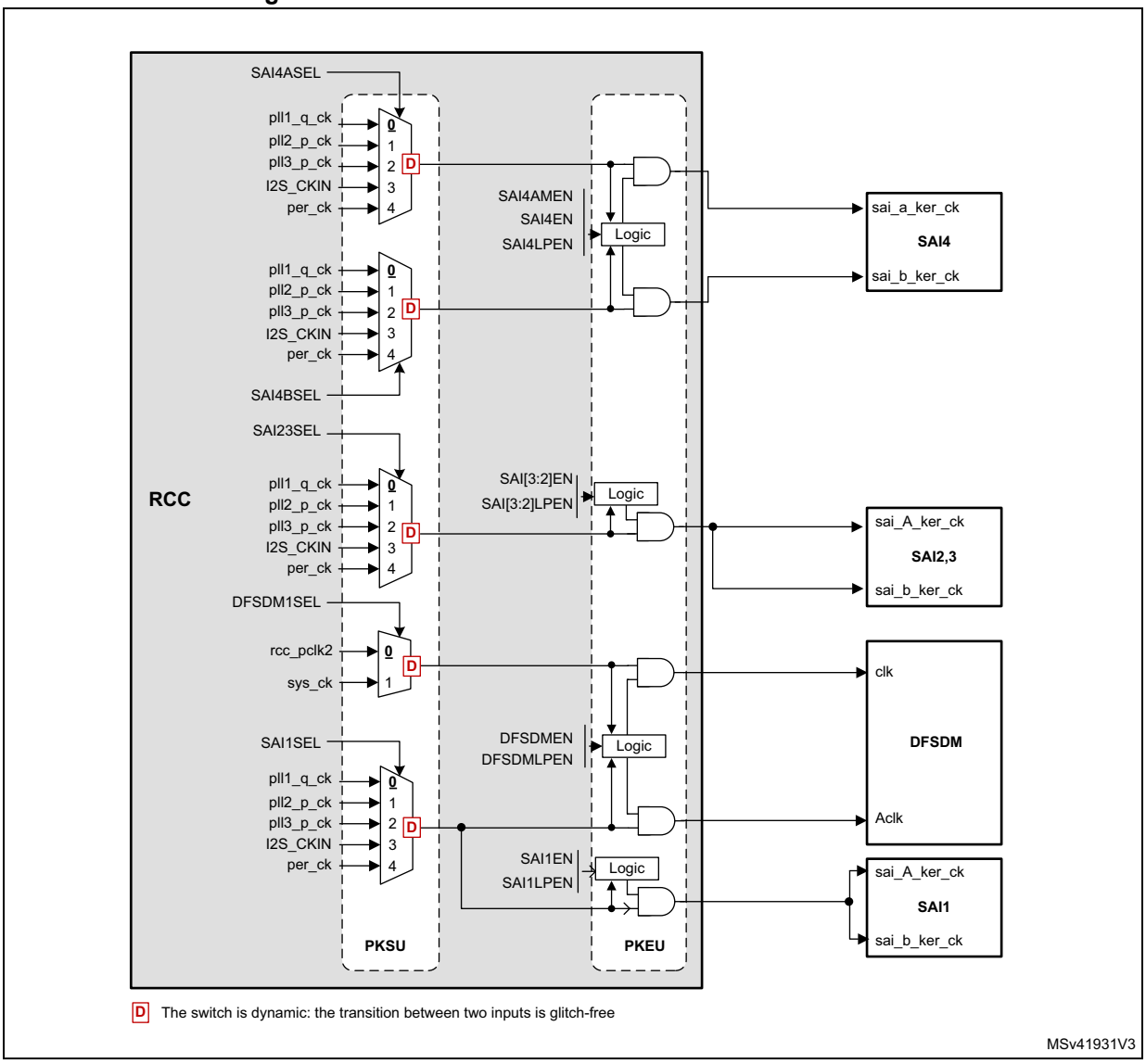
Note: The SPDIFRX does not require a specific frequency, but only a kernel clock frequency high enough to make the peripheral work properly. Refer to the SPDIFRX description for more details.

DFSDM1 can use the same clock as SAI1A. This is useful when DFSDM1 is used for audio applications.

To improve the flexibility, SAI4 can use different clock for each sub-block.

The SPI/I2S1, 2, and 3 share the same kernel clock source (see Figure 47).

Figure 46. Kernel clock distribution for SAIs and DFSDM



1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripherals. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

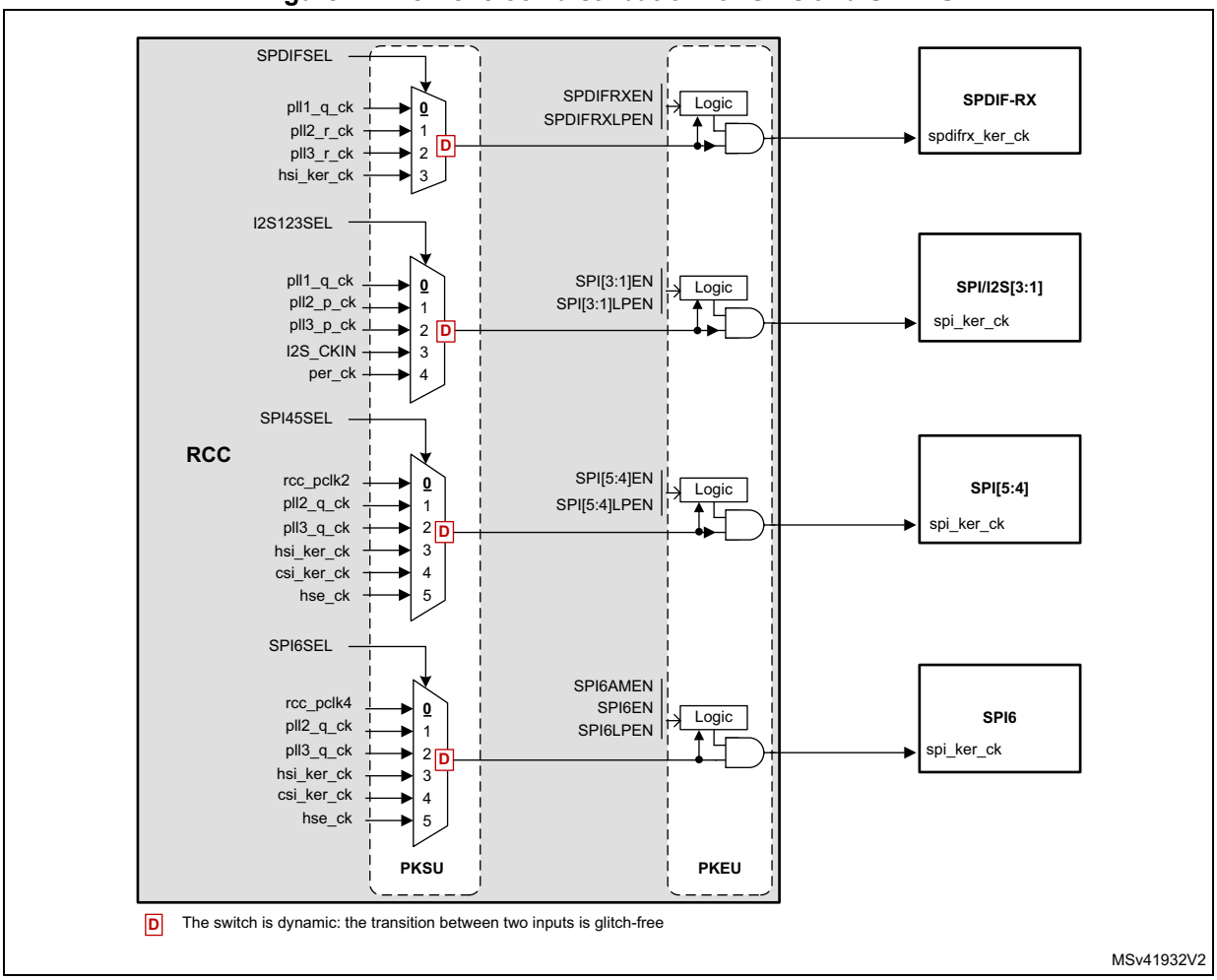
Peripherals dedicated to control and data transfer

Peripherals such as SPIs, I2Cs, UARTs do not need a specific kernel clock frequency but a clock fast enough to generate the correct baud rate, or the required bit clock on the serial interface. For that purpose the source can be selected among:

- PLL1 when the amount of active PLLs has to be reduced
- PLL2 or PLL3 if better flexibility is required. As an example, this solution allows changing the frequency bus via PLL1 without affecting the speed of some serial interfaces.
- HSI or CSI for low-power use-cases or when the peripheral has to quickly wake up from Stop mode (i.e. UART, I2C...).

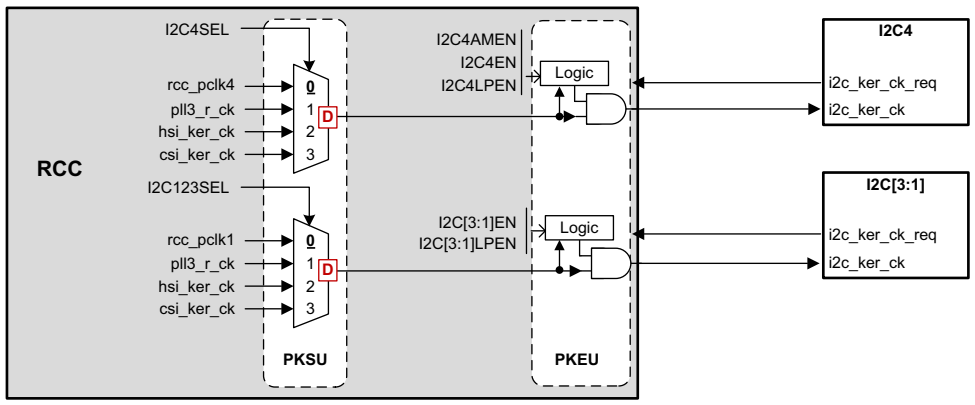
Note: UARTs also need the LSE clock when high baud rates are not required.

Figure 47. Kernel clock distribution for SPIs and SPI/I2S



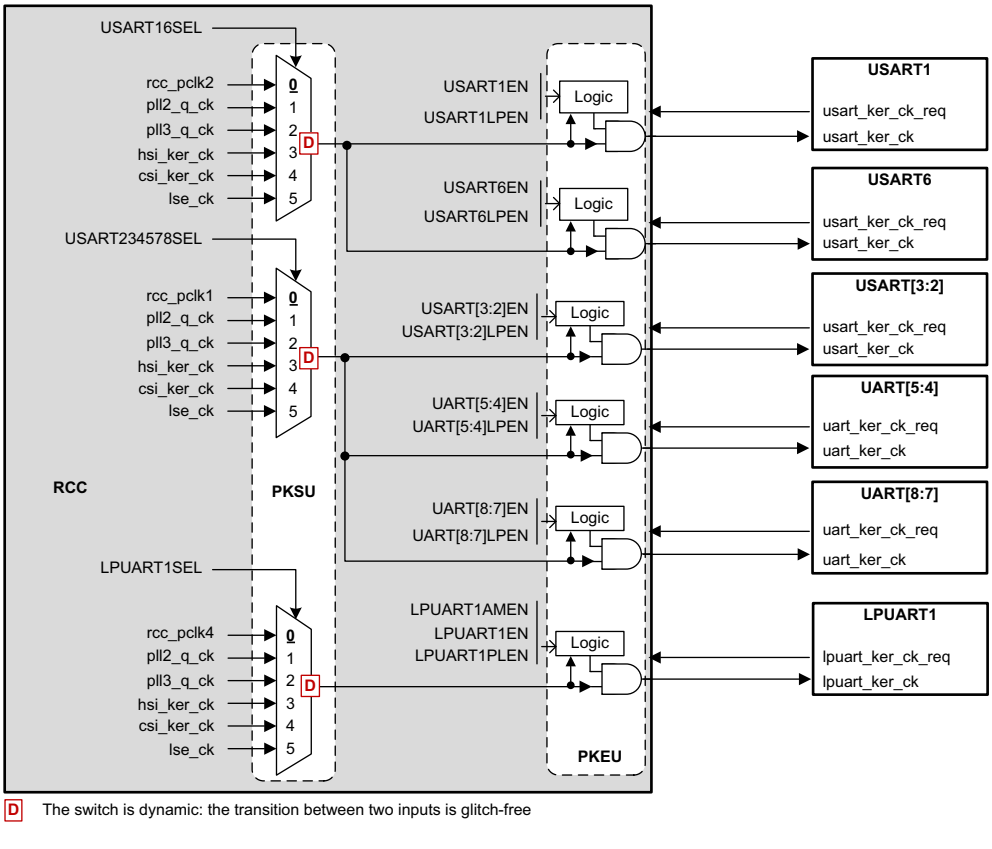
1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

Figure 48. Kernel clock distribution for I2Cs



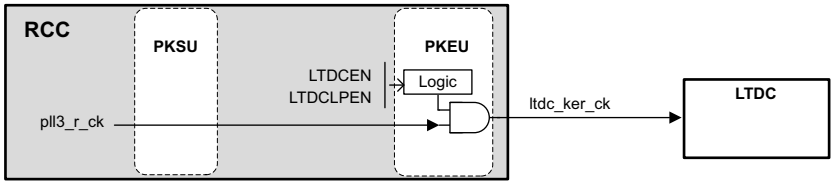
1. **X** represents the selected MUX input after a system reset
2. This figure does not show the connection of the bus interface clock to the peripheral, for details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

Figure 49. Kernel clock distribution for UARTs, USARTs and LPUART1



1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral, for details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

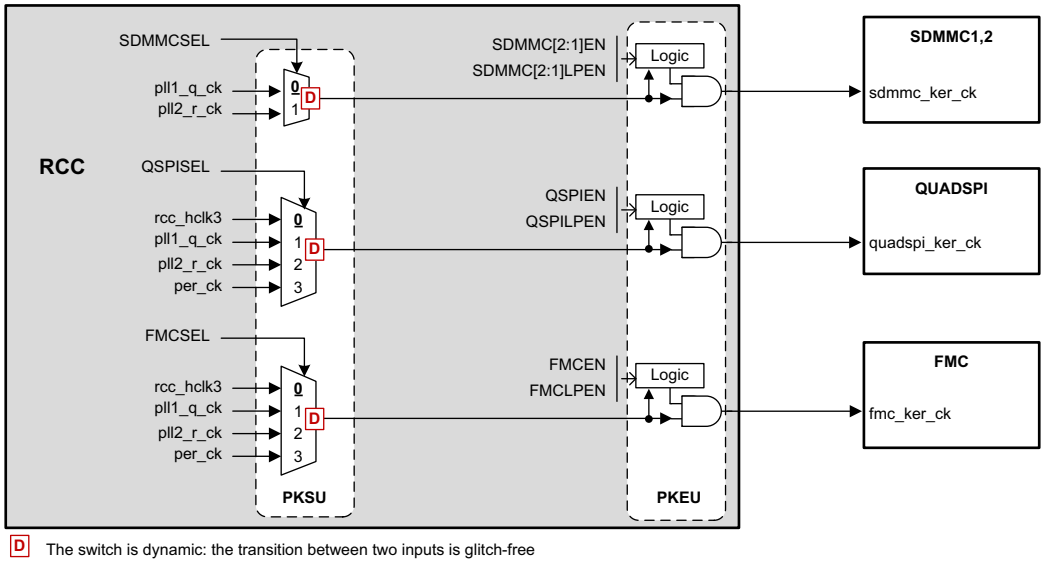
Figure 50. Kernel clock distribution for LTDC



1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

The FMC, QUADSPI and SDMMC1/2 can also use a clock different from the bus interface clock for more flexibility.

Figure 51. Kernel clock distribution for SDMMC, QUADSPI and FMC

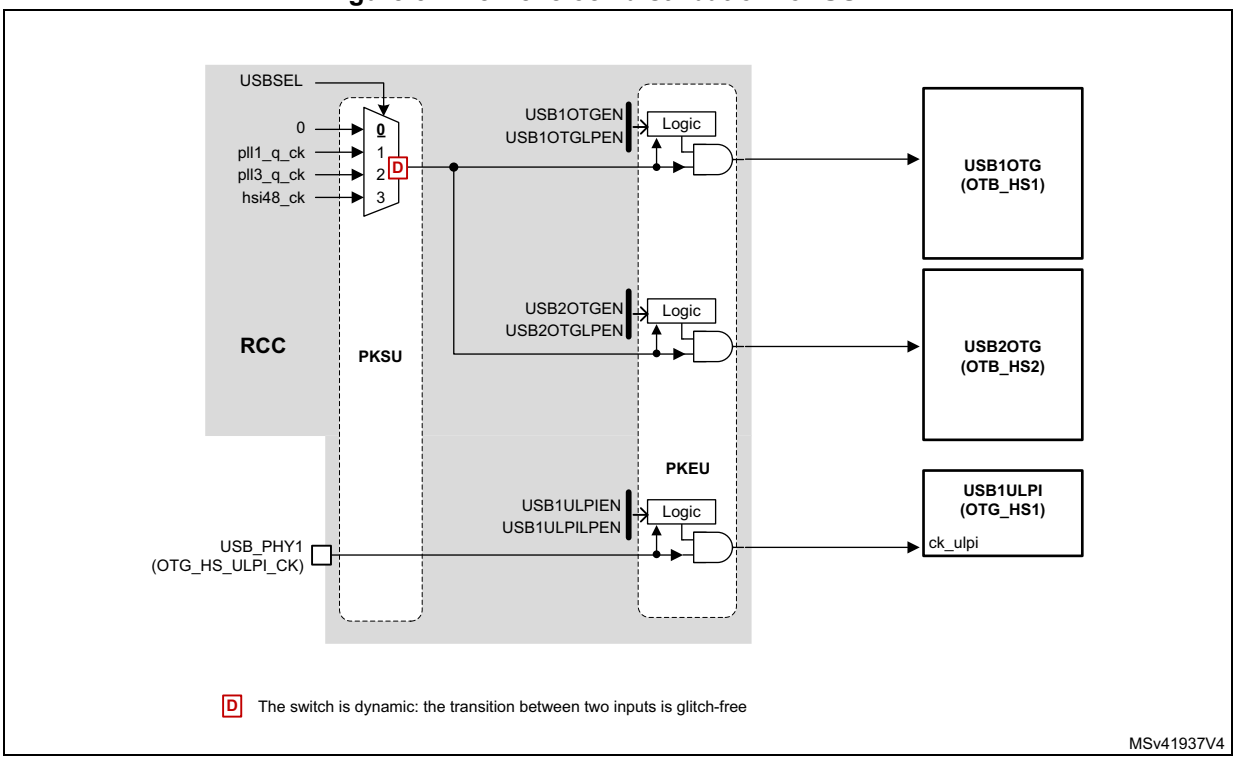


1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

Figure 52 shows the clock distribution for the USB blocks. The USB1ULPI block receives its clock from the external PHY.

The USBxOTG blocks receive the clock for USB communications which can be selected among different sources thanks to the MUX controlled by USBSEL.

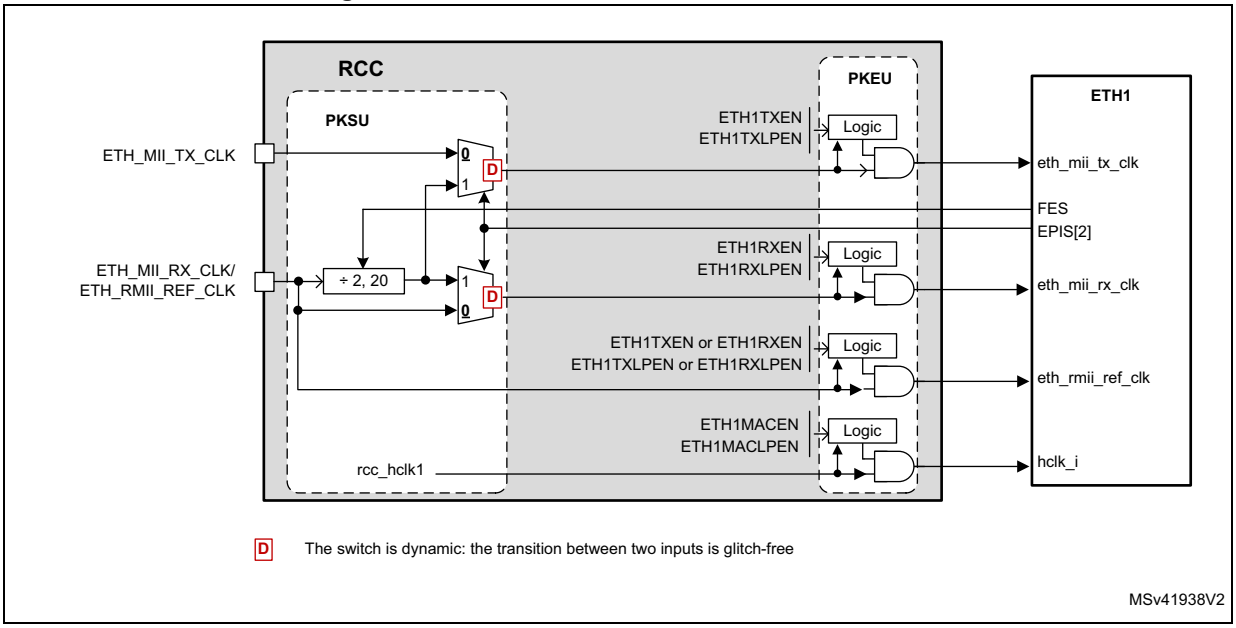
Figure 52. Kernel clock distribution for USB (2)



1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

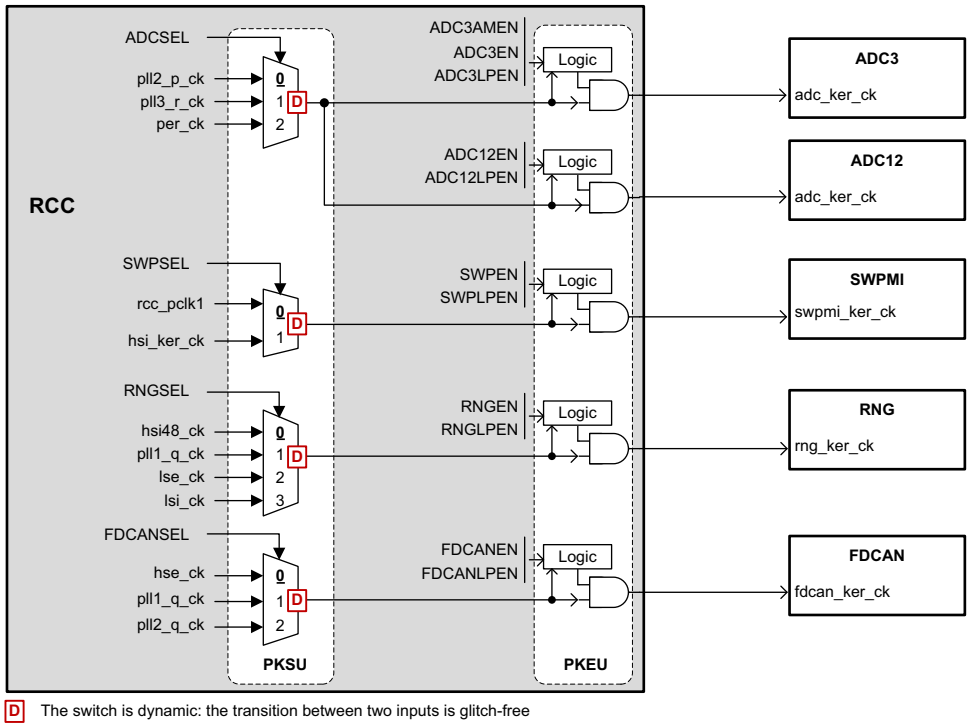
The Ethernet transmit and receive clocks shall be provided from an external Ethernet PHY. The clock selection for the RX and TX path is controlled via the SYSCFG block.

Figure 53. Kernel clock distribution for Ethernet



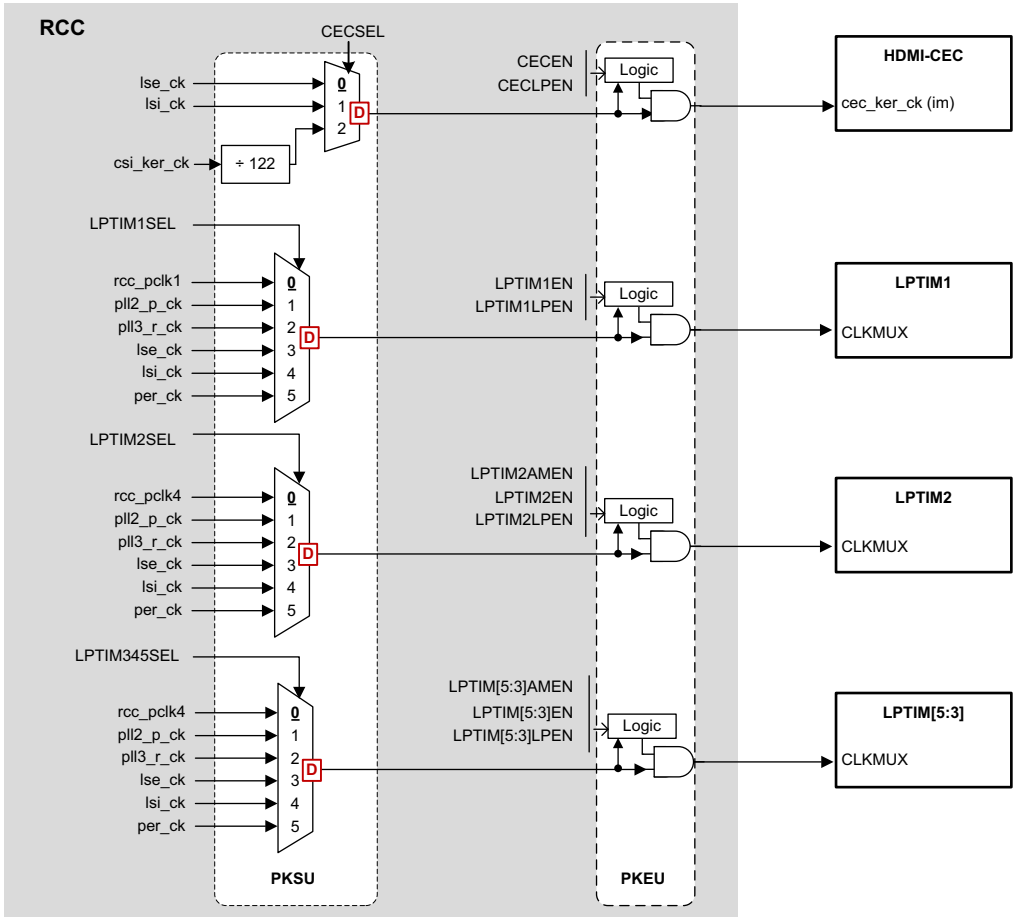
1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

Figure 54. Kernel clock distribution For ADCs, SWPMI, RNG and FDCAN (2)



1. **X** represents the selected MUX input after a system reset.
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

Figure 55. Kernel clock distribution for LPTIMs and HDMI-CEC (2)



1. **X** represents the selected MUX input after a system reset
2. This figure does not show the connection of the bus interface clock to the peripheral. For details on each enable cell, please refer to *Section 7.5.11: Peripheral clock gating control*.

RTC/AWU clock

The **rtc_ck** clock source can be:

- the **hse_1M_ck** (**hse_ck** divided by a programmable prescaler)
- the **lse_ck**
- or the **lsi_ck** clock

The source clock is selected by programming the RTCSEL[1:0] bits in the *RCC Backup Domain Control Register (RCC_BDCR)* and the RTCPRE[5:0] bits in the *RCC Clock Configuration Register (RCC_CFGR)*.

This selection cannot be modified without resetting the Backup domain.

If the LSE is selected as RTC clock, the RTC will work normally even if the backup or the V_{DD} supply disappears.

The LSE clock is in the Backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as RTC clock, the RTC continues working even if the V_{DD} supply is switched OFF, provided the V_{BAT} supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the V_{CORE} supply is powered off.

The **rtc_ck** clock is enabled through RTCEN bit located in the *RCC Backup Domain Control Register (RCC_BDCR)*.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLPEN bits located in RCC_APB4ENR/LPENR registers.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency ($F_{APB} < 7 \times F_{RTCLCK}$), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result than the first one. Otherwise a third read access must be performed.

Watchdog clocks

The RCC provides the clock for the four watchdog blocks available on the circuit. The independent watchdog (IWDG1) is connected to the LSI. The window watchdog (WWDG1) are connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced ON and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDGs.

Caution: Before enabling the WWDG1, the application must set the WW1RSC bit to ‘1’. If the WW1RSC remains to ‘0’, when the WWDG1 is enabled, its the behavior is not guaranteed. The WW1RSC bit is located in *RCC Global Control Register (RCC_GCR)*.

Clock frequency measurement using TIMx

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

- Calibrating the HSI or CSI with the LSE

  The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI or CSI. This requires to use the HSI or CSI as system clock source either directly or via PLL1. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

  The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio): the precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

  The HSI and CSI oscillators have dedicated user-accessible calibration bits for this purpose (see *RCC Internal Clock Source Calibration Register (RCC_ICSCR)*). When

the HSI or CSI are used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- Calibrating the LSI with the HSI

  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultralow power LSI oscillator has a large manufacturing process deviation. By measuring it versus the HSI clock source, it is possible to determine its frequency with the precision of the HSI. The measured value can be used to have more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy.

7.5.9 General clock concept overview

The RCC handles the distribution of the CPU, bus interface and peripheral clocks for the system (D1, D2 and D3 domains), according to the operating mode of each function (refer to *Section 7.5.1: Clock naming convention* for details on clock definitions).

For each peripheral, the application can control the activation/deactivation of its kernel and bus interface clock. Prior to use a peripheral, the CPU has to enable it (by setting PERxEN to '1'), and define if this peripheral remains active in CSleep mode (by setting PERxLPEN to '1'). This is called 'allocation' of a peripheral to the CPU (refer to *Section 7.5.10: Peripheral allocation* for more details).

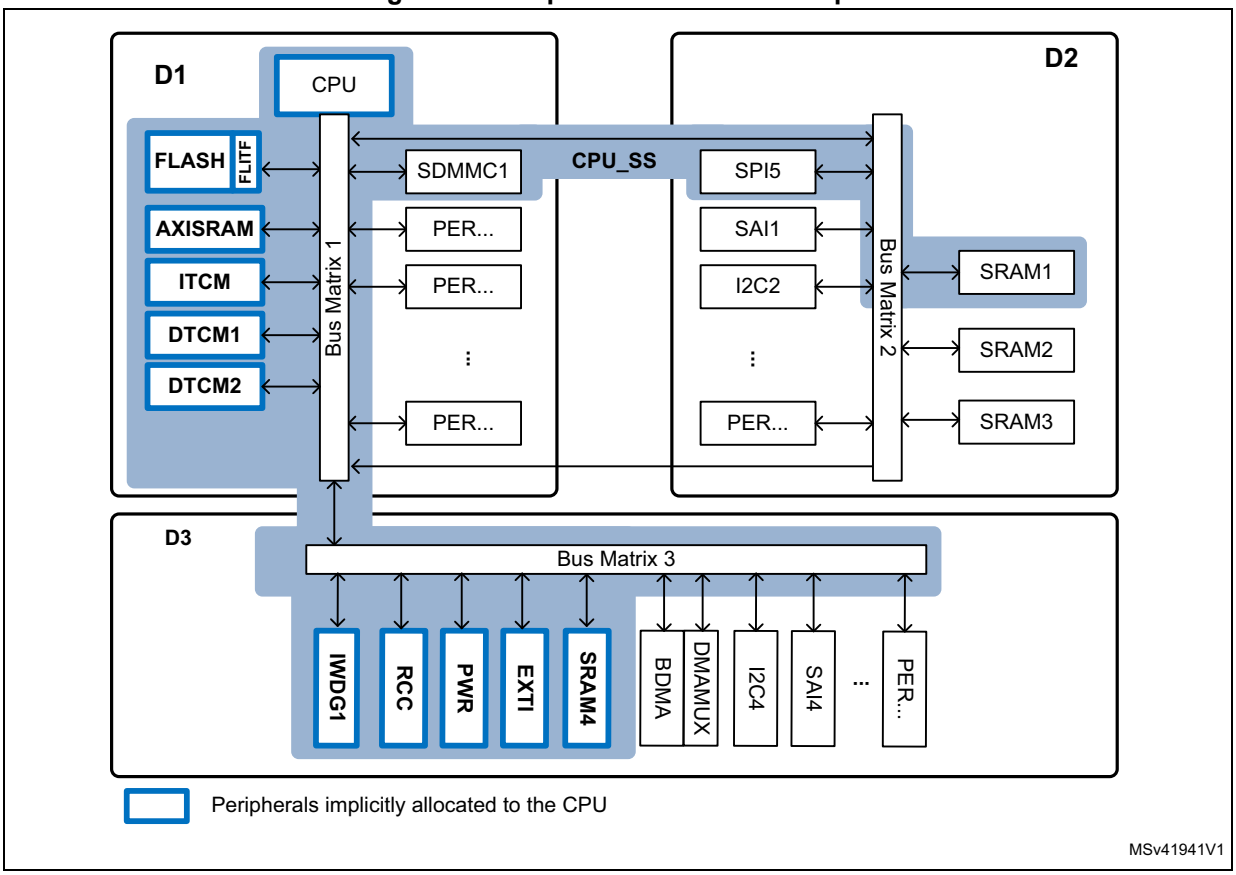
The peripheral allocation is used by the RCC to automatically control the clock gating according to the CPU and domain modes, and by the PWR to control the supply voltages of D1, D2 and D3 domains.

Figure 56 gives an example of peripheral allocation:

- The CPU enabled SDMMC1, SPI5 and SRAM1, AXISRAM, ITCM, DTCM1, DTCM2 and SRAM4 are implicitly allocated to the CPU. The group composed of the CPU, bus matrix 1/2/3 and allocated peripherals makes up a sub-system (CPU_SS).

Note: The FLASH, AXISRAM, ITCM, DTCM1, DTCM2, SRAM4, IWGD1, IWGD2, PWR, EXTI and RCC are common resources and are implicitly allocated to the CPU.

Figure 56. Peripheral allocation example



When the CPU enters CStop mode, the RCC automatically disables the bus interface and kernel clocks of all the peripherals of the CPU_SS as well as the CPU clock. The PLLs, if enabled, are not disabled by the RCC since D3 is still running.

The D3 domain can be kept in DRun mode while the CPU is in CStop mode and D1 and D2 domains are in DStop or DStandby mode. This is done by setting RUN_D3 bit in PWR_CPUCR registers.

- If RUN_D3 is set to ‘1’, then D3 is maintained in DRun mode, independently from the CPU modes (see *PWR CPU control register (PWR_CPUCR)*.
- If RUN_D3 is set to ‘0’, then D3 domain enters DStop or DStandby mode when the CPU enters CStop mode (see *Table 56*).

Note that the CPU can control if D1, D2 or D3 domains are allowed to enter in DStandby when conditions are met, via bits PDDS_D1, PDDS_D2 and PDDS_D3 of *PWR CPU control register (PWR_CPUCR)*.

A wakeup event will be able to exit D1, D2 and D3 domains from DStandby or DStop mode.

In addition, more autonomy can be given to some peripherals located into D3 domain (refer to *Section : D3 domain Autonomous mode* for details).

D3 domain Autonomous mode

The Autonomous mode allows to deliver the peripheral clocks to peripherals located in D3, even if the CPU is in CStop mode. When a peripheral has its autonomous bit enabled, it receives its peripheral clocks according to D3 domain state, if the CPU is in CStop mode:

- If the D3 domain is in DRun mode, peripherals with Autonomous mode activated receive their peripheral clocks,
- If the D3 domain is in DStop mode, no peripheral clock is provided.

The Autonomous mode does not prevent the D3 domain to enter DStop or DStandby mode.

The autonomous bits are located in *RCC D3 Autonomous mode Register (RCC_D3AMR)*.

For example, the CPU can enter CStop mode, while the SAI4 is filling the SRAM4 with data received from an external device via BDMA. When the amount of received data is reached, the CPU can be activated by a wakeup event. This can be done by setting the SAI4, the BDMA, and SRAM4 in Autonomous mode, while keeping D3 in DRun mode (RUN_D3 set to '1'). In this example, the RCC does not switch off the PLLs as the D3 domain is always in DRun mode.

It is possible to go a step further with power consumption reduction by combining the Autonomous mode with the capability of some peripherals (UARTs, I2Cs) to request the kernel clock on their own, without waking-up the CPU. For example, if the system is expecting messages via I2C4, the whole system can be put in Stop mode. When the I2C4 peripheral detects a START bit, it will generate a "kernel clock request". This request enables the HSI or CSI, and a kernel clock is provided only to the requester (in our example the I2C4). The I2C4 then decodes the incoming message. Several cases are then possible:

- If the device address of the message does not match, then I2C4 releases its "kernel clock request" until a new START condition is detected.
- If the device address of the incoming message matches, it has to be stored into D3 local memory. I2C4 is able to generate a wakeup event on address match to switch the D3 domain to DRun mode. The message is then transferred into memory via BDMA, and the D3 domain go back to DStop mode without any CPU activation. Note that if the amount of data transferred into memory reached the transfer count, the BDMA can also generate an interrupt to wake-up the CPU.
- If the device address of the incoming message matches and the peripheral is setup to wake up the CPU, then I2C4 generates a wakeup event to activate the CPU.

Please refer to the description of EXTI block in order see which peripheral is able to perform a wake-up event to which domain.

Memory handling

The CPU can access all the memory areas available in the product:

- AXISRAM, ITCM, DTCM1, DTCM2 and FLASH,
- SRAM1, SRAM2 and SRAM3,
- SRAM4 and BKPRAM.

As shown in *Figure 56*, FLASH, AXISRAM, SRAM4, ITCM, DTCM1 and DTCM2 are implicitly allocated to the CPU. As a result, there is no enable bit allowing the CPU to allocate these memories.

If the CPU wants to use memories located into D2 domain (SRAM1, SRAM2 and SRAM3), it has to enable them.

The BKPRAM has a dedicated enable in order to gate the bus interface clock. The CPU needs to enable the BKPRAM prior to use it.

Note: The memory interface clocks (Flash and RAM interfaces) can be stopped by software during CSleep mode (via DxSRAMyLPEN bits).

Refer to *Peripheral clock gating control* and *CPU and bus matrix clock gating control* sections for details on clock enabling.

System states overview

Table 56 gives an overview of the system states with respect to the D1, D2 and D3 domain modes.

- The system remains in Run mode as long as D3 is in DRun mode. Several sub-states of system Run exist that are not detailed here (refer *Power control (PWR)* for more information).
- When the D1 domain is in DRun, the D2 domain can be in DRun, DStop or DStandby. When the D1 domain is in DStop or DStandby, the D2 domain can no longer remain in DRun it will switch to DStop or DStandby according to PDDS_D2 bit.
- D3 can run while D1 and D2 are in DStop/DStandby mode thanks to RUN_D3 bits of PWR_CPUCR registers or when D3 is in Autonomous mode.
- The system remains in Stop mode as long as D3 is in DStop mode. This means implicitly that D1 and D2 are in DStop or DStandby. As soon as D1 or D2 exits DStop or DStandby, D3 switches to DRun mode.
- The system remains in Standby mode as long as D1, D2 and D3 are in DStandby.
- Domain states versus CPU states:
  - When the D1 domain is in DRun mode, it means that its bus matrix is clocked, and the CPU is in CRun mode.
  - When the D2 domain is in DRun mode, it means that its bus matrix is clocked, and the CPU is in CRun mode with at least a peripheral of D2 domain allocated.
  - When the D1 domain is in DStop mode it means that its bus matrix is no longer clocked, and the CPU is in CStop mode.
  - When the D2 domain is in DStop mode it means that its bus matrix is no longer clocked. This situation happens when:

    - the CPU did not allocate peripherals of D2 domain,

    - the CPU allocated peripherals of D2 domain, but the CPU is in CStop or CStandby,
  - When a domain is in DStandby mode, it means that the domain including its CPU are powered down.

Table 56. System states overview

System State	D1 State	D2 State	D3 State
Run	DRun	DRun/DStop/DStandby	DRun
	DStop/DStandby	DStop/DStandby	
Stop	DStop/DStandby	DStop/ DStandby	DStop
Standby	DStandby	DStandby	DStandby

7.5.10 Peripheral allocation

The CPU can allocate a peripheral and hence control its kernel and bus interface clock.

The CPU can allocate a peripheral by setting the dedicated PERxEN bit located into:

- RCC_xxxxENR registers or
- RCC_C1_xxxxENR registers.

The CPU can control the peripheral clocks gating when it is in CSleep mode via the PERxLPEN bits located into:

- RCC_xxxxLPENR registers or
- RCC_C1_xxxxLPENR registers.

Refer to *Section 7.7.1: Register mapping overview* for additional information.

The peripheral allocation bits (PERxEN bits) are used by the hardware to provide the kernel and bus interface clocks to the peripherals. However they are also used to link peripherals to the CPU (CPU sub-system). In this way, the hardware is able to safely gate the peripheral clocks and bus matrix clocks according to CPU states. The PWR block also uses this information to control properly the domain states.

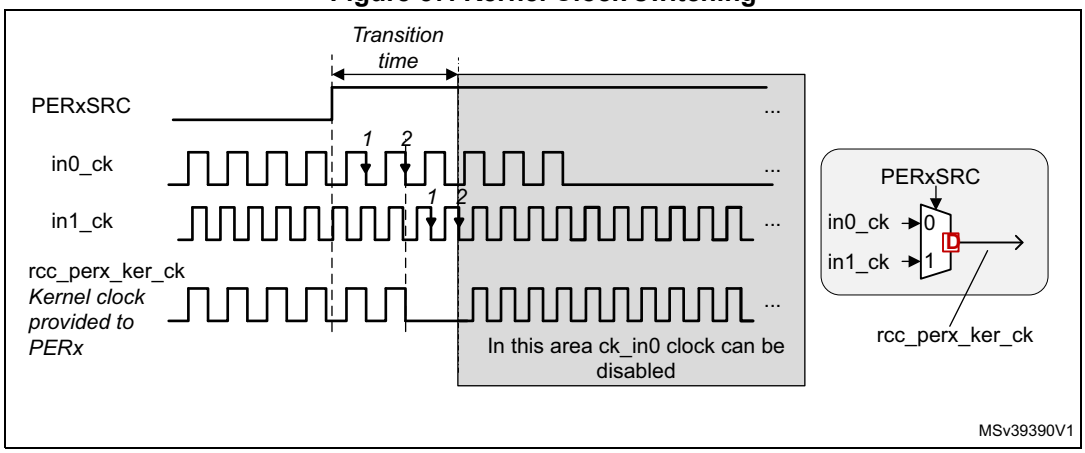
Clock switches and gating

- Clock switching delays

  The input selected by the kernel clock switches can be changed dynamically without generating spurs or timing violation. As a consequence, switching from the original to the new input can only be performed if a clock is present on both inputs. If it not the case, no clock will be provided to the peripheral. To recover from this situation, the user has to provide a valid clock to both inputs.

  During the transition from one input to another, the kernel clock provided to the peripheral will be gated, in the worst case, during 2 clock cycles of the previously selected clock and 2 clock cycles of the new selected clock. As shown in *Figure 57*, both input clocks shall be present during transition time.

Figure 57. Kernel Clock switching



- Clock enabling delays

  In the same way, the clock gating logic synchronizes the enable command (coming generally from a kernel clock request or PERxEN bits) with the selected clock, in order to avoid generation of spurs.

  – A maximum delay of two periods of the enabled clock may occur between the enable command and the first rising edge of the clock. The enable command can be the rising edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request asserted by a peripheral.

  – A maximum delay of 1.5 periods of the disabled clock may occur between the disable command and the last falling edge of the clock. The disable command can be the falling edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request released by a peripheral.

Note: Both the kernel clock and the bus interface clock are affected by this re-synchronization delay.

In addition, the clock enabling delay may strongly increase if the application is enabling for the first time a peripheral which is not located into the same domain. This is due to the fact that the domain where the peripheral is located could be in DStop or DStandby mode. The domain must be switched to DRun mode before the application can use this peripheral.

As an example, if the CPU enables a peripheral located in the D2 domain while the D2 domain is in DStop/DStandby mode, then the power controller (PWR) has first to provide a supply voltage to D2, then the RCC has to wait for an acknowledge from the PWR before enabling the clocks of the D2 domain. To handle properly this situation the RCC and the PWR blocks feature four flags:

– D1CKRDY/D2CKRDY located in *RCC Source Control Register (RCC_CR)*

– SBF_D1 and SBF_D2 located in *PWR CPU control register (PWR_CPUCR)*.

The following sequence can be followed to avoid this issue:

a) Enable the peripheral clocks (i.e. allocate the peripheral) by writing the corresponding PERxEN bit to ‘1’ in the RCC_xxxxENR register,

b) Read back the RCC_xxxxENR register to make sure that the previous write operation is not pending into a write buffer.

c) If the peripheral is located in a different domain, perform the two next steps:

   Read DxCKRDY until it is set to ‘1’.

   Write SBF_Dx to zero and read-back the value, in order to check if the domain where the peripheral is located is still in DStandby. If the corresponding bit is read at ‘1’, it means that the domain is still in DStandby. Repeat this operation until SBF_Dx is equal to ‘0’, then continue the other steps.

d) Perform a dummy read operation into a register of the enabled peripheral. This operation will take at least 2 clock cycles, which is equal to the max delay of the enable command.

e) The peripheral can then be used.

Note: When the bus interface clock is not active, read or write accesses to the peripheral registers are not supported. A read access will return invalid data. A write access will be ignored and will not create any bus errors.

7.5.11 Peripheral clock gating control

As mentioned previously, each peripheral requires a bus interface clock, named **rcc_perx_bus_ck** (for peripheral ‘x’). This clock can be an APB, AHB or AXI clock, according to which bus the peripheral is connected.

The clocks used as bus interface for peripherals located in D1 domain, could be **rcc_aclk**, **rcc_hclk3** or **rcc_pclk3**, depending on the bus connected to each peripheral. For simplicity sake, these clocks are named **rcc_bus_d1_ck**.

In the same way, the signal named **rcc_bus_d2_ck** represents **rcc_hclk1, rcc_hclk2**, **rcc_pclk1** or **rcc_pclk2**, depending on the bus connected to each peripheral of D2 domain.

Similarly, the signal **rcc_bus_d3_ck** represents **rcc_hclk4** or **rcc_pclk4** for peripherals located in D3.

Some peripherals (SAI, UART...) also require a dedicated clock for their communication interface. This clock is generally asynchronous with respect to the bus interface clock. It is named kernel clock (**perx_ker_ckreq**). Both clocks can be gated according to several conditions detailed hereafter.

As shown in *Figure 58*, enabling the kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN and PERxLPEN bits

  PERxEN represents the peripheral enable (allocation) bit for the CPU. The CPU can write these bits to ‘1’ via RCC_C1_xxxxENR or RCC_xxxxENR registers.
- PERxAMEN bits

  The PERxAMEN bits are belong to *RCC D3 Autonomous mode Register (RCC_D3AMR)*.
- CPU state (**c_sleep** and **c_deepsleep** signals)
- D3 domain state (**d3_deepsleep** signal)
- The kernel clock request (**perx_ker_ckreq**) of the peripheral itself, when the feature is available.

Refer to *Section 7.5.10: Peripheral allocation* for more details.

Figure 58. Peripheral kernel clock enable logic details

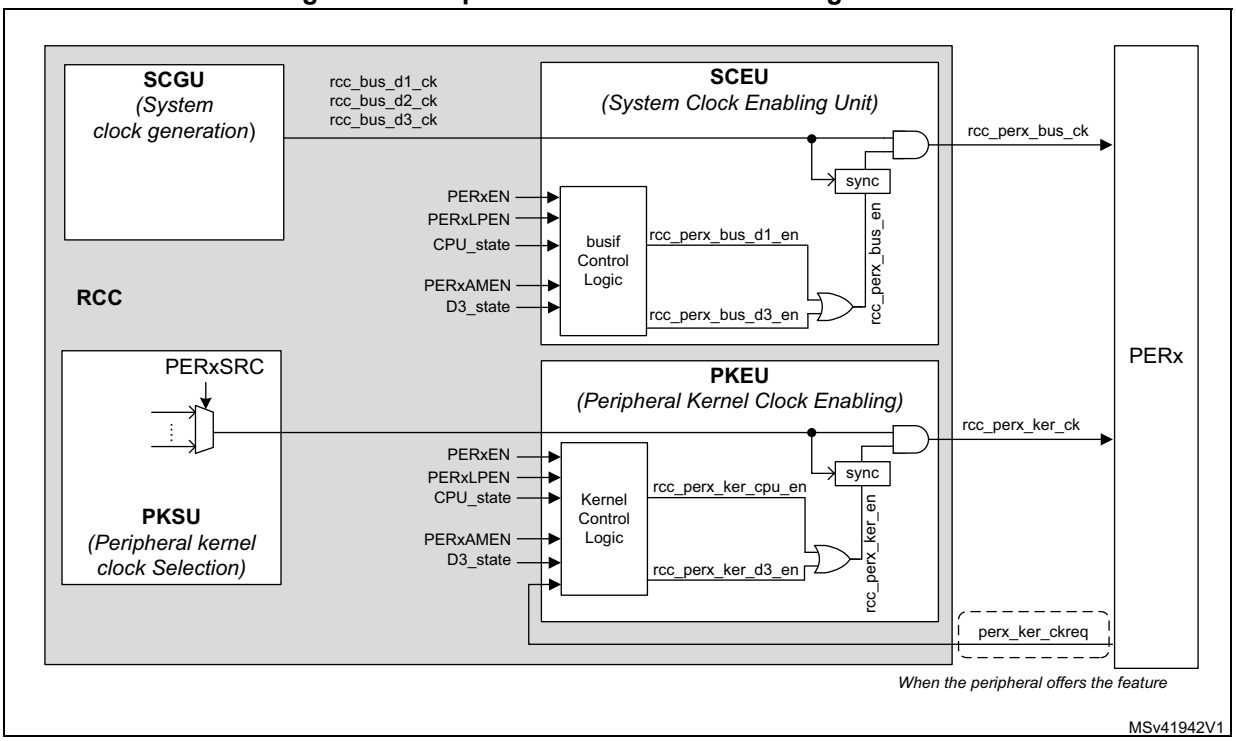


Table 57 gives a detailed description of the enabling logic of the peripheral clocks for peripherals located in D1 or D2 domain and allocated by the CPU.

Table 57. Peripheral clock enabling for D1 and D2 peripherals

PERxEN	PERxLPEN	PERxSRC	perx_ker_ckreq	CPU State	rcc_perx_ker_c_en	rcc_perx_bus_d1_en	Comments
0	X	X	X	X	0	0	No clock provided to the peripheral, because PERxEN='0'
1	X	X	X	CRun	1	1	Kernel and bus interface clocks are provided to the peripheral, because the CPU is in CRun, and PERxEN='1'
1	0	X	X	CSleep	0	0	No clock provided to the peripheral, because the CPU is in CSleep, and PERxLPEN='0'
1	1	X	X		1	1	Kernel and bus interface clocks are provided to the peripheral, because CPU is in CSleep, and PERxLPEN='1'
1	0	X	X	CStop	0	0	No clock provided to the peripheral because the PERxLPEN bit is set to '0'.
1	1	no lsi_ck and no lse_ck and no hsi_ker_ck and no csi_ker_ck	X		0	0	No clock provided to the peripheral because CPU is in CStop and **lse_ck** or **lsi_ck** or **hsi_ker_ck** or **csi_ker_ck** are not selected.
1	1	lsi_ck or lse_ck	X		1 (1)	0	Kernel clock is provided to the peripheral because PERxEN = PERxLPEN='1' and **lsi_ck** or **lse_ck** are selected. The bus interface clock is no provided as the CPU is in CStop
1	1	hsi_ker_ck or csi_ker_ck	1		1	0	Kernel clock is provided to the peripheral because **req_ker_perx** = '1', and PERxEN = PERxLPEN='1' and **hsi_ker_ck** or **csi_ker_ck** are selected. The bus interface clock is no provided as the CPU is in CStop
1	1	hsi_ker_ck or csi_ker_ck	0		0	0	No clock provided to the peripheral because CPU is in CStop, and no kernel clock request pending

1. For RNG block, the kernel clock is not delivered if the CPU to which it is allocated is in CStop mode, even if the clock selected is lsi_ck or lse_ck.

As a summary, we can state that the kernel clock is provided to the peripherals located on domains D1 and D2 when the following conditions are met:

1. The CPU is in CRun mode, and the peripheral is allocated.
2. The CPU is in CSleep mode, and the peripheral is allocated with PERxLPEN = '1'.
3. The CPU is in CStop mode, and the peripheral is allocated with PERxLPEN = '1', and the peripheral generates a kernel clock request, and the selected clock is **hsi_ker_ck** or **csi_ker_ck**.
4. The CPU is in CStop mode, and the peripheral is allocated with PERxLPEN = '1', and the kernel source clock of the peripheral is **lse_ck** or **lsi_ck**.

The bus interface clock will be provided to the peripherals only when conditions 1 or 2 are met.

Table 58 gives a detailed description of the enabling logic of the kernel clock for all peripherals located in D3.

Table 58. Peripheral clock enabling for D3 peripherals

PERxEN	PERxLPEN	PERxAMEN	PERxSRC	perx_ker_ckreq	CPU State	D3 State	rcc_perx_ker_d3_en	rcc_perx_bus_d3_en	Comments
0	X	X	X	X	Any	Any	0	0	No clock provided to the peripheral, as PERxEN='0'
1	X	X	X	X	CRun	DRun	1	1	Kernel and bus interface clocks are provided to the peripheral, because the CPU is in CRun, and PERxEN='1'
1	0	X	X	X	CSleep		0	0	No clock provided to the peripheral, because the CPU is in CSleep, and PERxLPEN='0'
1	1	X	X	X			1	1	Kernel and bus interface clocks are provided to the peripheral, because the CPU is in CSleep, and PERxLPEN='1'
1	X	0	X	X	CStop		0	0	As the CPU is in CStop, and PERxEN='1', then the kernel clock gating depends on D3 state and PERxAMEN bits. No clock provided to the peripheral because PERxAMEN = '0'.
1	X	1	X	X			1	1	The kernel and bus interface clocks are provided because even if the CPU is in CStop mode, D3 is in DRun mode, with PERxEN and PERxAMEN bits set to '1'.
1	X	1	not lse_ck and not lsi_ck	0		DStop	0	0	No clock provided to the peripheral, because D3 is in DStop, **req_ker_perx** = '0', and **lse_ck** or **lsi_ck** are not selected.

Table 58. Peripheral clock enabling for D3 peripherals (continued)

PERxEN	PERxLPEN	PERxAMEN	PERxSRC	perx_ker_ckreq	CPU State	D3 State	rcc_perx_ker_d3_en	rcc_perx_bus_d3_en	Comments
1	X	1	not hsi_ker_ck and not csi_ker_ck and not lse_ck and not lsi_ck	1	CStop	DStop	0	0	No clock provided to the peripheral, because even if **req_ker_perx** = '0', **lse_ck** or **lsi_ck** or **hsi_ker_ck** or **csi_ker_ck** are not selected.
1	X	1	hsi_ker_ck or csi_ker_ck	1			1	0	Kernel clock is provided to the peripheral because **req_ker_perx** = '1', and PERxEN = PERxAMEN='1', and the selected clock is **hsi_ker_ck** or **csi_ker_ck**. The bus interface clock is not provided as D3 is in DStop.
1	X	1	lse_ck or lsi_ck	X			1	0	Kernel clock is provided to the peripheral because PERxEN = PERxAMEN='1' and **lse_ck** or **lsi_ck** are selected, while D3 is in STOP. The bus interface clock is not provided as D3 is in DSTOP.

As a summary, we can state that the kernel clock is provided to the peripherals of D3 if the following conditions are met:

1. The CPU is in CRun mode, and the peripheral is allocated.
2. The CPU is in CSleep mode, and the peripheral is allocated with PERxLPEN = '1'.
3. The CPU is in CStop mode, and the peripheral is allocated and D3 domain is in DRun mode with PERxAMEN = '1'.
4. The CPU is in CStop mode, and the peripheral is allocated, and D3 domain is in DStop mode with PERxAMEN = '1', and the peripheral is generating a kernel clock request and the kernel clock source is **hsi_ker_ck** or **csi_ker_ck**.
5. The CPU is in CStop mode, and the peripheral is allocated, and D3 domain is in DStop mode with PERxAMEN = '1', and the kernel clock source of the peripheral is **lse_ck** or **lsi_ck**.

The bus interface clock will be provided to the peripherals only when condition 1, 2 or 3 is met.

Note: When they are set to '1', the autonomous bits indicate that the associated peripheral will receive a kernel clock according to D3 state, and not according to the mode of the CPU.

Only I2C, U(S)ART and LPUART peripherals are able to request the kernel clock. This feature gives to the peripheral the capability to transfer data with an optimal power consumption.

The autonomous bits dedicated to some peripherals located in D3 domain allow the data transfer with external devices without activating the CPU.

*In order for the LPTIMER to operate with **lse_ck** or **lsi_ck** when the circuit is in Stop mode, the user application has to select the **lsi_ck** or **lse_ck** input via LPTIMxSEL fields, and set bit LPTIMxAMEN and LPTIMxLPEN to '1'.*

7.5.12 CPU and bus matrix clock gating control

For each domain it is possible to control the activation/deactivation of the CPU clock and bus matrix clock.

For information about convention naming, refer to *Section 7.5.11: Peripheral clock gating control*.

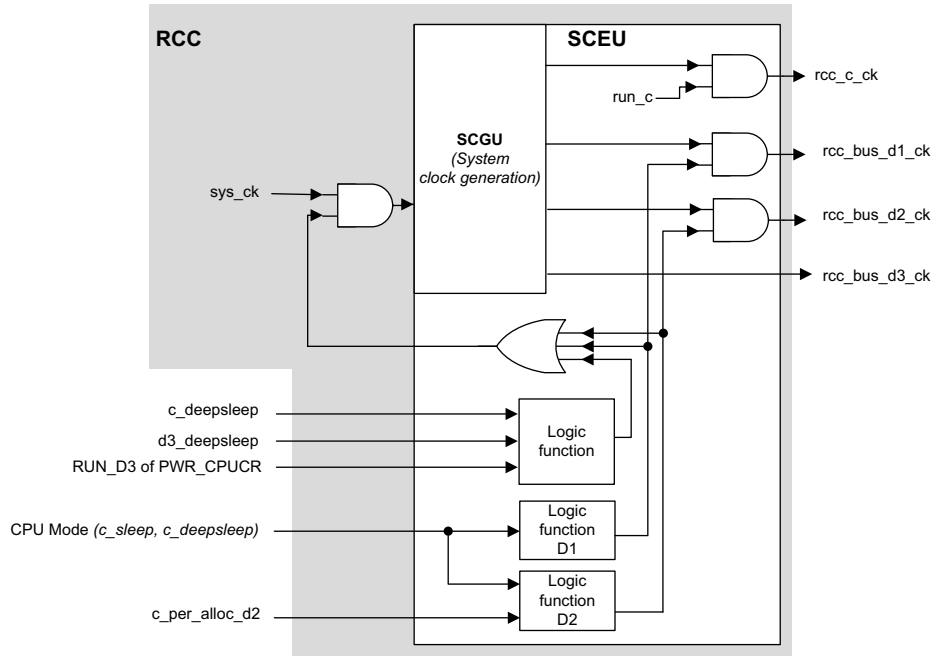
The clocks of the CPU, AHB and AXI bridges and APB busses are enabled according to the rules hereafter:

- The CPU clock **rcc_c_ck** is enabled when the CPU is in CRun mode.
- The AXI bridge clock is enabled when the CPU is in CRun mode.
- The D2 domain AHB bridges clocks are enabled when:
  - The CPU is in CRun or,
  - If the CPU is in CSleep with at least a peripheral (master) connected to this bus having both its PERxEN and PERxLPEN set to '1' or,
  - If the CPU is in CSleep with at least an APB bus having its clock enabled.
- The D3 domain AHB bridge clock is enabled when:
  - The CPU is in CRun or CSleep mode or,
  - When the RUN_D3 bit is set to '1', independently of CPU modes or,
  - When the **d3_deepsleep** signal is inactive ('0'), independently of CPU modes.
- The APB1,2,3 busses are enabled when:
  - The CPU is in CRun or,
  - If the CPU is in CSleep with at least a peripheral connected to this bus having both its PERxEN and PERxLPEN set to '1'.
- The APB4 bus is enabled when: the D3 domain is in DRun.

As shown in the *Figure 59*, the enabling of the core and bus clock of each domain depends on several input signals:

- **c_sleep** and **c_deepsleep** signals from the CPU,
- **d3_sleepdeep** signal,
- RCC_xxxxENR.PERxEN bits of peripherals located on D2 domain

Figure 59. Bus clock enable logic



7.6 RCC Interrupts

The RCC provides 3 interrupt lines:

- **rcc_it**: a general interrupt line, providing events when the PLLs are ready, or when the oscillators are ready.
- **rcc_hsecss_it**: an interrupt line dedicated to the failure detection of the HSE Clock Security System.
- **rcc_lsecss_it**: an interrupt line dedicated to the failure detection of the LSE Clock Security System.

The interrupt enable is controlled via *RCC Clock Source Interrupt Enable Register (RCC_CIER)*, except for the HSE CSS failure. When the HSE CSS feature is enabled, it not possible to mask the interrupt generation.

The interrupt flags can be checked via *RCC Clock Source Interrupt Flag Register (RCC_CIFR)*, and those flags can be cleared via *RCC Clock Source Interrupt Clear Register (RCC_CICR)*.

Note: The interrupt flags are not relevant if the corresponding interrupt enable bit is not set.

Table 59 gives a summary of the interrupt sources, and the way to control them.

Table 59. Interrupt sources and control

Interrupt Source	Description	Interrupt enable	Action to clear interrupt	Interrupt Line
LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYC to ‘1’	rcc_it
LSERDYF	LSE ready	LSERDYIE	Set LSERDYC to ‘1’	
HSIDRYF	HSI ready	HSIDRYIE	Set HSIRDYC to ‘1’	
HSERDYF	HSE ready	HSERDYIE	Set HSERDYC to ‘1’	
CSIRDYF	CSI ready	CSIRDYIE	Set CSIRDYC to ‘1’	
HSI48RDYF	HSI48 ready	HSI48RDYIE	Set HSI48RDYC to ‘1’	
PLL1RDYF	PLL1 ready	PLL1RDYIE	Set PLL1RDYC to ‘1’	
PLL2RDYF	PLL2 ready	PLL2RDYIE	Set PLL2RDYC to ‘1’	
PLL3RDYF	PLL3 ready	PLL3RDYIE	Set PLL3RDYC to ‘1’	
LSECSSF	LSE Clock security system failure	LSECSSFIE (1)	Set LSECSSC to ‘1’	rcc_lsecss_it
HSECSSF	HSE Clock security system failure	-(2)	Set HSECSSC to ‘1’	rcc_hsecss_it

1. The security system feature must also be enabled (LSECSSON = ‘1’), in order to generate interrupts.
2. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = ‘1’).

7.7 RCC register description

7.7.1 Register mapping overview

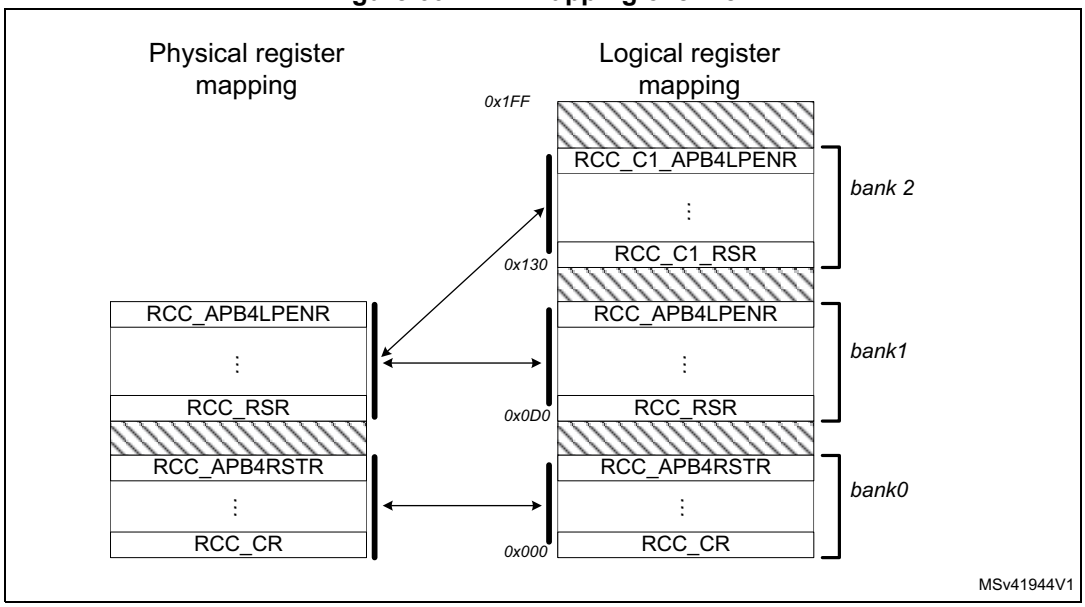
Note that the control of PERxEN and PERxLPEN bits can be performed at two different address offset:0x0D0 and 0x130. So the application can use the registers named:

- RCC_xxxxENR or RCC_C1_xxxxENR to control the PERxEN bits
- RCC_xxxxLPENR or RCC_C1_xxxxLPENR to control the PERxLPEN bits
- RCC_RSR or RCC_C1_RSR to control the reset flag status bits.

This feature is provided to insure the compatibility with other products of this family.

Figure 60 shows the RCC mapping overview.

Figure 60. RCC mapping overview



7.7.2 RCC Source Control Register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PLL3RDY	PLL3ON	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	Res.	HSECSSON	HSEBYP	HSERDY	HSEON
		r	rw	r	rw	r	rw					rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D2CKRDY	D1CKRDY	HSI48RDY	HSI48ON	Res.	Res.	CSIKERON	CSIRDY	CSION	Res.	HSIDIVF	HSIDIV[1:0]		HSIRDY	HSIKERON	HSION
r	r	r	rw			rw	r	rw		r	rw	rw	r	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **PLL3RDY**: PLL3 clock ready flag
Set by hardware to indicate that the PLL3 is locked.
0: PLL3 unlocked (default after reset)
1: PLL3 locked

Bit 28 **PLL3ON**: PLL3 enable
Set and cleared by software to enable PLL3.
Cleared by hardware when entering Stop or Standby mode.
0: PLL3 OFF (default after reset)
1: PLL3 ON

Bit 27 **PLL2RDY**: PLL2 clock ready flag
Set by hardware to indicate that the PLL2 is locked.
0: PLL2 unlocked (default after reset)
1: PLL2 locked

Bit 26 **PLL2ON**: PLL2 enable
Set and cleared by software to enable PLL2.
Cleared by hardware when entering Stop or Standby mode.
0: PLL2 OFF (default after reset)
1: PLL2 ON

Bit 25 **PLL1RDY**: PLL1 clock ready flag
Set by hardware to indicate that the PLL1 is locked.
0: PLL1 unlocked (default after reset)
1: PLL1 locked

Bit 24 **PLL1ON**: PLL1 enable
Set and cleared by software to enable PLL1.
Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to ‘0’, if the PLL1 output is used as the system clock.
0: PLL1 OFF (default after reset)
1: PLL1 ON

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **HSECSSON**: HSE Clock Security System enable
Set by software to enable Clock Security System on HSE.
This bit is “set only” (disabled by a system reset or when the system enters in Standby mode). When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.
0: Clock Security System on HSE OFF (Clock detector OFF) (default after reset)
1: Clock Security System on HSE ON (Clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 **HSEBYP**: HSE clock bypass
Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit, to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed (default after reset)
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag
Set by hardware to indicate that the HSE oscillator is stable.
0: HSE clock is not ready (default after reset)
1: HSE clock is ready

Bit 16 **HSEON**: HSE clock enable
Set and cleared by software.
Cleared by hardware to stop the HSE when entering Stop or Standby mode.
This bit cannot be cleared if the HSE is used directly (via SW mux) as system clock or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to ‘1’).
0: HSE is OFF (default after reset)
1: HSE is ON

Bit 15 **D2CKRDY**: D2 domain clocks ready flag
Set by hardware to indicate that the D2 domain clocks are available.
0: D2 domain clocks are not available (default after reset)
1: D2 domain clocks are available

Bit 14 **D1CKRDY**: D1 domain clocks ready flag
Set by hardware to indicate that the D1 domain clocks (CPU, bus and peripheral) are available.
0: D1 domain clocks are not available (default after reset)
1: D1 domain clocks are available

Bit 13 **HSI48RDY**: HSI48 clock ready flag
Set by hardware to indicate that the HSI48 oscillator is stable.
0: HSI48 clock is not ready (default after reset)
1: HSI48 clock is ready

Bit 12 **HSI48ON**: HSI48 clock enable
Set by software and cleared by software or by the hardware when the system enters to Stop or Standby mode.
0: HSI48 is OFF (default after reset)
1: HSI48 is ON

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **CSIKERON**: CSI clock enable in Stop mode

Set and reset by software to force the CSI to ON, even in Stop mode, in order to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.

0: no effect on CSI (default after reset)

1: CSI is forced to ON even in Stop mode

Bit 8 **CSIRDY**: CSI clock ready flag

Set by hardware to indicate that the CSI oscillator is stable. This bit is activated only if the RC is enabled by CSION (it is not activated if the CSI is enabled by CSIKERON or by a peripheral request).

0: CSI clock is not ready (default after reset)

1: CSI clock is ready

Bit 7 **CSION**: CSI clock enable

Set and reset by software to enable/disable CSI clock for system and/or peripheral.

Set by hardware to force the CSI to ON when the system leaves Stop mode, if STOPWUCK = ‘1’ or STOPKERWUCK = ‘1’.

This bit cannot be cleared if the CSI is used directly (via SW mux) as system clock or if the CSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to ‘1’).

0: CSI is OFF (default after reset)

1: CSI is ON

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HSIDIVF**: HSI divider flag

Set and reset by hardware.

As a write operation to HSIDIV has not an immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF will go immediately to ‘0’ when HSIDIV value is changed, and will be set back to ‘1’ when the output frequency matches the value programmed into HSIDIV.

0: new division ratio not yet propagated to **hsi(_ker)_ck** (default after reset)

1: **hsi(_ker)_ck** clock frequency reflects the new HSIDIV value

Bits 4:3 **HSIDIV[1:0]**: HSI clock divider

Set and reset by software.

These bits allow selecting a division ratio in order to configure the wanted HSI clock frequency. The HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to ‘1’). In that case, the new HSIDIV value is ignored.

00: Division by 1, **hsi(_ker)_ck** = 64 MHz (default after reset)

01: Division by 2, **hsi(_ker)_ck** = 32 MHz

10: Division by 4, **hsi(_ker)_ck** = 16 MHz

11: Division by 8, **hsi(_ker)_ck** = 8 MHz

Bit 2 **HSIRDY**: HSI clock ready flag

Set by hardware to indicate that the HSI oscillator is stable.

0: HSI clock is not ready (default after reset)

1: HSI clock is ready

Bit 1 **HSIKERON**: High Speed Internal clock enable in Stop mode

Set and reset by software to force the HSI to ON, even in Stop mode, in order to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION.

0: no effect on HSI (default after reset)

1: HSI is forced to ON even in Stop mode

Bit 0 **HSION**: High Speed Internal clock enable

Set and cleared by software.

Set by hardware to force the HSI to ON when the product leaves Stop mode, if STOPWUCK = ‘0’ or STOPKERWUCK = ‘0’.

Set by hardware to force the HSI to ON when the product leaves Standby mode or in case of a failure of the HSE which is used as the system clock source.

This bit cannot be cleared if the HSI is used directly (via SW mux) as system clock or if the HSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to ‘1’).

0: HSI is OFF

1: HSI is ON (default after reset)

7.7.3 RCC Internal Clock Source Calibration Register (RCC_ICSCR)

Address offset: 0x004

Reset value: 0x4000 0xxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CSITRIM[4:0]					CSICAL[7:0]								HSITRIM[5:4]	
	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSITRIM[3:0]				HSICAL[11:0]											
rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **CSITRIM[4:0]**: CSI clock trimming

Set by software to adjust calibration.

CSITRIM field is added to the engineering option bytes loaded during reset phase (flash_csi_opt) in order to form the calibration trimming value.

CSICAL = CSITRIM + flash_csi_opt.

Note: The reset value of the field is 0x10.

Bits 25:18 **CSICAL[7:0]**: CSI clock calibration

Set by hardware by option byte loading during system reset **nreset**.

Adjusted by software through trimming bits CSITRIM.

This field represents the sum of engineering option byte calibration value and CSITRIM bits value.

Bits 17:12 **HSITRIM[5:0]**: HSI clock trimming

Set by software to adjust calibration.

HSITRIM field is added to the engineering option bytes loaded during reset phase (flash_hsi_opt) in order to form the calibration trimming value.

HSICAL = HSITRIM + flash_hsi_opt.

Note: The reset value of the field is 0x20.

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration

Set by hardware by option byte loading during system reset **nreset**.

Adjusted by software through trimming bits HSITRIM.

This field represents the sum of engineering option byte calibration value and HSITRIM bits value.

7.7.4 RCC Clock Recovery RC Register (RCC_CRRCR)

Address offset: 0x008

Reset value: 0x0000 0xxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
						r									

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **HSI48CAL[9:0]**: Internal RC 48 MHz clock calibration

Set by hardware by option byte loading during system reset **nreset**.

Read-only.

7.7.5 RCC Clock Configuration Register (RCC_CFGR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2[2:0]			MCO2PRE[3:0]				MCO1[2:0]			MCO11PRE[3:0]				Res.	Res.
rw			rw				rw			rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMPRE	HRTIMSEL	RTCPRE[5:0]						STOPKERWUCK	STOPWUCK	SWS[2:0]			SW[2:0]		
rw	rw	rw						rw	rw	r			rw		

Bits 31:29 **MCO2[2:0]**: Micro-controller clock output 2

Set and cleared by software. Clock source selection may generate glitches on MCO2.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: System clock selected (**sys_ck**) (default after reset)
001: PLL2 oscillator clock selected (**pll2_p_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_p_ck**)
100: CSI clock selected (**csi_ck**)
101:LSI clock selected (**lsi_ck**)
others: reserved

Bits 28:25 **MCO2PRE[3:0]**: MCO2 prescaler

Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 24:22 **MCO1[2:0]**: Micro-controller clock output 1

Set and cleared by software. Clock source selection may generate glitches on MCO1.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: HSI clock selected (**hsi_ck**) (default after reset)
001: LSE oscillator clock selected (**lse_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_q_ck**)
100: HSI48 clock selected (**hsi48_ck**)
others: reserved

Bits 21:18 **MCO1PRE[3:0]**: MCO1 prescaler

Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

0000: prescaler disabled (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **TIMPRE**: Timers clocks prescaler selection

This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domains.

0: The Timers kernel clock is equal to **rcc_hclk1** if D2PPREx is corresponding to division by 1 or 2, else it is equal to 2 x $\mathbf{F_{rcc_pclkx_d2}}$ (default after reset)
1: The Timers kernel clock is equal to **rcc_hclk1** if D2PPREx is corresponding to division by 1, 2 or 4, else it is equal to 4 x $\mathbf{F_{rcc_pclkx_d2}}$
Please refer to *Table 52: Ratio between clock timer and pclk*

Bit 14 **HRTIMSEL**: High Resolution Timer clock prescaler selection

This bit is set and reset by software to control the clock frequency of high resolution the timer (HRTIM).

0: The HRTIM prescaler clock source is the same as other timers. (default after reset)
1: The HRTIM prescaler clock source is the CPU clock (**rcc_c_ck**).

Bits 13:8 **RTCPRE[5:0]**: HSE division factor for RTC clock

Set and cleared by software to divide the HSE to generate a clock for RTC.
Caution: The software has to set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.

000000: no clock (default after reset)
000001: no clock
000010: HSE/2
000011: HSE/3
000100: HSE/4
...
111110: HSE/62
111111: HSE/63

Bit 7 **STOPKERWUCK**: Kernel clock selection after a wake up from system Stop

Set and reset by software to select the Kernel wakeup clock from system Stop.

0: The HSI is selected as wake up clock from system Stop (default after reset)
1: The CSI is selected as wake up clock from system Stop
See *Section 7.5.7: Handling clock generators in Stop and Standby mode* for details.

Bit 6 **STOPWUCK**: System clock selection after a wake up from system Stop

Set and reset by software to select the system wakeup clock from system Stop.

The selected clock is also used as emergency clock for the Clock Security System on HSE.

0: The HSI is selected as wake up clock from system Stop (default after reset)

1: The CSI is selected as wake up clock from system Stop

See *Section 7.5.7: Handling clock generators in Stop and Standby mode* for details.

Caution: STOPWUCK must not be modified when the Clock Security System is enabled (by HSECSSON bit) and the system clock is HSE (SWS=”10”) or a switch on HSE is requested (SW=”10”).

Bits 5:3 **SWS[2:0]**: System clock switch status

Set and reset by hardware to indicate which clock source is used as system clock.

000: HSI used as system clock (**hsi_ck**) (default after reset)

001: CSI used as system clock (**csi_ck**)

010: HSE used as system clock (**hse_ck**)

011: PLL1 used as system clock (**pll1_p_ck**)

others: Reserved

Bits 2:0 **SW[2:0]**: System clock switch

Set and reset by software to select system clock source (**sys_ck**).

Set by hardware in order to:

– force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system Stop mode

– force the selection of the HSI in case of failure of the HSE when used directly or indirectly as system clock.

000: HSI selected as system clock (**hsi_ck**) (default after reset)

001: CSI selected as system clock (**csi_ck**)

010: HSE selected as system clock (**hse_ck**)

011: PLL1 selected as system clock (**pll1_p_ck**)

others: Reserved

7.7.6 RCC Domain 1 Clock Configuration Register (RCC_D1CFGR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	D1CPRE[3:0]				Res.	D1PPRE[2:0]			HPRE[3:0]			
				rw					rw			rw			

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **D1CPRE[3:0]**: D1 domain Core prescaler

Set and reset by software to control D1 domain CPU clock division factor.

Changing this division ratio has an impact on the frequency of the CPU clock, and all bus matrix clocks.

The clocks are divided by the new prescaler factor. This factor ranges from 1 to 16 periods of the slowest APB clock among **rcc_pclk[4:1]** after D1CPRE update. The application can check if the new division factor is taken into account by reading back this register.

0xxx: **sys_ck** not divided (default after reset)

1000: **sys_ck** divided by 2

1001: **sys_ck** divided by 4

1010: **sys_ck** divided by 8

1011: **sys_ck** divided by 16

1100: **sys_ck** divided by 64

1101: **sys_ck** divided by 128

1110: **sys_ck** divided by 256

1111: **sys_ck** divided by 512

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **D1PPRE[2:0]**: D1 domain APB3 prescaler

Set and reset by software to control the division factor of **rcc_pclk3**.
The clock is divided by the new prescaler factor from 1 to 16 cycles of **rcc_hclk3** after D1PPRE write.
0xx: **rcc_pclk3** = **rcc_hclk3** (default after reset)
100: **rcc_pclk3** = **rcc_hclk3** / 2
101: **rcc_pclk3** = **rcc_hclk3** / 4
110: **rcc_pclk3** = **rcc_hclk3** / 8
111: **rcc_pclk3** = **rcc_hclk3** / 16

Bits 3:0 **HPRE[3:0]**: D1 domain AHB prescaler

Set and reset by software to control the division factor of **rcc_hclk3** and **rcc_aclk**. Changing this division ratio has an impact on the frequency of all bus matrix clocks.
0xxx: **rcc_hclk3** = **sys_d1cpre_ck** (default after reset)
1000: **rcc_hclk3** = **sys_d1cpre_ck** / 2
1001: **rcc_hclk3** = **sys_d1cpre_ck** / 4
1010: **rcc_hclk3** = **sys_d1cpre_ck** / 8
1011: **rcc_hclk3** = **sys_d1cpre_ck** / 16
1100: **rcc_hclk3** = **sys_d1cpre_ck** / 64
1101: **rcc_hclk3** = **sys_d1cpre_ck** / 128
1110: **rcc_hclk3** = **sys_d1cpre_ck** / 256
1111: **rcc_hclk3** = **sys_d1cpre_ck** / 512

*Note: The clocks are divided by the new prescaler factor from1 to 16 periods of the slowest APB clock among **rcc_pclk[4:1]** after HPRE update.*

*Note: Note also that **rcc_hclk3** = **rcc_aclk**.*

Caution: Care must be taken when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change and before lowering the V_{CORE} voltage, this register must be read in order to check that the new prescaler value has been taken into account.

Depending on the clock source frequency and the voltage range, the software application has to program a correct value in HPRE to make sure that the system frequency does not exceed the maximum frequency.

7.7.7 RCC Domain 2 Clock Configuration Register (RCC_D2CFGR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	D2PPRE2[2:0]			Res.	D2PPRE1[2:0]			Res.	Res.	Res.	Res.
					rw				rw						

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **D2PPRE2[2:0]**: D2 domain APB2 prescaler

Set and reset by software to control D2 domain APB2 clock division factor.
The clock is divided by the new prescaler factor from 1 to 16 cycles of **rcc_hclk1** after D2PPRE2 write.
0xx: **rcc_pclk2** = **rcc_hclk1** (default after reset)
100: **rcc_pclk2** = **rcc_hclk1** / 2
101: **rcc_pclk2** = **rcc_hclk1** / 4
110: **rcc_pclk2** = **rcc_hclk1** / 8
111: **rcc_pclk2** = **rcc_hclk1** / 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **D2PPRE1[2:0]**: D2 domain APB1 prescaler

Set and reset by software to control D2 domain APB1 clock division factor.
The clock is divided by the new prescaler factor from 1 to 16 cycles of **rcc_hclk1** after D2PPRE1 write.
0xx: **rcc_pclk1** = **rcc_hclk1** (default after reset)
100: **rcc_pclk1** = **rcc_hclk1** / 2
101: **rcc_pclk1** = **rcc_hclk1** / 4
110: **rcc_pclk1** = **rcc_hclk1** / 8
111: **rcc_pclk1** = **rcc_hclk1** / 16

Bits 3:0 Reserved, must be kept at reset value.

7.7.8 RCC Domain 3 Clock Configuration Register (RCC_D3CFGR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D3PPRE[2:0]			Res.			
									rw						

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **D3PPRE[2:0]**: D3 domain APB4 prescaler

Set and reset by software to control D3 domain APB4 clock division factor.
The clock is divided by the new prescaler factor from 1 to 16 cycles of **rcc_hclk4** after D3PPRE write.
0xx: **rcc_pclk4** = **rcc_hclk4** (default after reset)
100: **rcc_pclk4** = **rcc_hclk4** / 2
101: **rcc_pclk4** = **rcc_hclk4** / 4
110: **rcc_pclk4** = **rcc_hclk4** / 8
111: **rcc_pclk4** = **rcc_hclk4** / 16

Bits 3:0 Reserved, must be kept at reset value.

7.7.9 RCC PLLs Clock Source Selection Register (RCC_PLLCKSELR)

Address offset: 0x028

Reset value: 0x0202 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	DIVM3[5:0]						Res.	Res.	DIVM2[5:4]	
						rw								rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVM2[3:0]				Res.	Res.	DIVM1[5:0]						Res.	Res.	PLLSRC[1:0]	
rw						rw								rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:20 **DIVM3[5:0]:** Prescaler for PLL3

Set and cleared by software to configure the prescaler of the PLL3.
The hardware does not allow any modification of this prescaler when PLL3 is enabled (PLL3ON = ‘1’).
In order to save power when PLL3 is not used, the value of DIVM3 must be set to ‘0’.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32 (default after reset)
...
111111: division by 63

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:12 **DIVM2[5:0]**: Prescaler for PLL2

Set and cleared by software to configure the prescaler of the PLL2.
The hardware does not allow any modification of this prescaler when PLL2 is enabled (PLL2ON = ‘1’).
In order to save power when PLL2 is not used, the value of DIVM2 must be set to ‘0’.
000000: prescaler disabled
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32 (default after reset)
...
111111: division by 63

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:4 **DIVM1[5:0]**: Prescaler for PLL1

Set and cleared by software to configure the prescaler of the PLL1.
The hardware does not allow any modification of this prescaler when PLL1 is enabled (PLL1ON = ‘1’).
In order to save power when PLL1 is not used, the value of DIVM1 must be set to ‘0’.
000000: prescaler disabled
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32 (default after reset)
...
111111: division by 63

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC[1:0]:** DIVMx and PLLs clock source selection

Set and reset by software to select the PLL clock source.
These bits can be written only when all PLLs are disabled.
In order to save power, when no PLL is used, the value of PLLSRC must be set to ‘11’.
00: HSI selected as PLL clock (**hsi_ck**) (default after reset)
01: CSI selected as PLL clock (**csi_ck**)
10: HSE selected as PLL clock (**hse_ck**)
11: No clock send to DIVMx divider and PLLs

7.7.10 RCC PLLs Configuration Register (RCC_PLLCFGR)

Address offset: 0x02C

Reset value: 0x01FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVR3EN	DIVQ3EN	DIVP3EN	DIVR2EN	DIVQ2EN	DIVP2EN	DIVR1EN	DIVQ1EN	DIVP1EN
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL3RGE[[1:0]		PLL3VCOSEL	PLL3FRACEN	PLL2RGE[[1:0]		PLL2VCOSEL	PLL2FRACEN	PLL1RGE[[1:0]		PLL1VCOSEL	PLL1FRACEN
				rw		rw	rw	rw		rw	rw	rw		rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIVR3EN**: PLL3 DIVR divider output enable
Set and reset by software to enable the **pll3_r_ck** output of the PLL3.
To save power, DIVR3EN and DIVR3 bits must be set to ‘0’ when the **pll3_r_ck** is not used.
This bit can be written only when the PLL3 is disabled (PLL3ON = ‘0’ and PLL3RDY = ‘0’).
0: **pll3_r_ck** output is disabled
1: **pll3_r_ck** output is enabled (default after reset)

Bit 23 **DIVQ3EN**: PLL3 DIVQ divider output enable
Set and reset by software to enable the **pll3_q_ck** output of the PLL3.
To save power, DIVR3EN and DIVR3 bits must be set to ‘0’ when the **pll3_r_ck** is not used.
This bit can be written only when the PLL3 is disabled (PLL3ON = ‘0’ and PLL3RDY = ‘0’).
0: **pll3_q_ck** output is disabled
1: **pll3_q_ck** output is enabled (default after reset)

Bit 22 **DIVP3EN**: PLL3 DIVP divider output enable
Set and reset by software to enable the **pll3_p_ck** output of the PLL3.
This bit can be written only when the PLL3 is disabled (PLL3ON = ‘0’ and PLL3RDY = ‘0’).
To save power, DIVR3EN and DIVR3 bits must be set to ‘0’ when the **pll3_r_ck** is not used.
0: **pll3_p_ck** output is disabled
1: **pll3_p_ck** output is enabled (default after reset)

Bit 21 **DIVR2EN**: PLL2 DIVR divider output enable
Set and reset by software to enable the **pll2_r_ck** output of the PLL2.
To save power, DIVR3EN and DIVR3 bits must be set to ‘0’ when the **pll3_r_ck** is not used.
This bit can be written only when the PLL2 is disabled (PLL2ON = ‘0’ and PLL2RDY = ‘0’).
0: **pll2_r_ck** output is disabled
1: **pll2_r_ck** output is enabled (default after reset)

Bit 20 **DIVQ2EN**: PLL2 DIVQ divider output enable
Set and reset by software to enable the **pll2_q_ck** output of the PLL2.
To save power, DIVR3EN and DIVR3 bits must be set to ‘0’ when the **pll3_r_ck** is not used.
This bit can be written only when the PLL2 is disabled (PLL2ON = ‘0’ and PLL2RDY = ‘0’).
0: **pll2_q_ck** output is disabled
1: **pll2_q_ck** output is enabled (default after reset)

Bit 19 **DIVP2EN**: PLL2 DIVP divider output enable
Set and reset by software to enable the **pll2_p_ck** output of the PLL2.
This bit can be written only when the PLL2 is disabled (PLL2ON = '0' and PLL2RDY = '0').
To save power, DIVR3EN and DIVR3 bits must be set to '0' when the **pll3_r_ck** is not used.
0: **pll2_p_ck** output is disabled
1: **pll2_p_ck** output is enabled (default after reset)

Bit 18 **DIVR1EN**: PLL1 DIVR divider output enable
Set and reset by software to enable the **pll1_r_ck** output of the PLL1.
To save power, DIVR3EN and DIVR3 bits must be set to '0' when the **pll3_r_ck** is not used.
This bit can be written only when the PLL1 is disabled (PLL1ON = '0' and PLL1RDY = '0').
0: **pll1_r_ck** output is disabled
1: **pll1_r_ck** output is enabled (default after reset)

Bit 17 **DIVQ1EN**: PLL1 DIVQ divider output enable
Set and reset by software to enable the **pll1_q_ck** output of the PLL1.
In order to save power, when the **pll1_q_ck** output of the PLL1 is not used, the **pll1_q_ck** must be disabled.
This bit can be written only when the PLL1 is disabled (PLL1ON = '0' and PLL1RDY = '0').
0: **pll1_q_ck** output is disabled
1: **pll1_q_ck** output is enabled (default after reset)

Bit 16 **DIVP1EN**: PLL1 DIVP divider output enable
Set and reset by software to enable the **pll1_p_ck** output of the PLL1.
This bit can be written only when the PLL1 is disabled (PLL1ON = '0' and PLL1RDY = '0').
In order to save power, when the **pll1_p_ck** output of the PLL1 is not used, the **pll1_p_ck** must be disabled.
0: **pll1_p_ck** output is disabled
1: **pll1_p_ck** output is enabled (default after reset)

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:10 **PLL3RGE[1:0]**: PLL3 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL3.
These bits must be written before enabling the PLL3.
00: The PLL3 input (**ref3_ck**) clock range frequency is between 1 and 2 MHz (default after reset)
01: The PLL3 input (**ref3_ck**) clock range frequency is between 2 and 4 MHz
10: The PLL3 input (**ref3_ck**) clock range frequency is between 4 and 8 MHz
11: The PLL3 input (**ref3_ck**) clock range frequency is between 8 and 16 MHz

Bit 9 **PLL3VCOSEL**: PLL3 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL3.
This bit must be written before enabling the PLL3.
0: Wide VCO range:192 to 836 MHz (default after reset)
1: Medium VCO range:150 to 420 MHz

Bit 8 **PLL3FRACEN**: PLL3 fractional latch enable
Set and reset by software to latch the content of FRACN3 into the Sigma-Delta modulator.
In order to latch the FRACN3 value into the Sigma-Delta modulator, PLL3FRACEN must be set to '0', then set to '1': the transition 0 to 1 transfers the content of FRACN3 into the modulator. Please refer to *Section : PLL initialization phase* for additional information.

Bits 7:6 **PLL2RGE[1:0]**: PLL2 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL2.
These bits must be written before enabling the PLL2.
00: The PLL2 input (**ref2_ck**) clock range frequency is between 1 and 2 MHz (default after reset)
01: The PLL2 input (**ref2_ck**) clock range frequency is between 2 and 4 MHz
10: The PLL2 input (**ref2_ck**) clock range frequency is between 4 and 8 MHz
11: The PLL2 input (**ref2_ck**) clock range frequency is between 8 and 16 MHz

Bit 5 **PLL2VCOSEL**: PLL2 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL2.
This bit must be written before enabling the PLL2.
0: Wide VCO range:192 to 836 MHz (default after reset)
1: Medium VCO range:150 to 420 MHz

Bit 4 **PLL2FRACEN**: PLL2 fractional latch enable
Set and reset by software to latch the content of FRACN2 into the Sigma-Delta modulator.
In order to latch the FRACN2 value into the Sigma-Delta modulator, PLL2FRACEN must be set to '0', then set to '1': the transition 0 to 1 transfers the content of FRACN2 into the modulator. Please refer to *Section : PLL initialization phase* for additional information.

Bits 3:2 **PLL1RGE[1:0]**: PLL1 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL1.
This bit must be written before enabling the PLL1.
00: The PLL1 input (**ref1_ck**) clock range frequency is between 1 and 2 MHz (default after reset)
01: The PLL1 input (**ref1_ck**) clock range frequency is between 2 and 4 MHz
10: The PLL1 input (**ref1_ck**) clock range frequency is between 4 and 8 MHz
11: The PLL1 input (**ref1_ck**) clock range frequency is between 8 and 16 MHz

Bit 1 **PLL1VCOSEL**: PLL1 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL1.
These bits must be written before enabling the PLL1.
0: Wide VCO range: 192 to 836 MHz (default after reset)
1: Medium VCO range: 150 to 420 MHz

Bit 0 **PLL1FRACEN**: PLL1 fractional latch enable
Set and reset by software to latch the content of FRACN1 into the Sigma-Delta modulator.
In order to latch the FRACN1 value into the Sigma-Delta modulator, PLL1FRACEN must be set to '0', then set to '1': the transition 0 to 1 transfers the content of FRACN1 into the modulator. Please refer to *Section : PLL initialization phase* for additional information.

7.7.11 RCC PLL1 Dividers Configuration Register (RCC_PLL1DIVR)

Address offset: 0x030

Reset value: 0x0101 0280

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DIVR1[6:0]							Res.	DIVQ1[6:0]						
	rw								rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVP1[6:0]								DIVN1[8:0]							
rw								rw							

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **DIVR1[6:0]**: PLL1 DIVR division factor

Set and reset by software to control the frequency of the **pll1_r_ck** clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = ‘0’ and PLL1RDY = ‘0’).
0000000: **pll1_r_ck** = **vco1_ck**
0000001: **pll1_r_ck** = **vco1_ck** / 2 (default after reset)
0000010: **pll1_r_ck** = **vco1_ck** / 3
0000011: **pll1_r_ck** = **vco1_ck** / 4
...
1111111: **pll1_r_ck** = **vco1_ck** / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **DIVQ1[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the **pll1_q_ck** clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = ‘0’ and PLL1RDY = ‘0’).
0000000: **pll1_q_ck** = **vco1_ck**
0000001: **pll1_q_ck** = **vco1_ck** / 2 (default after reset)
0000010: **pll1_q_ck** = **vco1_ck** / 3
0000011: **pll1_q_ck** = **vco1_ck** / 4
...
1111111: **pll1_q_ck** = **vco1_ck** / 128

Bits 15:9 **DIVP1[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the **pll1_p_ck** clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = ‘0’ and PLL1RDY = ‘0’).

Note that odd division factors are not allowed.

0000000: Not allowed
0000001: **pll1_p_ck** = **vco1_ck** / 2 (default after reset)
0000010: Not allowed
0000011: **pll1_p_ck** = **vco1_ck** / 4
...
1111111: **pll1_p_ck** = **vco1_ck** / 128

Bits 8:0 **DIVN1[8:0]**: Multiplication factor for PLL1 VCO

Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL1ON = ‘0’ and PLL1RDY = ‘0’).
0x003: DIVN1 = 4
0x004: DIVN1 = 5
0x005: DIVN1 = 6
...
0x080: DIVN1 = 129 (default after reset)
...
0x1FF: DIVN1 = 512
Others: wrong configurations

Caution: The software has to set correctly these bits to insure that the VCO output frequency is between its valid frequency range, which is:

- 192 to 836 MHz if PLL1VCOSEL = ‘0’
- 150 to 420 MHz if PLL1VCOSEL = ‘1’
-

VCO output frequency = F_{ref1_ck} x DIVN1, when fractional value 0 has been loaded into FRACN1, with:

- DIVN1 between 4 and 512
- The input frequency F_{ref1_ck} between 1MHz and 16 MHz

7.7.12 RCC PLL1 Fractional Divider Register (RCC_PLL1FRACR)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACN1[12:0]													Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **FRACN1[12:0]**: Fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

Caution: The software has to set correctly these bits to insure that the VCO output frequency is between its valid frequency range, which is:

- 192 to 836 MHz if PLL1VCOSEL = '0'
- 150 to 420 MHz if PLL1VCOSEL = '1'

VCO output frequency = F_{ref1_ck} x $\left(\text{DIVN1} + (\text{FRACN1} / 2^{13})\right)$, with

- DIVN1 shall be between 4 and 512
- FRACN1 can be between 0 and 2^{13}- 1
- The input frequency F_{ref1_ck} shall be between 1 and 16 MHz.

To change the FRACN value on-the-fly even if the PLL is enabled, the application has to proceed as follow:

- set the bit PLL1FRACEN to '0',
- write the new fractional value into FRACN1,
- set the bit PLL1FRACEN to '1'.

Bits 2:0 Reserved, must be kept at reset value.

7.7.13 RCC PLL2 Dividers Configuration Register (RCC_PLL2DIVR)

Address offset: 0x038

Reset value: 0x0101 0280

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DIVR2[6:0]							Res.	DIVQ2[6:0]						
	rw								rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVP2[6:0]								DIVN2[8:0]							
rw								rw							

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **DIVR2[6:0]**: PLL2 DIVR division factor

Set and reset by software to control the frequency of the **pll2_r_ck** clock.

These bits can be written only when the PLL2 is disabled (PLL2ON = ‘0’ and PLL2RDY = ‘0’).

0000000: **pll2_r_ck** = **vco2_ck**

0000001: **pll2_r_ck** = **vco2_ck** / 2 (default after reset)

0000010: **pll2_r_ck** = **vco2_ck** / 3

0000011: **pll2_r_ck** = **vco2_ck** / 4

...

1111111: **pll2_r_ck** = **vco2_ck** / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **DIVQ2[6:0]**: PLL2 DIVQ division factor

Set and reset by software to control the frequency of the **pll2_q_ck** clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = ‘0’ and PLL2RDY = ‘0’).
0000000: **pll2_q_ck** = **vco2_ck**
0000001: **pll2_q_ck** = **vco2_ck** / 2 (default after reset)
0000010: **pll2_q_ck** = **vco2_ck** / 3
0000011: **pll2_q_ck** = **vco2_ck** / 4
...
1111111: **pll2_q_ck** = **vco2_ck** / 128

Bits 15:9 **DIVP2[6:0]**: PLL2 DIVP division factor

Set and reset by software to control the frequency of the **pll2_p_ck** clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = ‘0’ and PLL2RDY = ‘0’).

0000000: **pll2_p_ck** = **vco2_ck**
0000001: **pll2_p_ck** = **vco2_ck** / 2 (default after reset)
0000010: **pll2_p_ck** = **vco2_ck** / 3
0000011: **pll2_p_ck** = **vco2_ck** / 4
...
1111111: **pll2_p_ck** = **vco2_ck** / 128

Bits 8:0 **DIVN2[8:0]**: Multiplication factor for PLL2 VCO

Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL2ON = ‘0’ and PLL2RDY = ‘0’).

Caution: The software has to set correctly these bits to insure that the VCO output frequency is between its valid frequency range, which is:

- 192 to 836 MHz if PLL2VCOSEL = ‘0’
- 150 to 420 MHz if PLL2VCOSEL = ‘1’

VCO output frequency = F_{ref2_ck} x DIVN2, when fractional value 0 has been loaded into FRACN2, with

- DIVN2 between 4 and 512
- The input frequency F_{ref2_ck} between 1MHz and 16MHz

0x003: DIVN2 = 4
0x004: DIVN2 = 5
0x005: DIVN2 = 6
...
0x080: DIVN2 = 129 (default after reset)
...
0x1FF: DIVN2 = 512
Others: wrong configurations

7.7.14 RCC PLL2 Fractional Divider Register (RCC_PLL2FRACR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACN2[12:0]													Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **FRACN2[12:0]**: Fractional part of the multiplication factor for PLL2 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

Caution: The software has to set correctly these bits to insure that the VCO output frequency is between its valid frequency range, which is:

- 192 to 836 MHz if PLL2VCOSEL = '0'
- 150 to 420 MHz if PLL2VCOSEL = '1'

VCO output frequency = F_{ref2_ck} x $\left(DIVN2 + (FRACN2 / 2^{13})\right)$, with

- DIVN2 shall be between 4 and 512
- FRACN2 can be between 0 and 2^{13} - 1
- The input frequency F_{ref2_ck} shall be between 1 and 16 MHz

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application has to proceed as follow:

- set the bit PLL2FRACEN to '0',
- write the new fractional value into FRACN2,
- set the bit PLL2FRACEN to '1'.

Bits 2:0 Reserved, must be kept at reset value.

7.7.15 RCC PLL3 Dividers Configuration Register (RCC_PLL3DIVR)

Address offset: 0x040

Reset value: 0x0101 0280

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DIVR3[6:0]							Res.	DIVQ3[6:0]						
	rw								rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVP3[6:0]								DIVN3[8:0]							
rw								rw							

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **DIVR3[6:0]**: PLL3 DIVR division factor

Set and reset by software to control the frequency of the **pll3_r_ck** clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = ‘0’ and PLL3RDY = ‘0’).
0000000: **pll3_r_ck** = **vco3_ck**
0000001: **pll3_r_ck** = **vco3_ck** / 2 (default after reset)
0000010: **pll3_r_ck** = **vco3_ck** / 3
0000011: **pll3_r_ck** = **vco3_ck** / 4
...
1111111: **pll3_r_ck** = **vco3_ck** / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **DIVQ3[6:0]**: PLL3 DIVQ division factor

Set and reset by software to control the frequency of the **pll3_q_ck** clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = ‘0’ and PLL3RDY = ‘0’).
0000000: **pll3_q_ck** = **vco3_ck**
0000001: **pll3_q_ck** = **vco3_ck** / 2 (default after reset)
0000010: **pll3_q_ck** = **vco3_ck** / 3
0000011: **pll3_q_ck** = **vco3_ck** / 4
...
1111111: **pll3_q_ck** = **vco3_ck** / 128

Bits 15:9 **DIVP3[6:0]**: PLL3 DIVP division factor

Set and reset by software to control the frequency of the **pll3_p_ck** clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = ‘0’ and PLL3RDY = ‘0’).

0000000: **pll3_p_ck** = **vco3_ck**
0000001: **pll3_p_ck** = **vco3_ck** / 2 (default after reset)
0000010: **pll3_p_ck** = **vco3_ck** / 3
0000011: **pll3_p_ck** = **vco3_ck** / 4
...
1111111: pll3_p_ck = **vco3_ck** / 128

Bits 8:0 **DIVN3[7:0]**: Multiplication factor for PLL3 VCO

Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL3ON = ‘0’ and PLL3RDY = ‘0’).

Caution: The software has to set correctly these bits to insure that the VCO output frequency is between its valid frequency range, which is:

- 192 to 836 MHz if PLL3VCOSEL = ‘0’
- 150 to 420 MHz if PLL3VCOSEL = ‘1’

VCO output frequency = F_{ref3_ck} x DIVN3, when fractional value 0 has been loaded into FRACN3, with

- DIVN3 between 4 and 512
- The input frequency F_{ref3_ck} between 1MHz and 16MHz

0x003: DIVN3 = 4
0x004: DIVN3 = 5
0x005: DIVN3 = 6
...
0x080: DIVN3 = 129 (default after reset)
...
0x1FF: DIVN3 = 512
Others: wrong configurations

7.7.16 RCC PLL3 Fractional Divider Register (RCC_PLL3FRACR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACN3[12:0]													Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **FRACN3[12:0]**: Fractional part of the multiplication factor for PLL3 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL3 VCO.

Caution: The software has to set correctly these bits to insure that the VCO output frequency is between its valid frequency range, which is:

- 192 to 836 MHz if PLL3VCOSEL = '0'
- 150 to 420 MHz if PLL3VCOSEL = '1'

VCO output frequency = F_{ref3_ck} x $\left(\text{DIVN3} + (\text{FRACN3} / 2^{13})\right)$, with

- DIVN3 shall be between 4 and 512
- FRACN3 can be between 0 and 2^{13} - 1
- The input frequency F_{ref3_ck} shall be between 1 and 16 MHz

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application has to proceed as follow:

- set the bit PLL1FRACEN to '0',
- write the new fractional value into FRACN1,
- set the bit PLL1FRACEN to '1'.

Bits 2:0 Reserved, must be kept at reset value.

7.7.17 RCC Domain 1 Kernel Clock Configuration Register (RCC_D1CCIPR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CKPERSEL[1:0] (1)		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMCSEL (1)
		rw													rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPISEL[1:0] (1)		Res.	Res.	FMCSEL[1:0] (1)	
										rw				rw	

1. Changing the clock source on-the-fly is allowed and will not generate any timing violation. However the user has to make use that both the previous and the new clock sources are present during the switching, and during the whole transition time. Please refer to *Section : Clock switches and gating*.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **CKPERSEL[1:0]**: **per_ck** clock source selection
- 00: **hsi_ker_ck** clock selected as **per_ck** clock (default after reset)
- 01: **csi_ker_ck** clock selected as **per_ck** clock
- 10: **hse_ck** clock selected as **per_ck** clock
- 11: reserved, the **per_ck** clock is disabled

Bits 27:17 Reserved, must be kept at reset value.

Bit 16 **SDMMCSEL**: SDMMC kernel clock source selection
- 0: **pll1_q_ck** clock is selected as kernel peripheral clock (default after reset)
- 1: **pll2_r_ck** clock is selected as kernel peripheral clock

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **QSPISEL[1:0]**: QUADSPI kernel clock source selection
- 00: **rcc_hclk3** clock selected as kernel peripheral clock (default after reset)
- 01: **pll1_q_ck** clock selected as kernel peripheral clock
- 10: **pll2_r_ck** clock selected as kernel peripheral clock
- 11: **per_ck** clock selected as kernel peripheral clock

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **FMCSEL[1:0]**: FMC kernel clock source selection
- 00: **rcc_hclk3** clock selected as kernel peripheral clock (default after reset)
- 01: **pll1_q_ck** clock selected as kernel peripheral clock
- 10: **pll2_r_ck** clock selected as kernel peripheral clock
- 11: **per_ck** clock selected as kernel peripheral clock

7.7.18 RCC Domain 2 Kernel Clock Configuration Register (RCC_D2CCIP1R)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWPSEL (1)	Res.	FDCANSEL[1:0] (1)		Res.	Res.	Res.	DFSDM1SEL (1)	Res.	Res.	SPDIFSEL[1:0] (1)		Res.	SPI45SEL[2:0] (1)		
rw		rw					rw			rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI123SEL[2:0] (1)			Res.	Res.	Res.	SAI23SEL[2:0] (1)			Res.	Res.	Res.	SAI1SEL[2:0] (1)		
	rw						rw						rw		

1. Changing the clock source on-the-fly is allowed and will not generate any timing violation. However the user has to make sure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Please refer to *Section : Clock switches and gating*.

Bit 31 **SWPSEL**: SWPMI kernel clock source selection
Set and reset by software.
0: **pclk** is selected as SWPMI kernel clock (default after reset)
1: **hsi_ker_ck** clock is selected as SWPMI kernel clock

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **FDCANSEL**: FDCAN kernel clock source selection
Set and reset by software.
00: **hse_ck** clock is selected as FDCAN kernel clock (default after reset)
01: **pll1_q_ck** clock is selected as FDCAN kernel clock
10: **pll2_q_ck** clock is selected as FDCAN kernel clock
11: reserved, the kernel clock is disabled

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DFSDM1SEL**: DFSDM1 kernel **Clk** clock source selection
Set and reset by software.
Note: the DFSDM1 Aclk Clock Source Selection is done by SAI1SEL.
0: **rcc_pclk2** is selected as DFSDM1 Clk kernel clock (default after reset)
1: **sys_ck** clock is selected as DFSDM1 Clk kernel clock

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **SPDIFSEL[1:0]**: SPDIFRX kernel clock source selection
00: **pll1_q_ck** clock selected as SPDIFRX kernel clock (default after reset)
01: **pll2_r_ck** clock selected as SPDIFRX kernel clock
10: **pll3_r_ck** clock selected as SPDIFRX kernel clock
11: **hsi_ker_ck** clock selected as SPDIFRX kernel clock

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **SPI45SEL[2:0]**: SPI4 and 5 kernel clock source selection
Set and reset by software.
000: APB clock is selected as kernel clock (default after reset)
001: **pll2_q_ck** clock is selected as kernel clock
010: **pll3_q_ck** clock is selected as kernel clock
011: **hsi_ker_ck** clock is selected as kernel clock
100: **csi_ker_ck** clock is selected as kernel clock
101: **hse_ck** clock is selected as kernel clock
others: reserved, the kernel clock is disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **SPI123SEL[2:0]**: SPI/I2S1,2 and 3 kernel clock source selection
Set and reset by software.

Caution: If the selected clock is the external clock and this clock is stopped, it will not be possible to switch to another clock. Refer to *Section : Clock switches and gating* for additional information.

000: **pll1_q_ck** clock selected as SPI/I2S1,2 and 3 kernel clock (default after reset)
001: **pll2_p_ck** clock selected as SPI/I2S1,2 and 3 kernel clock
010: **pll3_p_ck** clock selected as SPI/I2S1,2 and 3 kernel clock
011: I2S_CKIN clock selected as SPI/I2S1,2 and 3 kernel clock
100: **per_ck** clock selected as SPI/I2S1,2 and 3 kernel clock
others: reserved, the kernel clock is disabled

Note: I2S_CKIN is an external clock taken from a pin.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:6 **SAI23SEL[2:0]**: SAI2 and SAI3 kernel clock source selection
Set and reset by software.

Caution: If the selected clock is the external clock and this clock is stopped, it will not be possible to switch to another clock. Refer to *Section : Clock switches and gating* for additional information.

000: **pll1_q_ck** clock selected as SAI2 and SAI3 kernel clock (default after reset)
001: **pll2_p_ck** clock selected as SAI2 and SAI3 kernel clock
010: **pll3_p_ck** clock selected as SAI2 and SAI3 kernel clock
011: I2S_CKIN clock selected as SAI2 and SAI3 kernel clock
100: **per_ck** clock selected as SAI2 and SAI3 kernel clock
others: reserved, the kernel clock is disabled

Note: I2S_CKIN is an external clock taken from a pin.

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **SAI1SEL[2:0]**: SAI1 and DFSDM1 kernel **Aclk** clock source selection
Set and reset by software.

Caution: If the selected clock is the external clock and this clock is stopped, it will not be possible to switch to another clock. Refer to *Section : Clock switches and gating* for additional information.

Note: DFSDM1 Clock Source Selection is done by DFSDM1SEL.

000: **pll1_q_ck** clock selected as SAI1 and DFSDM1 **Aclk** kernel clock (default after reset)
001: **pll2_p_ck** clock selected as SAI1 and DFSDM1 **Aclk** kernel clock
010: **pll3_p_ck** clock selected as SAI1 and DFSDM1 **Aclk** kernel clock
011: I2S_CKIN clock selected as SAI1 and DFSDM1 **Aclk** kernel clock
100: **per_ck** clock selected as SAI1 and DFSDM1 **Aclk** kernel clock
others: reserved, the kernel clock is disabled

Note: I2S_CKIN is an external clock taken from a pin.

7.7.19 RCC Domain 2 Kernel Clock Configuration Register (RCC_D2CCIP2R)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LPTIM1SEL[2:0] (1)			Res.	Res.	Res.	Res.	CECSEL[1:0] (1)		USBSEL[1:0] (1)		Res.	Res.	Res.	Res.
	rw							rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C123SEL[1:0] (1)		Res.	Res.	RNGSEL[1:0] (1)		Res.	Res.	USART16SEL[2:0] (1)			USART234578SEL[2:0] (1)		
		rw				rw				rw			rw		

1. Changing the clock source on-the-fly is allowed and will not generate any timing violation. However the user has to make sure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Please refer to *Section : Clock switches and gating*.

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection
Set and reset by software.
000: **rcc_pclk1** clock selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** clock selected as kernel peripheral clock
010: **pll3_r_ck** clock selected as kernel peripheral clock
011: **lse_ck** clock selected as kernel peripheral clock
100: **lsi_ck** clock selected as kernel peripheral clock
101: **per_ck** clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:22 **CECSEL[1:0]**: HDMI-CEC kernel clock source selection
Set and reset by software.
00: **lse_ck** clock is selected as kernel clock (default after reset)
01: **lsi_ck** clock is selected as kernel clock
10: **csi_ker_ck** divided by 122 is selected as kernel clock
11: reserved, the kernel clock is disabled

Bits 21:20 **USBSEL[1:0]**: USBOTG 1 and 2 kernel clock source selection
Set and reset by software.
00: Disable the kernel clock (default after reset)
01: **pll1_q_ck** clock is selected as kernel clock
10: **pll3_q_ck** clock is selected as kernel clock
11: **hsi48_ck** clock is selected as kernel clock

Bits 19:14 Reserved, must be kept at reset value.

Bits 13:12 **I2C123SEL[1:0]**: I2C1,2,3 kernel clock source selection
Set and reset by software.
00: **rcc_pclk1** clock is selected as kernel clock (default after reset)
01: **pll3_r_ck** clock is selected as kernel clock
10: **hsi_ker_ck** clock is selected as kernel clock
11: **csi_ker_ck** clock is selected as kernel clock

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **RNGSEL[1:0]**: RNG kernel clock source selection
Set and reset by software.
00: **hsi8_ck** clock is selected as kernel clock (default after reset)
01: **pll1_q_ck** clock is selected as kernel clock
10: **lse_ck** clock is selected as kernel clock
11: **lsi_ck** clock is selected as kernel clock

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:3 **USART16SEL[2:0]**: USART1 and 6 kernel clock source selection
Set and reset by software.
000: **rcc_pclk2** clock is selected as kernel clock (default after reset)
001: **pll2_q_ck** clock is selected as kernel clock
010: **pll3_q_ck** clock is selected as kernel clock
011: **hsi_ker_ck** clock is selected as kernel clock
100: **csi_ker_ck** clock is selected as kernel clock
101: **lse_ck** clock is selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 2:0 **USART234578SEL[2:0]**: USART2/3, UART4,5, 7/8 (APB1) kernel clock source selection
Set and reset by software.
000: **rcc_pclk1** clock is selected as kernel clock (default after reset)
001: **pll2_q_ck** clock is selected as kernel clock
010: **pll3_q_ck** clock is selected as kernel clock
011: **hsi_ker_ck** clock is selected as kernel clock
100: **csi_ker_ck** clock is selected as kernel clock
101: **lse_ck** clock is selected as kernel clock
others: reserved, the kernel clock is disabled

7.7.20 RCC Domain 3 Kernel Clock Configuration Register (RCC_D3CCIPR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPI6SEL[2:0] (1)			Res.	SAI4BSEL[2:0] (1)			SAI4ASEL[2:0] (1)			Res.	Res.	Res.	ADCSEL[1:0] (1)	
	rw				rw			rw						rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM345SEL[2:0] (1)			LPTIM2SEL[2:0] (1)			I2C4SEL[1:0] (1)		Res.	Res.	Res.	Res.	Res.	LPUART1SEL[2:0] (1)		
rw			rw			rw							rw		

1. Changing the clock source on-the-fly is allowed, and will not generate any timing violation. However the user has to make sure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Please refer to *Section : Clock switches and gating*.

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SPI6SEL[2:0]**: SPI6 kernel clock source selection

Set and reset by software.

000: **rcc_pclk4** clock selected as kernel peripheral clock (default after reset)

001: **pll2_q_ck** clock selected as kernel peripheral clock

010: **pll3_q_ck** clock selected as kernel peripheral clock

011: **hsi_ker_ck** clock selected as kernel peripheral clock

100: **csi_ker_ck** clock selected as kernel peripheral clock

101: **hse_ck** clock selected as kernel peripheral clock

others: reserved, the kernel clock is disabled

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SAI4BSEL[2:0]**: Sub-Block B of SAI4 kernel clock source selection

Set and reset by software.

Caution: If the selected clock is the external clock and this clock is stopped, it will not be possible to switch to another clock. Refer to *Section : Clock switches and gating* for additional information.

000: **pll1_q_ck** clock selected as kernel peripheral clock (default after reset)

001: **pll2_p_ck** clock selected as kernel peripheral clock

010: **pll3_p_ck** clock selected as kernel peripheral clock

011: I2S_CKIN clock selected as kernel peripheral clock

100: **per_ck** clock selected as kernel peripheral clock

others: reserved, the kernel clock is disabled

Note: I2S_CKIN is an external clock taken from a pin.

Bits 23:21 **SAI4ASEL[2:0]**: Sub-Block A of SAI4 kernel clock source selection

Set and reset by software.

Caution: If the selected clock is the external clock and this clock is stopped, it will not be possible to switch to another clock. Refer to *Section : Clock switches and gating* for additional information.

000: **pll1_q_ck** clock selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** clock selected as kernel peripheral clock
010: **pll3_p_ck** clock selected as kernel peripheral clock
011: I2S_CKIN clock selected as kernel peripheral clock
100: **per_ck** clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Note: I2S_CKIN is an external clock taken from a pin.

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **ADCSEL[1:0]**: SAR ADC kernel clock source selection

Set and reset by software.
00: **pll2_p_ck** clock selected as kernel peripheral clock (default after reset)
01: **pll3_r_ck** clock selected as kernel peripheral clock
10: **per_ck** clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bits 15:13 **LPTIM345SEL[2:0]**: LPTIM3,4,5 kernel clock source selection

Set and reset by software.
000: **rcc_pclk4** clock selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** clock selected as kernel peripheral clock
010: **pll3_r_ck** clock selected as kernel peripheral clock
011: **lse_ck** clock selected as kernel peripheral clock
100: **lsi_ck** clock selected as kernel peripheral clock
101: **per_ck** clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bits 12:10 **LPTIM2SEL[2:0]**: LPTIM2 kernel clock source selection

Set and reset by software.
000: **rcc_pclk4** clock selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** clock selected as kernel peripheral clock
010: **pll3_r_ck** clock selected as kernel peripheral clock
011: **lse_ck** clock selected as kernel peripheral clock
100: **lsi_ck** clock selected as kernel peripheral clock
101: **per_ck** clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

Bits 9:8 **I2C4SEL[1:0]**: I2C4 kernel clock source selection
Set and reset by software.
00: **rcc_pclk4** clock selected as kernel peripheral clock (default after reset)
01: **pll3_r_ck** clock selected as kernel peripheral clock
10: **hsi_ker_ck** clock selected as kernel peripheral clock
11: **csi_ker_ck** clock selected as kernel peripheral clock

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **LPUART1SEL[2:0]**: LPUART1 kernel clock source selection
Set and reset by software.
000: **rcc_pclk_d3** clock is selected as kernel peripheral clock (default after reset)
001: **pll2_q_ck** clock is selected as kernel peripheral clock
010: **pll3_q_ck** clock is selected as kernel peripheral clock
011: **hsi_ker_ck** clock is selected as kernel peripheral clock
100: **csi_ker_ck** clock is selected as kernel peripheral clock
101: **lse_ck** clock is selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

7.7.21 RCC Clock Source Interrupt Enable Register (RCC_CIER)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSECSSIE	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LSECSSIE**: LSE clock security system Interrupt Enable

Set and reset by software to enable/disable interrupt caused by the Clock Security System on external 32 kHz oscillator.

0: LSE CSS interrupt disabled (default after reset)

1: LSE CSS interrupt enabled

Bit 8 **PLL3RDYIE**: PLL3 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL3 lock.

0: PLL3 lock interrupt disabled (default after reset)

1: PLL3 lock interrupt enabled

Bit 7 **PLL2RDYIE**: PLL2 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL2 lock.

0: PLL2 lock interrupt disabled (default after reset)

1: PLL2 lock interrupt enabled

Bit 6 **PLL1RDYIE**: PLL1 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL1 lock.

0: PLL1 lock interrupt disabled (default after reset)

1: PLL1 lock interrupt enabled

Bit 5 **HSI48RDYIE**: HSI48 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.

0: HSI48 ready interrupt disabled (default after reset)

1: HSI48 ready interrupt enabled

Bit 4 **CSIRDYIE**: CSI ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.

0: CSI ready interrupt disabled (default after reset)

1: CSI ready interrupt enabled

Bit 3 **HSERDYIE**: HSE ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.

0: HSE ready interrupt disabled (default after reset)

1: HSE ready interrupt enabled

Bit 2 **HSIRDYIE**: HSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

7.7.22 RCC Clock Source Interrupt Flag Register (RCC_CIFR)

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSF**: HSE clock security system Interrupt Flag
Reset by software by writing HSECSSC bit.
Set by hardware in case of HSE clock failure.
0: No clock security interrupt caused by HSE clock failure (default after reset)
1: Clock security interrupt caused by HSE clock failure

Bit 9 **LSECSSF**: LSE clock security system Interrupt Flag
Reset by software by writing LSECSSC bit.
Set by hardware when a failure is detected on the external 32 kHz oscillator and LSECSSIE is set.
0: No failure detected on the external 32 kHz oscillator (default after reset)
1: A failure is detected on the external 32 kHz oscillator

Bit 8 **PLL3RDYF**: PLL3 ready Interrupt Flag
Reset by software by writing PLL3RDYC bit.
Set by hardware when the PLL3 locks and PLL3RDYIE is set.
0: No clock ready interrupt caused by PLL3 lock (default after reset)
1: Clock ready interrupt caused by PLL3 lock

Bit 7 **PLL2RDYF**: PLL2 ready Interrupt Flag
Reset by software by writing PLL2RDYC bit.
Set by hardware when the PLL2 locks and PLL2RDYIE is set.
0: No clock ready interrupt caused by PLL2 lock (default after reset)
1: Clock ready interrupt caused by PLL2 lock

Bit 6 **PLL1RDYF**: PLL1 ready Interrupt Flag
Reset by software by writing PLL1RDYC bit.
Set by hardware when the PLL1 locks and PLL1RDYIE is set.
0: No clock ready interrupt caused by PLL1 lock (default after reset)
1: Clock ready interrupt caused by PLL1 lock

Bit 5 **HSI48RDYF**: HSI48 ready Interrupt Flag
Reset by software by writing HSI48RDYC bit.
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
0: No clock ready interrupt caused by the HSI48 oscillator (default after reset)
1: Clock ready interrupt caused by the HSI48 oscillator

Bit 4 **CSIRDYF**: CSI ready Interrupt Flag
Reset by software by writing CSIRDYC bit.
Set by hardware when the CSI clock becomes stable and CSIRDYIE is set.
0: No clock ready interrupt caused by the CSI (default after reset)
1: Clock ready interrupt caused by the CSI

Bit 3 **HSERDYF**: HSE ready Interrupt Flag
Reset by software by writing HSERDYC bit.
Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE

Bit 2 **HSIRDYF**: HSI ready Interrupt Flag
Reset by software by writing HSIRDYC bit.
Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
0: No clock ready interrupt caused by the HSI (default after reset)
1: Clock ready interrupt caused by the HSI

Bit 1 **LSERDYF**: LSE ready Interrupt Flag
Reset by software by writing LSERDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF**: LSI ready Interrupt Flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
0: No clock ready interrupt caused by the LSI (default after reset)
1: Clock ready interrupt caused by the LSI

7.7.23 RCC Clock Source Interrupt Clear Register (RCC_CICR)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	CSIRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSC**: HSE clock security system Interrupt Clear
Set by software to clear HSECSSF.
Reset by hardware when clear done.
0: HSECSSF no effect (default after reset)
1: HSECSSF cleared

Bit 9 **LSECSSC**: LSE clock security system Interrupt Clear
Set by software to clear LSECSSF.
Reset by hardware when clear done.
0: LSECSSF no effect (default after reset)
1: LSECSSF cleared

Bit 8 **PLL3RDYC**: PLL3 ready Interrupt Clear
Set by software to clear PLL3RDYF.
Reset by hardware when clear done.
0: PLL3RDYF no effect (default after reset)
1: PLL3RDYF cleared

Bit 7 **PLL2RDYC**: PLL2 ready Interrupt Clear
Set by software to clear PLL2RDYF.
Reset by hardware when clear done.
0: PLL2RDYF no effect (default after reset)
1: PLL2RDYF cleared

Bit 6 **PLL1RDYC**: PLL1 ready Interrupt Clear
Set by software to clear PLL1RDYF.
Reset by hardware when clear done.
0: PLL1RDYF no effect (default after reset)
1: PLL1RDYF cleared

Bit 5 **HSI48RDYC**: HSI48 ready Interrupt Clear
Set by software to clear HSI48RDYF.
Reset by hardware when clear done.
0: HSI48RDYF no effect (default after reset)
1: HSI48RDYF cleared

Bit 4 **CSIRDYC**: CSI ready Interrupt Clear
Set by software to clear CSIRDYF.
Reset by hardware when clear done.
0: CSIRDYF no effect (default after reset)
1: CSIRDYF cleared

Bit 3 **HSERDYC**: HSE ready Interrupt Clear
Set by software to clear HSERDYF.
Reset by hardware when clear done.
0: HSERDYF no effect (default after reset)
1: HSERDYF cleared

Bit 2 **HSIRDYC**: HSI ready Interrupt Clear
Set by software to clear HSIRDYF.
Reset by hardware when clear done.
0: HSIRDYF no effect (default after reset)
1: HSIRDYF cleared

Bit 1 **LSERDYC**: LSE ready Interrupt Clear
Set by software to clear LSERDYF.
Reset by hardware when clear done.
0: LSERDYF no effect (default after reset)
1: LSERDYF cleared

Bit 0 **LSIRDYC**: LSI ready Interrupt Clear
Set by software to clear LSIRDYF.
Reset by hardware when clear done.
0: LSIRDYF no effect (default after reset)
1: LSIRDYF cleared

7.7.24 RCC Backup Domain Control Register (RCC_BDCR)

Address offset: 0x070

Reset value: 0x0000 0000, reset by Backup domain reset.

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

After a system reset, the RCC_BDCR register is write-protected. To modify this register, the DBP bit in the *PWR control register 1 (PWR_CR1)* has to be set to '1'. RCC_BDCR bits are only reset after a backup domain reset (see *Section 7.4.6: Backup domain reset*). Any other internal or external reset will not have any effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw						rwo			r	rs	rw		rw	r	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **BDRST:** Backup domain software reset
Set and reset by software.
0: Reset not activated (default after backup domain reset)
1: Resets the entire VSW domain

Bit 15 **RTCEN**: RTC clock enable
Set and reset by software.
0: **rtc_ck** clock is disabled (default after backup domain reset)
1: **rtc_ck** clock enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**:RTC clock source selection
Set by software to select the clock source for the RTC. These bits can be written only one time (except in case of failure detection on LSE). These bits must be written before LSECSSON is enabled. The BDRST bit can be used to reset them, then it can be written one time again.
If HSE is selected as RTC clock: this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).
00: No clock (default after backup domain reset)
01: LSE clock used as RTC clock
10: LSI clock used as RTC clock
11: HSE clock divided by RTCPRE value is used as RTC clock

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LSECSSD**: LSE clock security system failure detection

Set by hardware to indicate when a failure has been detected by the Clock Security System on the external 32 kHz oscillator.

0: No failure detected on 32 kHz oscillator (default after backup domain reset)

1: Failure detected on 32 kHz oscillator

Bit 5 **LSECSSON**: LSE clock security system enable

Set by software to enable the Clock Security System on 32 kHz oscillator.

LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware), and after RTCSEL is selected.

Once enabled this bit cannot be disabled, except after a LSE failure detection (LSECSSD = '1'). In that case the software **must** disable LSECSSON.

0: Clock Security System on 32 kHz oscillator OFF (default after backup domain reset)

1: Clock Security System on 32 kHz oscillator ON

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator driving capability

Set by software to select the driving capability of the LSE oscillator.

00: Lowest drive (default after backup domain reset)

01: Medium low drive

10: Medium high drive

11: Highest drive

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and reset by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (by LSEON) or ready (LSERDY = '1')

0: LSE oscillator not bypassed (default after backup domain reset)

1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready

Set and reset by hardware to indicate when the LSE is stable. This bit needs 6 cycles of **lse_ck** clock to fall down after LSEON has been set to '0'.

0: LSE oscillator not ready (default after backup domain reset)

1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enabled

Set and reset by software.

0: LSE oscillator OFF (default after backup domain reset)

1: LSE oscillator ON

7.7.25 RCC Clock Control and Status Register (RCC_CSR)

Address offset: 0x074

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
														r	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LSIRDY**: LSI oscillator ready
Set and reset by hardware to indicate when the Low Speed Internal RC oscillator is stable.
This bit needs 3 cycles of **lsi_ck** clock to fall down after LSION has been set to '0'.
This bit can be set even when LSION is not enabled if there is a request for LSI clock by the Clock Security System on LSE or by the Low Speed Watchdog or by the RTC.
0: LSI clock is not ready (default after reset)
1: LSI clock is ready

Bit 0 **LSION**: LSI oscillator enable
Set and reset by software.
0: LSI is OFF (default after reset)
1: LSI is ON

7.7.26 RCC AHB3 Reset Register (RCC_AHB3RSTR)

Address offset: 0x07C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1RST
rs															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIRST	Res.	FMCRST	Res.	Res.	Res.	Res.	Res.	Res.	JPGDECRST	DMA2DRST	Res.	Res.	Res.	MDMARST
	rw		rw							rw	rw				rw

Bit 31 **CPURST**: CPU reset
Set by software, Reset by hardware
0: does not reset the CPU (default after reset)
1: resets the CPU, bit auto cleared to ‘0’.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC1RST**: SDMMC1 and SDMMC1 delay block reset
Set and reset by software.
0: does not reset SDMMC1 and SDMMC1 Delay block (default after reset)
1: resets SDMMC1 and SDMMC1 Delay block

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIRST**: QUADSPI and QUADSPI delay block reset
Set and reset by software.
0: does not reset QUADSPI and QUADSPI Delay block (default after reset)
1: resets QUADSPI and QUADSPI Delay block

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCRST**: FMC block reset
Set and reset by software.
0: does not reset FMC block (default after reset)
1: resets FMC block

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **JPGDECRST**: JPGDEC block reset
Set and reset by software.
0: does not reset JPGDEC block (default after reset)
1: resets JPGDEC block

Bit 4 **DMA2DRST**: DMA2D block reset
Set and reset by software.
0: does not reset DMA2D block (default after reset)
1: resets DMA2D block

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **MDMARST**: MDMA block reset
Set and reset by software.
0: does not reset MDMA block (default after reset)
1: resets MDMA block

7.7.27 RCC AHB1 Peripheral Reset Register(RCC_AHB1RSTR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	USB2OTGRST	Res.	USB1OTGRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw		rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETH1MACRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12RST	Res.	Res.	Res.	DMA2RST	DMA1RST
rw										rw				rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **USB2OTGRST**: USB2OTG (OTG_HS2) block reset
Set and reset by software.
0: does not reset USB2OTG block (default after reset)
1: resets USB2OTG block

Bit 26 Reserved, must be kept at reset value.

Bit 25 **USB1OTGRST**: USB1OTG (OTG_HS1) block reset
Set and reset by software.
0: does not reset USB1OTG block (default after reset)
1: resets USB1OTG block

Bits 24:16 Reserved, must be kept at reset value.

Bit 15 **ETH1MACRST**: ETH1MAC block reset
Set and reset by software.
0: does not reset ETH1MAC block (default after reset)
1: resets ETH1MAC block

Bits 14:6 Reserved, must be kept at reset value.

Bit 5 **ADC12RST**: ADC1 and 2 block reset
Set and reset by software.
0: does not reset ADC1 and 2 block (default after reset)
1: resets ADC1 and 2 block

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **DMA2RST**: DMA2 block reset
Set and reset by software.
0: does not reset DMA2 block (default after reset)
1: resets DMA2 block

Bit 0 **DMA1RST**: DMA1 block reset
Set and reset by software.
0: does not reset DMA1 block (default after reset)
1: resets DMA1 block

7.7.28 RCC AHB2 Peripheral Reset Register (RCC_AHB2RSTR)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2RST	Res.	Res.	RNGRST	HASHRST	CRYPTRST	Res.	Res.	Res.	CAMITFRST
						rw			rw	rw	rw				rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **SDMMC2RST**: SDMMC2 and SDMMC2 Delay block reset
Set and reset by software.
0: does not reset SDMMC2 and SDMMC2 Delay block (default after reset)
1: resets SDMMC2 and SDMMC2 Delay block

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RNGRST**: Random Number Generator block reset
Set and reset by software.
0: does not reset RNG block (default after reset)
1: resets RNG block

Bit 5 **HASHRST**: Hash block reset
Set and reset by software.
0: does not reset hash block (default after reset)
1: resets hash block

Bit 4 **CRYPTRST**: Cryptography block reset
Set and reset by software.
0: does not reset cryptography block (default after reset)
1: resets cryptography block

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **CAMITFRST**: CAMITF block reset
Set and reset by software.
0: does not reset the CAMITF block (default after reset)
1: resets the CAMITF block

7.7.29 RCC AHB4 Peripheral Reset Register (RCC_AHB4RSTR)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HSEMRST	ADC3RST	Res.	Res.	BDMARST	Res.	CRCRST	Res.	Res.	Res.
						rw	rw			rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **HSEMRST**: HSEM block reset
Set and reset by software.
0: does not reset the HSEM block (default after reset)
1: resets the HSEM block

Bit 24 **ADC3RST**: ADC3 block reset
Set and reset by software.
0: does not reset the ADC3 block (default after reset)
1: resets the ADC3 block

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **BDMARST**: BDMA block reset
Set and reset by software.
0: does not reset the BDMA block (default after reset)
1: resets the BDMA block

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CRCRST**: CRC block reset
Set and reset by software.
0: does not reset the CRC block (default after reset)
1: resets the CRC block

Bits 18:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKRST**: GPIOK block reset
Set and reset by software.
0: does not reset the GPIOK block (default after reset)
1: resets the GPIOK block

Bit 9 **GPIOJRST**: GPIOJ block reset
Set and reset by software.
0: does not reset the GPIOJ block (default after reset)
1: resets the GPIOJ block

Bit 8 **GPIOIRST**: GPIOI block reset
Set and reset by software.
0: does not reset the GPIOI block (default after reset)
1: resets the GPIOI block

Bit 7 **GPIOHRST**: GPIOH block reset
Set and reset by software.
0: does not reset the GPIOH block (default after reset)
1: resets the GPIOH block

Bit 6 **GPIOGRST**: GPIOG block reset
Set and reset by software.
0: does not reset the GPIOG block (default after reset)
1: resets the GPIOG block

Bit 5 **GPIOFRST**: GPIOF block reset
Set and reset by software.
0: does not reset the GPIOF block (default after reset)
1: resets the GPIOF block

Bit 4 **GPIOERST**: GPIOE block reset
Set and reset by software.
0: does not reset the GPIOE block (default after reset)
1: resets the GPIOE block

Bit 3 **GPIODRST**: GPIOD block reset
Set and reset by software.
0: does not reset the GPIOD block (default after reset)
1: resets the GPIOD block

Bit 2 **GPIOCRST**: GPIOC block reset
Set and reset by software.
0: does not reset the GPIOC block (default after reset)
1: resets the GPIOC block

Bit 1 **GPIOBRST**: GPIOB block reset
Set and reset by software.
0: does not reset the GPIOB block (default after reset)
1: resets the GPIOB block

Bit 0 **GPIOARST**: GPIOA block reset
Set and reset by software.
0: does not reset the GPIOA block (default after reset)
1: resets the GPIOA block

7.7.30 RCC APB3 Peripheral Reset Register (RCC_APB3RSTR)

Address offset: 0x08C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCRST	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **LTDCRST**: LTDC block reset
Set and reset by software.
0: does not reset the LTDC block (default after reset)
1: resets the LTDC block

Bits 2:0 Reserved, must be kept at reset value.

7.7.31 RCC APB1 Peripheral Reset Register (RCC_APB1LRSTR)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8RST	UART7RST	DAC12RST	Res.	HDMICECRST	Res.	Res.	Res.	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	SPDIFRXRST
rw	rw	rw		rw				rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8RST**: UART8 block reset
Set and reset by software.
0: does not reset the UART8 block (default after reset)
1: resets the UART8 block

Bit 30 **UART7RST**: UART7 block reset
Set and reset by software.
0: does not reset the UART7 block (default after reset)
1: resets the UART7 block

Bit 29 **DAC12RST**: DAC1 and 2 Blocks Reset
Set and reset by software.
0: does not reset the DAC1 and 2 blocks (default after reset)
1: resets the DAC1 and 2 blocks

Bit 28 Reserved, must be kept at reset value.

Bit 27 **HDMICECRST**: HDMI-CEC block reset
Set and reset by software.
0: does not reset the HDMI-CEC block (default after reset)
1: resets the HDMI-CEC block

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **I2C3RST**: I2C3 block reset
Set and reset by software.
0: does not reset the I2C3 block (default after reset)
1: resets the I2C3 block

Bit 22 **I2C2RST**: I2C2 block reset
Set and reset by software.
0: does not reset the I2C2 block (default after reset)
1: resets the I2C2 block

Bit 21 **I2C1RST**: I2C1 block reset
Set and reset by software.
0: does not reset the I2C1 block (default after reset)
1: resets the I2C1 block

Bit 20 **UART5RST**: UART5 block reset
Set and reset by software.
0: does not reset the UART5 block (default after reset)
1: resets the UART5 block

Bit 19 **UART4RST**: UART4 block reset
Set and reset by software.
0: does not reset the UART4 block (default after reset)
1: resets the UART4 block

Bit 18 **USART3RST**: USART3 block reset
Set and reset by software.
0: does not reset the USART3 block (default after reset)
1: resets the USART3 block

Bit 17 **USART2RST**: USART2 block reset
Set and reset by software.
0: does not reset the USART2 block (default after reset)
1: resets the USART2 block

Bit 16 **SPDIFRXRST**: SPDIFRX block reset
Set and reset by software.
0: does not reset the SPDIFRX block (default after reset)
1: resets the SPDIFRX block

Bit 15 **SPI3RST**: SPI3 block reset
Set and reset by software.
0: does not reset the SPI3 block (default after reset)
1: resets the SPI3 block

Bit 14 **SPI2RST**: SPI2 block reset
Set and reset by software.
0: does not reset the SPI2 block (default after reset)
1: resets the SPI2 block

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RST**: LPTIM1 block reset
Set and reset by software.
0: does not reset the LPTIM1 block (default after reset)
1: resets the LPTIM1 block

Bit 8 **TIM14RST**: TIM14 block reset
Set and reset by software.
0: does not reset the TIM14 block (default after reset)
1: resets the TIM14 block

Bit 7 **TIM13RST**: TIM13 block reset
Set and reset by software.
0: does not reset the TIM13 block (default after reset)
1: resets the TIM13 block

Bit 6 **TIM12RST**: TIM12 block reset
Set and reset by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block

Bit 5 **TIM7RST**: TIM7 block reset
Set and reset by software.
0: does not reset the TIM7 block (default after reset)
1: resets the TIM7 block

Bit 4 **TIM6RST**: TIM6 block reset
Set and reset by software.
0: does not reset the TIM6 block (default after reset)
1: resets the TIM6 block

Bit 3 **TIM5RST**: TIM5 block reset
Set and reset by software.
0: does not reset the TIM5 block (default after reset)
1: resets the TIM5 block

Bit 2 **TIM4RST**: TIM4 block reset
Set and reset by software.
0: does not reset the TIM4 block (default after reset)
1: resets the TIM4 block

Bit 1 **TIM3RST**: TIM3 block reset
Set and reset by software.
0: does not reset the TIM3 block (default after reset)
1: resets the TIM3 block

Bit 0 **TIM2RST**: TIM2 block reset
Set and reset by software.
0: does not reset the TIM2 block (default after reset)
1: resets the TIM2 block

7.7.32 RCC APB1 Peripheral Reset Register (RCC_APB1HRSTR)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	MDIOSRST	OPAMPRST	Res.	SWPRST	CRSRST	Res.
							rw			rw	rw		rw	rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **FDCANRST**: FDCAN block reset
Set and reset by software.
0: does not reset the FDCAN block (default after reset)
1: resets the FDCAN block

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSRST**: MDIOS block reset
Set and reset by software.
0: does not reset the MDIOS block (default after reset)
1: resets the MDIOS block

Bit 4 **OPAMPRST**: OPAMP block reset
Set and reset by software.
0: does not reset the OPAMP block (default after reset)
1: resets the OPAMP block

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SWPRST**: SWPMI block reset
Set and reset by software.
0: does not reset the SWPMI block (default after reset)
1: resets the SWPMI block

Bit 1 **CRSRST**: Clock Recovery System reset
Set and reset by software.
0: does not reset CRS (default after reset)
1: resets CRS

Bit 0 Reserved, must be kept at reset value.

7.7.33 RCC APB2 Peripheral Reset Register (RCC_APB2RSTR)

Address offset: 0x098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	HRTIMRST	DFSDM1RST	Res.	Res.	Res.	SAI3RST	SAI2RST	SAI1RST	Res.	SPI5RST	Res.	TIM17RST	TIM16RST	TIM15RST
		rw	rw				rw	rw	rw		rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SPI4RST	SPI1RST	Res.	Res.	Res.	Res.	Res.	Res.	USART6RST	USART1RST	Res.	Res.	TIM8RST	TIM1RST
		rw	rw							rw	rw			rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **HRTIMRST**: HRTIM block reset
Set and reset by software.
0: does not reset the HRTIM block (default after reset)
1: resets the HRTIM block

Bit 28 **DFSDM1RST**: DFSDM1 block reset
Set and reset by software.
0: does not reset DFSDM1 block (default after reset)
1: resets DFSDM1 block

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **SAI3RST**: SAI3 block reset
Set and reset by software.
0: does not reset the SAI3 block (default after reset)
1: resets the SAI3 block

Bit 23 **SAI2RST**: SAI2 block reset
Set and reset by software.
0: does not reset the SAI2 block (default after reset)
1: resets the SAI2 block

Bit 22 **SAI1RST**: SAI1 block reset
Set and reset by software.
0: does not reset the SAI1 (default after reset)
1: resets the SAI1

Bit 21 Reserved, must be kept at reset value.

Bit 20 **SPI5RST**: SPI5 block reset
Set and reset by software.
0: does not reset the SPI5 block (default after reset)
1: resets the SPI5 block

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST**: TIM17 block reset
Set and reset by software.
0: does not reset the TIM17 block (default after reset)
1: resets the TIM17 block

Bit 17 **TIM16RST**: TIM16 block reset
Set and reset by software.
0: does not reset the TIM16 block (default after reset)
1: resets the TIM16 block

Bit 16 **TIM15RST**: TIM15 block reset
Set and reset by software.
0: does not reset the TIM15 block (default after reset)
1: resets the TIM15 block

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SPI4RST**: SPI4 block reset
Set and reset by software.
0: does not reset the SPI4 block (default after reset)
1: resets the SPI4 block

Bit 12 **SPI1RST**: SPI1 block reset
Set and reset by software.
0: does not reset the SPI1 block (default after reset)
1: resets the SPI1 block

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **USART6RST**: USART6 block reset
Set and reset by software.
0: does not reset the USART6 block (default after reset)
1: resets the USART6 block

Bit 4 **USART1RST**: USART1 block reset
Set and reset by software.
0: does not reset the USART1 block (default after reset)
1: resets the USART1 block

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8RST**: TIM8 block reset
Set and reset by software.
0: does not reset the TIM8 block (default after reset)
1: resets the TIM8 block

Bit 0 **TIM1RST**: TIM1 block reset
Set and reset by software.
0: does not reset the TIM1 block (default after reset)
1: resets the TIM1 block

7.7.34 RCC APB4 Peripheral Reset Register (RCC_APB4RSTR)

Address offset: 0x09C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4RST	Res.	Res.	Res.	Res.	Res.
										rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFRST	COMP12RST	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST	Res.	I2C4RST	Res.	SPI6RST	Res.	LPUART1RST	Res.	SYSCFGRST	Res.
rw	rw		rw	rw	rw	rw		rw		rw		rw		rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **SAI4RST**: SAI4 block reset
Set and reset by software.
0: does not reset the SAI4 block (default after reset)
1: resets the SAI4 block

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **VREFRST**: VREF block reset
Set and reset by software.
0: does not reset the VREF block (default after reset)
1: resets the VREF block

Bit 14 **COMP12RST**: COMP12 Blocks Reset
Set and reset by software.
0: does not reset the COMP1 and 2 blocks (default after reset)
1: resets the COMP1 and 2 blocks

Bit 13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5RST**: LPTIM5 block reset
Set and reset by software.
0: does not reset the LPTIM5 block (default after reset)
1: resets the LPTIM5 block

Bit 11 **LPTIM4RST**: LPTIM4 block reset
Set and reset by software.
0: does not reset the LPTIM4 block (default after reset)
1: resets the LPTIM4 block

Bit 10 **LPTIM3RST**: LPTIM3 block reset
Set and reset by software.
0: does not reset the LPTIM3 block (default after reset)
1: resets the LPTIM3 block

Bit 9 **LPTIM2RST**: LPTIM2 block reset
Set and reset by software.
0: does not reset the LPTIM2 block (default after reset)
1: resets the LPTIM2 block

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4RST**: I2C4 block reset
Set and reset by software.
0: does not reset the I2C4 block (default after reset)
1: resets the I2C4 block

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6RST**: SPI6 block reset
Set and reset by software.
0: does not reset the SPI6 block (default after reset)
1: resets the SPI6 block

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1RST**: LPUART1 block reset
Set and reset by software.
0: does not reset the LPUART1 block (default after reset)
1: resets the LPUART1 block

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGRST**: SYSCFG block reset
Set and reset by software.
0: does not reset the SYSCFG block (default after reset)
1: resets the SYSCFG block

Bit 0 Reserved, must be kept at reset value.

7.7.35 RCC Global Control Register (RCC_GCR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WW1RSC
															rw1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **WW1RSC**: WWDG1 reset scope control

This bit can be set by software but is cleared by hardware during a system reset
In order to work properly, before enabling the WWDG1, this bit must be set to ‘1’.

7.7.36 RCC D3 Autonomous mode Register (RCC_D3AMR)

The Autonomous mode allows providing the peripheral clocks to peripherals located in D3, even if the CPU is in CStop mode. When a peripheral is enabled, and has its autonomous bit enabled, it receives its peripheral clocks according to D3 domain state, if the CPU is in CStop mode.

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SRAM4AMEN	BKPRAMAMEN	Res.	Res.	Res.	ADC3AMEN	Res.	Res.	SAI4AMEN	Res.	CRCAMEN	Res.	Res.	RTCAMEN
		rw	rw				rw			rw		rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFAMEN	COMP12AMEN	Res.	LPTIM5AMEN	LPTIM4AMEN	LPTIM3AMEN	LPTIM2AMEN	Res.	I2C4AMEN	Res.	SPI6AMEN	Res.	LPUART1AMEN	Res.	Res.	BDMAAMEN
rw	rw		rw	rw	rw	rw		rw		rw		rw			rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **SRAM4AMEN**: SRAM4 Autonomous mode enable
Set and reset by software.
0: SRAM4 clock is disabled when the CPU is in CStop (default after reset)
1: SRAM4 peripheral bus clock enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 28 **BKPSRAMAMEN**: Backup RAM Autonomous mode enable
Set and reset by software.
0: Backup RAM clock is disabled when the CPU is in CStop (default after reset)
1: Backup RAM clock enabling is controlled by D3 domain state.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 27 Reserved, must be kept at reset value.

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **ADC3AMEN**: ADC3 Autonomous mode enable
Set and reset by software.
0: ADC3 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: ADC3 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SAI4AMEN**: SAI4 Autonomous mode enable
Set and reset by software.
0: SAI4 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: SAI4 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CRCAMEN**: CRC Autonomous mode enable
Set and reset by software.
0: CRC peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: CRC peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **RTCAMEN**: RTC Autonomous mode enable
Set and reset by software.
0: RTC peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: RTC peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 15 **VREFAMEN**: VREF Autonomous mode enable
Set and reset by software.
0: VREF peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: VREF peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 14 **COMP12AMEN**: COMP12 Autonomous mode enable
Set and reset by software.
0: COMP12 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: COMP12 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5AMEN**: LPTIM5 Autonomous mode enable
Set and reset by software.
0: LPTIM5 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: LPTIM5 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 11 **LPTIM4AMEN**: LPTIM4 Autonomous mode enable
Set and reset by software.
0: LPTIM4 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: LPTIM4 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 10 **LPTIM3AMEN**: LPTIM3 Autonomous mode enable
Set and reset by software.
0: LPTIM3 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: LPTIM3 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 9 **LPTIM2AMEN**: LPTIM2 Autonomous mode enable
Set and reset by software.
0: LPTIM2 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: LPTIM2 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4AMEN**: I2C4 Autonomous mode enable
Set and reset by software.
0: I2C4 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: I2C4 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6AMEN**: SPI6 Autonomous mode enable
Set and reset by software.
0: SPI6 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: SPI6 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1AMEN**: LPUART1 Autonomous mode enable
Set and reset by software.
0: LPUART1 peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: LPUART1 peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **BDMAAMEN**: BDMA and DMAMUX Autonomous mode enable
Set and reset by software.
0: BDMA and DMAMUX peripheral clocks are disabled when the CPU is in CStop (default after reset)
1: BDMA and DMAMUX peripheral clocks enabled when D3 domain is in DRun.
Refer to *Section 7.5.11: Peripheral clock gating control* for additional information

7.7.37 RCC Reset Status Register (RCC_RSR)

This register can be accessed via two different offset address.

Table 60. RCC_RSR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_RSR	0x0D0	0x00FE 0000 (1)
RCC_C1_RSR	0x130	

1. Reset by power-on reset only

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LPWRRSTF	Res.	WWDG1RSTF	Res.	IWDG1RSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	D2RSTF	D1RSTF	Res.	CPURSTF	RMVF
	r		r		r		r	r	r	r	r	r		r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LPWRRSTF**: Reset due to illegal D1 DStandby or CPU CStop flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when D1 domain goes erroneously in DStandby or when CPU goes erroneously in CStop.
0: No illegal reset occurred (default after power-on reset)
1: Illegal D1 DStandby or CPU CStop reset occurred

Bit 29 Reserved, must be kept at reset value.

Bit 28 **WWDG1RSTF**: Window Watchdog reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a window watchdog reset occurs.
0: No window watchdog reset occurred from WWDG1 (default after power-on reset)
1: window watchdog reset occurred from WWDG1

Bit 27 Reserved, must be kept at reset value.

Bit 26 **IWDG1RSTF**: Independent Watchdog reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when an independent watchdog reset occurs.
0: No independent watchdog reset occurred (default after power-on reset)
1: Independent watchdog reset occurred

Bit 25 Reserved, must be kept at reset value.

Bit 24 **SFTRSTF**: System reset from CPU reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when the system reset is due to CPU.The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the CM7.
0: No CPU software reset occurred (default after power-on reset)
1: A system reset has been generated by the CPU

Bit 23 **PORRSTF**: POR/PDR reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a POR/PDR reset occurs.
0: No POR/PDR reset occurred
1: POR/PDR reset occurred (default after power-on reset)

Bit 22 **PINRSTF**: Pin reset flag (NRST) (1)
Reset by software by writing the RMVF bit.
Set by hardware when a reset from pin occurs.
0: No reset from pin occurred
1: Reset from pin occurred (default after power-on reset)

Bit 21 **BORRSTF**: BOR reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a BOR reset occurs (**pwr_bor_rst**).
0: No BOR reset occurred
1: BOR reset occurred (default after power-on reset)

Bit 20 **D2RSTF**: D2 domain power switch reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a D2 domain exits from DStandby or after of power-on reset. Refer to *Table 51* for details.
0: No D2 domain power switch reset occurred
1: A D2 domain power switch (ePOD2) reset occurred (default after power-on reset)

Bit 19 **D1RSTF**: D1 domain power switch reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a D1 domain exits from DStandby or after of power-on reset. Refer to *Table 51* for details.
0: No D1 domain power switch reset occurred
1: A D1 domain power switch (ePOD1) reset occurred (default after power-on reset)

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CPURSTF**: CPU reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware every time a CPU reset occurs.
0: No CPU reset occurred
1: A CPU reset occurred (default after power-on reset)

Bit 16 **RMVF**: Remove reset flag
Set and reset by software to reset the value of the reset flags.
0: Reset of the reset flags not activated (default after power-on reset)
1: Reset the value of the reset flags

Bits 15:0 Reserved, must be kept at reset value.

1. Refer to *Table 51: Reset source identification (RCC_RSR)* for details on flag behavior.

7.7.38 RCC AHB3 Clock Register (RCC_AHB3ENR)

This register can be accessed via two different offset address.

Table 61. RCC_AHB3ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB3ENR	0x0D4	0x0000 0000
RCC_C1_AHB3ENR	0x134	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIEN	Res.	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	JPGDECEN	DMA2DEN	Res.	Res.	Res.	MDMAEN
	rw		rw							rw	rw				rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC1EN**: SDMMC1 and SDMMC1 Delay Clock Enable
Set and reset by software.
0: SDMMC1 and SDMMC1 Delay clock disabled (default after reset)
1: SDMMC1 and SDMMC1 Delay clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIEN**: QUADSPI and QUADSPI Delay Clock Enable
Set and reset by software.
0: QUADSPI and QUADSPI Delay clock disabled (default after reset)
1: QUADSPI and QUADSPI Delay clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCEN**: FMC Peripheral Clocks Enable
Set and reset by software.
0: FMC peripheral clocks disabled (default after reset)
1: FMC peripheral clocks enabled
The peripheral clocks of the FMC are: the kernel clock selected by FMCSEL and provided to fmc_ker_ck input, and the **rcc_hclk3** bus interface clock.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **JPGDECEN**: JPGDEC Peripheral Clock Enable
Set and reset by software.
0: JPGDEC peripheral clock disabled (default after reset)
1: JPGDEC peripheral clock enabled

Bit 4 **DMA2DEN**: DMA2D Peripheral Clock Enable
Set and reset by software.
0: DMA2D peripheral clock disabled (default after reset)
1: DMA2D peripheral clock enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **MDMAEN**: MDMA Peripheral Clock Enable
Set and reset by software.
0: MDMA peripheral clock disabled (default after reset)
1: MDMA peripheral clock enabled

7.7.39 RCC AHB1 Clock Register (RCC_AHB1ENR)

This register can be accessed via two different offset address.

Table 62. RCC_AHB1ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB1ENR	0x0D8	0x0000 0000
RCC_C1_AHB1ENR	0x138	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	USB2OTGHSEN	USB1OTGHSULPIEN	USB1OTGHSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXEN	ETH1TXEN
				rw	rw	rw								rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETH1MACEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12EN	Res.	Res.	Res.	DMA2EN	DMA1EN
rw										rw				rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **USB2OTGHSEN**: USB2OTG (OTG_HS2) Peripheral Clocks Enable
Set and reset by software.
0: USB2OTG peripheral clocks disabled (default after reset)
1: USB2OTG peripheral clocks enabled
The peripheral clocks of the USB2OTG are: the kernel clock selected by USBSEL and the **rcc_hclk1** bus interface clock.

Bit 26 **USB1OTGHSULPIEN**: USB_PHY1 Clocks Enable
Set and reset by software.
0: USB1ULPI PHY clocks disabled (default after reset)
1: USB1ULPI PHY clocks enabled

Bit 25 **USB1OTGHSEN**: USB1OTG (OTG_HS1) Peripheral Clocks Enable
Set and reset by software.
0: USB1OTG peripheral clocks disabled (default after reset)
1: USB1OTG peripheral clocks enabled
The peripheral clocks of the USB1OTG are: the kernel clock selected by USBSEL and the **rcc_hclk1** bus interface clock.

Bits 24:18 Reserved, must be kept at reset value.

Bit 17 **ETH1RXEN**: Ethernet Reception Clock Enable
Set and reset by software.
0: Ethernet Reception clock disabled (default after reset)
1: Ethernet Reception clock enabled

Bit 16 **ETH1TXEN**: Ethernet Transmission Clock Enable
Set and reset by software.
0: Ethernet Transmission clock disabled (default after reset)
1: Ethernet Transmission clock enabled

Bit 15 **ETH1MACEN**: Ethernet MAC bus interface Clock Enable
Set and reset by software.
0: Ethernet MAC bus interface clock disabled (default after reset)
1: Ethernet MAC bus interface clock enabled

Bits 14:6 Reserved, must be kept at reset value.

Bit 5 **ADC12EN**: ADC1/2 Peripheral Clocks Enable
Set and reset by software.
0: ADC1 and 2 peripheral clocks disabled (default after reset)
1: ADC1 and 2 peripheral clocks enabled
The peripheral clocks of the ADC1 and 2 are: the kernel clock selected by ADCSEL and provided to adc_ker_ck input, and the **rcc_hclk1** bus interface clock.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **DMA2EN**: DMA2 Clock Enable
Set and reset by software.
0: DMA2 clock disabled (default after reset)
1: DMA2 clock enabled

Bit 0 **DMA1EN**: DMA1 Clock Enable
Set and reset by software.
0: DMA1 clock disabled (default after reset)
1: DMA1 clock enabled

7.7.40 RCC AHB2 Clock Register (RCC_AHB2ENR)

This register can be accessed via two different offset address.

Table 63. RCC_AHB2ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB2ENR	0x0DC	0x0000 0000
RCC_C1_AHB2ENR	0x13C	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM3EN	SRAM2EN	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2EN	Res.	Res.	RNGEN	HASHEN	CRYPTEN	Res.	Res.	Res.	DCMIEN
						rw			rw	rw	rw				rw

Bit 31 **SRAM3EN**: SRAM3 block enable
Set and reset by software.
When set, this bit indicates that the SRAM3 is allocated by the CPU. It causes the D2 domain to take into account also the CPU operation modes, i.e. keeping D2 domain in DRun when the CPU is in CRun.
0: SRAM3 interface clock is disabled. (default after reset)
1: SRAM3 interface clock is enabled.

Bit 30 **SRAM2EN**: SRAM2 block enable
Set and reset by software.
When set, this bit indicates that the SRAM2 is allocated by the CPU. It causes the D2 domain to take into account also the CPU operation modes, i.e. keeping D2 domain in DRun when the CPU is in CRun.
0: SRAM2 interface clock is disabled. (default after reset)
1: SRAM2 interface clock is enabled.

Bit 29 **SRAM1EN**: SRAM1 block enable
Set and reset by software.
When set, this bit indicates that the SRAM1 is allocated by the CPU. It causes the D2 domain to take into account also the CPU operation modes, i.e. keeping D2 domain in DRun when the CPU is in CRun.
0: SRAM1 interface clock is disabled. (default after reset)
1: SRAM1 interface clock is enabled.

Bits 28:10 Reserved, must be kept at reset value.

Bit 9 **SDMMC2EN**: SDMMC2 and SDMMC2 delay clock enable
Set and reset by software.
0: SDMMC2 and SDMMC2 Delay clock disabled (default after reset)
1: SDMMC2 and SDMMC2 Delay clock enabled

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RNGEN**: RNG peripheral clocks enable
Set and reset by software.
0: RNG peripheral clocks disabled (default after reset)
1: RNG peripheral clocks enabled:
The peripheral clocks of the RNG are: the kernel clock selected by RNGSEL and provided to **rng_ker_ck** input, and the **rcc_hclk2** bus interface clock.

Bit 5 **HASHEN**: HASH peripheral clock enable
Set and reset by software.
0: HASH peripheral clock disabled (default after reset)
1: HASH peripheral clock enabled

Bit 4 **CRYPTEN**: CRYPT peripheral clock enable
Set and reset by software.
0: CRYPT peripheral clock disabled (default after reset)
1: CRYPT peripheral clock enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIEN**: DCMI peripheral clock enable
Set and reset by software.
0: DCMI peripheral clock disabled (default after reset)
1: DCMI peripheral clock enabled

7.7.41 RCC AHB4 Clock Register (RCC_AHB4ENR)

This register can be accessed via two different offset address.

Table 64. RCC_AHB4ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB4ENR	0x0E0	0x0000 0000
RCC_C1_AHB4ENR	0x140	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKPRAMEN	Res.	Res.	HSEMEN	ADC3EN	Res.	Res.	BDMAEN	Res.	CRCEN	Res.	Res.	Res.
			rw			rw	rw			rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKPRAMEN**: Backup RAM Clock Enable
Set and reset by software.
0: Backup RAM clock disabled (default after reset)
1: Backup RAM clock enabled

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **HSEMEN**: HSEM peripheral clock enable
Set and reset by software.
0: HSEM peripheral clock disabled (default after reset)
1: HSEM peripheral clock enabled

Bit 24 **ADC3EN**: ADC3 Peripheral Clocks Enable
Set and reset by software.
0: ADC3 peripheral clocks disabled (default after reset)
1: ADC3 peripheral clocks enabled
The peripheral clocks of the ADC3 are: the kernel clock selected by ADCSEL and provided to adc_ker_ck input, and the **rcc_hclk4** bus interface clock.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **BDMAEN**: BDMA and DMAMUX2 Clock Enable
Set and reset by software.
0: BDMA and DMAMUX2 clock disabled (default after reset)
1: BDMA and DMAMUX2 clock enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CRCEN**: CRC peripheral clock enable
Set and reset by software.
0: CRC peripheral clock disabled (default after reset)
1: CRC peripheral clock enabled

Bits 18:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKEN**: GPIOK peripheral clock enable
Set and reset by software.
0: GPIOK peripheral clock disabled (default after reset)
1: GPIOK peripheral clock enabled

Bit 9 **GPIOJEN**: GPIOJ peripheral clock enable
Set and reset by software.
0: GPIOJ peripheral clock disabled (default after reset)
1: GPIOJ peripheral clock enabled

Bit 8 **GPIOIEN**: GPIOI peripheral clock enable
Set and reset by software.
0: GPIOI peripheral clock disabled (default after reset)
1: GPIOI peripheral clock enabled

Bit 7 **GPIOHEN**: GPIOH peripheral clock enable
Set and reset by software.
0: GPIOH peripheral clock disabled (default after reset)
1: GPIOH peripheral clock enabled

Bit 6 **GPIOGEN**: GPIOG peripheral clock enable
Set and reset by software.
0: GPIOG peripheral clock disabled (default after reset)
1: GPIOG peripheral clock enabled

Bit 5 **GPIOFEN**: GPIOF peripheral clock enable
Set and reset by software.
0: GPIOF peripheral clock disabled (default after reset)
1: GPIOF peripheral clock enabled

Bit 4 **GPIOEEN**: GPIOE peripheral clock enable
Set and reset by software.
0: GPIOE peripheral clock disabled (default after reset)
1: GPIOE peripheral clock enabled

Bit 3 **GPIODEN**: GPIOD peripheral clock enable
Set and reset by software.
0: GPIOD peripheral clock disabled (default after reset)
1: GPIOD peripheral clock enabled

Bit 2 **GPIOCEN**: GPIOC peripheral clock enable
Set and reset by software.
0: GPIOC peripheral clock disabled (default after reset)
1: GPIOC peripheral clock enabled

Bit 1 **GPIOBEN**: GPIOB peripheral clock enable
Set and reset by software.
0: GPIOB peripheral clock disabled (default after reset)
1: GPIOB peripheral clock enabled

Bit 0 **GPIOAEN**: GPIOA peripheral clock enable
Set and reset by software.
0: GPIOA peripheral clock disabled (default after reset)
1: GPIOA peripheral clock enabled

7.7.42 RCC APB3 Clock Register (RCC_APB3ENR)

This register can be accessed via two different offset address.

Table 65. RCC_APB3ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB3ENR	0x0E4	0x0000 0000
RCC_C1_APB3ENR	0x144	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1EN	Res.	Res.	LTDCEN	Res.	Res.	Res.
									rs			rw			

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **WWDG1EN**: WWDG1 Clock Enable
Set by software, and reset by hardware when a system reset occurs.
Note that in order to work properly, before enabling the WWDG1, the bit WW1RSC must be set to '1'.
0: WWDG1 peripheral clock disable (default after reset)
1: WWDG1 peripheral clock enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **LTDCEN**: LTDC peripheral clock enable
Provides the pixel clock (**ltdc_ker_ck**) to the LTDC block.
Set and reset by software.
0: LTDC peripheral clock disabled (default after reset)
1: LTDC peripheral clock provided to the LTDC block

Bits 2:0 Reserved, must be kept at reset value.

7.7.43 RCC APB1 Clock Register (RCC_APB1LENR)

This register can be accessed via two different offset address.

Table 66. RCC_APB1ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB1LENR	0x0E8	0x0000 0000
RCC_C1_APB1LENR	0x148	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8EN	UART7EN	DAC12EN	Res.	HDMICECEN	Res.	Res.	Res.	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	SPDIFRXEN
rw	rw	rw		rw				rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	Res.	Res.	Res.	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8EN**: UART8 Peripheral Clocks Enable
Set and reset by software.
0: UART8 peripheral clocks disable (default after reset)
1: UART8 peripheral clocks enabled
The peripheral clocks of the UART8 are: the kernel clock selected by USART234578SEL and provided to usart_ker_ck input, and the **rcc_pclk1** bus interface clock.

Bit 30 **UART7EN**: UART7 Peripheral Clocks Enable
Set and reset by software.
0: UART7 peripheral clocks disable (default after reset)
1: UART7 peripheral clocks enabled
The peripheral clocks of the UART7 are: the kernel clock selected by USART234578SEL and provided to usart_ker_ck input, and the **rcc_pclk1** bus interface clock.

Bit 29 **DAC12EN**: DAC1 and 2 peripheral clock enable
Set and reset by software.
0: DAC1 and 2 peripheral clock disable (default after reset)
1: DAC1 and 2 peripheral clock enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **HDMICECEN**: HDMI-CEC peripheral clock enable
Set and reset by software.
0: HDMI-CEC peripheral clock disable (default after reset)
1: HDMI-CEC peripheral clock enabled
The peripheral clocks of the HDMI-CEC are: the kernel clock selected by CECSEL and provided to **cec_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **I2C3EN**: I2C3 Peripheral Clocks Enable
Set and reset by software.
0: I2C3 peripheral clocks disable (default after reset)
1: I2C3 peripheral clocks enabled
The peripheral clocks of the I2C3 are: the kernel clock selected by I2C123SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 22 **I2C2EN**: I2C2 Peripheral Clocks Enable
Set and reset by software.
0: I2C2 peripheral clocks disable (default after reset)
1: I2C2 peripheral clocks enabled
The peripheral clocks of the I2C2 are: the kernel clock selected by I2C123SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 21 **I2C1EN**: I2C1 Peripheral Clocks Enable
Set and reset by software.
0: I2C1 peripheral clocks disable (default after reset)
1: I2C1 peripheral clocks enabled
The peripheral clocks of the I2C1 are: the kernel clock selected by I2C123SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 20 **UART5EN**: UART5 Peripheral Clocks Enable
Set and reset by software.
0: UART5 peripheral clocks disable (default after reset)
1: UART5 peripheral clocks enabled
The peripheral clocks of the UART5 are: the kernel clock selected by USART234578SEL and provided to **uart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 19 **UART4EN**: UART4 Peripheral Clocks Enable
Set and reset by software.
0: UART4 peripheral clocks disable (default after reset)
1: UART4 peripheral clocks enabled
The peripheral clocks of the UART4 are: the kernel clock selected by USART234578SEL and provided to **uart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 18 **USART3EN**: USART3 Peripheral Clocks Enable
Set and reset by software.
0: USART3 peripheral clocks disable (default after reset)
1: USART3 peripheral clocks enabled
The peripheral clocks of the USART3 are: the kernel clock selected by USART234578SEL and provided to **usart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 17 **USART2EN**: USART2 Peripheral Clocks Enable
Set and reset by software.
0: USART2 peripheral clocks disable (default after reset)
1: USART2 peripheral clocks enabled
The peripheral clocks of the USART2 are: the kernel clock selected by USART234578SEL and provided to **usart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 16 **SPDIFRXEN**: SPDIFRX Peripheral Clocks Enable
Set and reset by software.
0: SPDIFRX peripheral clocks disable (default after reset)
1: SPDIFRX peripheral clocks enabled
The peripheral clocks of the SPDIFRX are: the kernel clock selected by SPDIFSEL and provided to **spdifrx_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 15 **SPI3EN**: SPI3 Peripheral Clocks Enable
Set and reset by software.
0: SPI3 peripheral clocks disable (default after reset)
1: SPI3 peripheral clocks enabled
The peripheral clocks of the SPI3 are: the kernel clock selected by I2S123SRC and provided to **spi_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 14 **SPI2EN**: SPI2 Peripheral Clocks Enable
Set and reset by software.
0: SPI2 peripheral clocks disable (default after reset)
1: SPI2 peripheral clocks enabled
The peripheral clocks of the SPI2 are: the kernel clock selected by I2S123SRC and provided to **spi_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 Peripheral Clocks Enable
Set and reset by software.
0: LPTIM1 peripheral clocks disable (default after reset)
1: LPTIM1 peripheral clocks enabled
The peripheral clocks of the LPTIM1 are: the kernel clock selected by LPTIM1SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 8 **TIM14EN**: TIM14 peripheral clock enable
Set and reset by software.
0: TIM14 peripheral clock disable (default after reset)
1: TIM14 peripheral clock enabled

Bit 7 **TIM13EN**: TIM13 peripheral clock enable
Set and reset by software.
0: TIM13 peripheral clock disable (default after reset)
1: TIM13 peripheral clock enabled

Bit 6 **TIM12EN**: TIM12 peripheral clock enable
Set and reset by software.
0: TIM12 peripheral clock disable (default after reset)
1: TIM12 peripheral clock enabled

Bit 5 **TIM7EN**: TIM7 peripheral clock enable
Set and reset by software.
0: TIM7 peripheral clock disable (default after reset)
1: TIM7 peripheral clock enabled

Bit 4 **TIM6EN**: TIM6 peripheral clock enable
Set and reset by software.
0: TIM6 peripheral clock disable (default after reset)
1: TIM6 peripheral clock enabled

Bit 3 **TIM5EN**: TIM5 peripheral clock enable
Set and reset by software.
0: TIM5 peripheral clock disable (default after reset)
1: TIM5 peripheral clock enabled

Bit 2 **TIM4EN**: TIM4 peripheral clock enable
Set and reset by software.
0: TIM4 peripheral clock disable (default after reset)
1: TIM4 peripheral clock enabled

Bit 1 **TIM3EN**: TIM3 peripheral clock enable
Set and reset by software.
0: TIM3 peripheral clock disable (default after reset)
1: TIM3 peripheral clock enabled

Bit 0 **TIM2EN**: TIM2 peripheral clock enable
Set and reset by software.
0: TIM2 peripheral clock disable (default after reset)
1: TIM2 peripheral clock enabled

7.7.44 RCC APB1 Clock Register (RCC_APB1HENR)

This register can be accessed via two different offset address.

Table 67. RCC_APB1ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB1HENR	0x0EC	0x0000 0000
RCC_C1_APB1HENR	0x14C	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	MDIOSEN	OPAMPEN	Res.	SWPEN	CRSEN	Res.
							rw			rw	rw		rw	rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **FDCANEN**: FDCAN Peripheral Clocks Enable
Set and reset by software.
0: FDCAN peripheral clocks disable (default after reset)
1: FDCAN peripheral clocks enabled:
The peripheral clocks of the FDCAN are: the kernel clock selected by FDCANSEL and provided to **fdcan_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSEN**: MDIOS peripheral clock enable
Set and reset by software.
0: MDIOS peripheral clock disable (default after reset)
1: MDIOS peripheral clock enabled

Bit 4 **OPAMPEN**: OPAMP peripheral clock enable
Set and reset by software.
0: OPAMP peripheral clock disable (default after reset)
1: OPAMP peripheral clock enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SWPEN**: SWPMI Peripheral Clocks Enable
Set and reset by software.
0: SWPMI peripheral clocks disable (default after reset)
1: SWPMI peripheral clocks enabled:
The peripheral clocks of the SWPMI are: the kernel clock selected by SWPSEL and provided to **swpmi_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 1 **CRSEN**: Clock Recovery System peripheral clock enable
Set and reset by software.
0: CRS peripheral clock disable (default after reset)
1: CRS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

7.7.45 RCC APB2 Clock Register (RCC_APB2ENR)

This register can be accessed via two different offset address.

Table 68. RCC_APB2ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB2ENR	0x0F0	0x0000 0000
RCC_C1_APB2ENR	0x150	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	HRTIMEN	DFSDM1EN	Res.	Res.	Res.	SAI3EN	SAI2EN	SAI1EN	Res.	SPI5EN	Res.	TIM17EN	TIM16EN	TIM15EN
		rw	rw				rw	rw	rw		rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SPI4EN	SPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	USART6EN	USART1EN	Res.	Res.	TIM8EN	TIM1EN
		rw	rw							rw	rw			rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **HRTIMEN**: HRTIM peripheral clock enable
Set and reset by software.
0: HRTIM peripheral clock disabled (default after reset)
1: HRTIM peripheral clock enabled

Bit 28 **DFSDM1EN**: DFSDM1 Peripheral Clocks Enable
Set and reset by software.
0: DFSDM1 peripheral clocks disabled (default after reset)
1: DFSDM1 peripheral clocks enabled
DFSDM1 peripheral clocks are: the kernel clocks selected by SAI1SEL and DFSDM1SEL and provided to **Aclk** and **clk** inputs respectively, and the **rcc_pclk2** bus interface clock.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **SAI3EN**: SAI3 Peripheral Clocks Enable
Set and reset by software.
0: SAI3 peripheral clocks disabled (default after reset)
1: SAI3 peripheral clocks enabled
The peripheral clocks of the SAI3 are: the kernel clock selected by SAI23SEL and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs, and the **rcc_pclk2** bus interface clock.

Bit 23 **SAI2EN**: SAI2 Peripheral Clocks Enable
Set and reset by software.
0: SAI2 peripheral clocks disabled (default after reset)
1: SAI2 peripheral clocks enabled
The peripheral clocks of the SAI2 are: the kernel clock selected by SAI23SEL and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs, and the **rcc_pclk2** bus interface clock.

Bit 22 **SAI1EN**: SAI1 Peripheral Clocks Enable
Set and reset by software.
0: SAI1 peripheral clocks disabled (default after reset)
1: SAI1 peripheral clocks enabled:
The peripheral clocks of the SAI1 are: the kernel clock selected by SAI1SEL and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs, and the **rcc_pclk2** bus interface clock.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **SPI5EN**: SPI5 Peripheral Clocks Enable
Set and reset by software.
0: SPI5 peripheral clocks disabled (default after reset)
1: SPI5 peripheral clocks enabled:
The peripheral clocks of the SPI5 are: the kernel clock selected by SPI45SEL and provided to **spi_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM17 peripheral clock enable
Set and reset by software.
0: TIM17 peripheral clock disabled (default after reset)
1: TIM17 peripheral clock enabled

Bit 17 **TIM16EN**: TIM16 peripheral clock enable
Set and reset by software.
0: TIM16 peripheral clock disabled (default after reset)
1: TIM16 peripheral clock enabled

Bit 16 **TIM15EN**: TIM15 peripheral clock enable
Set and reset by software.
0: TIM15 peripheral clock disabled (default after reset)
1: TIM15 peripheral clock enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SPI4EN**: SPI4 Peripheral Clocks Enable
Set and reset by software.
0: SPI4 peripheral clocks disabled (default after reset)
1: SPI4 peripheral clocks enabled:
The peripheral clocks of the SPI4 are: the kernel clock selected by SPI45SEL and provided to **spi_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bit 12 **SPI1EN**: SPI1 Peripheral Clocks Enable
Set and reset by software.
0: SPI1 peripheral clocks disabled (default after reset)
1: SPI1 peripheral clocks enabled:
The peripheral clocks of the SPI1 are: the kernel clock selected by I2S123SRC and provided to **spi_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **USART6EN**: USART6 Peripheral Clocks Enable
Set and reset by software.
0: USART6 peripheral clocks disabled (default after reset)
1: USART6 peripheral clocks enabled:
The peripheral clocks of the USART6 are: the kernel clock selected by USART16SEL and provided to **usart_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bit 4 **USART1EN**: USART1 Peripheral Clocks Enable
Set and reset by software.
0: USART1 peripheral clocks disabled (default after reset)
1: USART1 peripheral clocks enabled:
The peripheral clocks of the USART1 are: the kernel clock selected by USART16SEL and provided to **usart_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8EN**: TIM8 peripheral clock enable
Set and reset by software.
0: TIM8 peripheral clock disabled (default after reset)
1: TIM8 peripheral clock enabled

Bit 0 **TIM1EN**: TIM1 peripheral clock enable
Set and reset by software.
0: TIM1 peripheral clock disabled (default after reset)
1: TIM1 peripheral clock enabled

7.7.46 RCC APB4 Clock Register (RCC_APB4ENR)

This register can be accessed via two different offset address.

Table 69. RCC_APB4ENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB4ENR	0x0F4	0x0001 0000
RCC_C1_APB4ENR	0x154	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	RTCAPBEN
										rw					rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFEN	COMP12EN	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN	Res.	I2C4EN	Res.	SPI6EN	Res.	LPUART1EN	Res.	SYSCFGEN	Res.
rw	rw		rw	rw	rw	rw		rw		rw		rw		rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **SAI4EN**: SAI4 Peripheral Clocks Enable
Set and reset by software.
0: SAI4 peripheral clocks disabled (default after reset)
1: SAI4 peripheral clocks enabled
The peripheral clocks of the SAI4 are: the kernel clocks selected by SAI4ASEL and SAI4BSEL, and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs respectively, and the **rcc_pclk4** bus interface clock.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **RTCAPBEN**: RTC APB Clock Enable
Set and reset by software.
0: The register clock interface of the RTC (APB) is disabled
1: The register clock interface of the RTC (APB) is enabled (default after reset)

Bit 15 **VREFEN**: VREF peripheral clock enable
Set and reset by software.
0: VREF peripheral clock disabled (default after reset)
1: VREF peripheral clock enabled

Bit 14 **COMP12EN**: COMP1/2 peripheral clock enable
Set and reset by software.
0: COMP1/2 peripheral clock disabled (default after reset)
1: COMP1/2 peripheral clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5EN**: LPTIM5 Peripheral Clocks Enable
Set and reset by software.
0: LPTIM5 peripheral clocks disabled (default after reset)
1: LPTIM5 peripheral clocks enabled
The peripheral clocks of the LPTIM5 are: the kernel clock selected by LPTIM345SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 11 **LPTIM4EN**: LPTIM4 Peripheral Clocks Enable
Set and reset by software.
0: LPTIM4 peripheral clocks disabled (default after reset)
1: LPTIM4 peripheral clocks enabled
The peripheral clocks of the LPTIM4 are: the kernel clock selected by LPTIM345SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 10 **LPTIM3EN**: LPTIM3 Peripheral Clocks Enable
Set and reset by software.
0: LPTIM3 peripheral clocks disabled (default after reset)
1: LPTIM3 peripheral clocks enabled
The peripheral clocks of the LPTIM3 are: the kernel clock selected by LPTIM345SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 9 **LPTIM2EN**: LPTIM2 Peripheral Clocks Enable
Set and reset by software.
0: LPTIM2 peripheral clocks disabled (default after reset)
1: LPTIM2 peripheral clocks enabled
The peripheral clocks of the LPTIM2 are: the kernel clock selected by LPTIM2SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4EN**: I2C4 Peripheral Clocks Enable
Set and reset by software.
0: I2C4 peripheral clocks disabled (default after reset)
1: I2C4 peripheral clocks enabled
The peripheral clocks of the I2C4 are: the kernel clock selected by I2C4SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6EN**: SPI6 Peripheral Clocks Enable
Set and reset by software.
0: SPI6 peripheral clocks disabled (default after reset)
1: SPI6 peripheral clocks enabled
The peripheral clocks of the SPI6 are: the kernel clock selected by SPI6SEL and provided to **spi_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1EN**: LPUART1 Peripheral Clocks Enable
Set and reset by software.
0: LPUART1 peripheral clocks disabled (default after reset)
1: LPUART1 peripheral clocks enabled
The peripheral clocks of the LPUART1 are: the kernel clock selected by LPUART1SEL and provided to **lpuart_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGEN**: SYSCFG peripheral clock enable
Set and reset by software.
0: SYSCFG peripheral clock disabled (default after reset)
1: SYSCFG peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

7.7.47 RCC AHB3 Sleep Clock Register (RCC_AHB3LPENR)

This register can be accessed via two different offset address.

Table 70. RCC_AHB3LPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB3LPENR	0x0FC	0xF001 5131
RCC_C1_AHB3LPENR	0x15C	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AXISRAMLPEN	ITCMLPEN	DTCM2LPEN	DTCM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1LPEN
rw	rw	rw	rw												rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPILPEN	Res.	FMCLPEN	Res.	Res.	Res.	FLASHLPEN	Res.	Res.	JPGDECLPEN	DMA2DLPEN	Res.	Res.	Res.	MDMALPEN
	rw		rw				rw			rw	rw				rw

Bit 31 **AXISRAMLPEN**: AXISRAM Block Clock Enable During CSleep mode
Set and reset by software.
0: AXISRAM interface clock disabled during CSleep mode
1: AXISRAM interface clock enabled during CSleep mode (default after reset)

Bit 30 **ITCMLPEN**: D1ITCM Block Clock Enable During CSleep mode
Set and reset by software.
0: D1 ITCM interface clock disabled during CSleep mode
1: D1 ITCM interface clock enabled during CSleep mode (default after reset)

Bit 29 **DTCM2LPEN**: D1 DTCM2 Block Clock Enable During CSleep mode
Set and reset by software.
0: D1 DTCM2 interface clock disabled during CSleep mode
1: D1 DTCM2 interface clock enabled during CSleep mode (default after reset)

Bit 28 **D1DTCM1LPEN**: D1DTCM1 Block Clock Enable During CSleep mode
Set and reset by software.
0: D1DTCM1 interface clock disabled during CSleep mode
1: D1DTCM1 interface clock enabled during CSleep mode (default after reset)

Bits 27:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC1LPEN**: SDMMC1 and SDMMC1 Delay Clock Enable During CSleep Mode
Set and reset by software.
0: SDMMC1 and SDMMC1 Delay clock disabled during CSleep mode
1: SDMMC1 and SDMMC1 Delay clock enabled during CSleep mode (default after reset)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPILPEN**: QUADSPI and QUADSPI Delay Clock Enable During CSleep Mode
Set and reset by software.
0: QUADSPI and QUADSPI Delay clock disabled during CSleep mode
1: QUADSPI and QUADSPI Delay clock enabled during CSleep mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCLPEN**: FMC Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: FMC peripheral clocks disabled during CSleep mode
1: FMC peripheral clocks enabled during CSleep mode (default after reset):
The peripheral clocks of the FMC are: the kernel clock selected by FMCSEL and provided to **fmc_ker_ck** input, and the **rcc_hclk3** bus interface clock.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHLPEN**: Flash interface Clock Enable During CSleep Mode
Set and reset by software.
0: Flash interface clock disabled during CSleep mode
1: Flash interface clock enabled during CSleep mode (default after reset)

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JPGDECLPEN**: JPGDEC Clock Enable During CSleep Mode
Set and reset by software.
0: JPGDEC peripheral clock disabled during CSleep mode
1: JPGDEC peripheral clock enabled during CSleep mode (default after reset)

Bit 4 **DMA2DLPEN**: DMA2D Clock Enable During CSleep Mode
Set and reset by software.
0: DMA2D peripheral clock disabled during CSleep mode
1: DMA2D peripheral clock enabled during CSleep mode (default after reset)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA Clock Enable During CSleep Mode
Set and reset by software.
0: MDMA peripheral clock disabled during CSleep mode
1: MDMA peripheral clock enabled during CSleep mode (default after reset)

7.7.48 RCC AHB1 Sleep Clock Register (RCC_AHB1LPENR)

This register can be accessed via two different offset address.

Table 71. RCC_AHB1LPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB1LPENR	0x100	0x1E03 C023
RCC_C1_AHB1LPENR	0x160	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	USB2OTGHSLPEN	USB1OTGHSULPILPEN	USB1OTGHSLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXLPEN	ETH1TXLPEN
				rw	rw	rw								rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETH1MACLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPEN	Res.	Res.	Res.	DMA2LPEN	DMA1LPEN
rw										rw				rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **USB2OTGHSLPEN**: USB2OTG (OTG_US2) peripheral clock enable during CSleep mode
Set and reset by software.
0: USB2OTG peripheral clocks disabled during CSleep mode
1: USB2OTG peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the USB2OTG are: the kernel clock selected by USBSEL and the **rcc_hclk1** bus interface clock.

Bit 26 **USB1OTGHSULPILPEN**: USB_PHY1 clock enable during CSleep mode
Set and reset by software.
0: USB_PHY1 peripheral clock disabled during CSleep mode
1: USB_PHY1 peripheral clock enabled during CSleep mode (default after reset)

Bit 25 **USB1OTGHSLPEN**: USB1OTG (OTG_HS1) peripheral clock enable during CSleep mode
Set and reset by software.
0: USB1OTG peripheral clock disabled during CSleep mode
1: USB1OTG peripheral clock enabled during CSleep mode (default after reset)
The peripheral clocks of the USB1OTG are: the kernel clock selected by USBSEL and the **rcc_hclk1** bus interface clock.

Bits 24:18 Reserved, must be kept at reset value.

Bit 17 **ETH1RXLPEN**: Ethernet Reception Clock Enable During CSleep Mode
Set and reset by software.
0: Ethernet Reception clock disabled during CSleep mode
1: Ethernet Reception clock enabled during CSleep mode (default after reset)

Bit 16 **ETH1TXLPEN**: Ethernet Transmission Clock Enable During CSleep Mode
Set and reset by software.
0: Ethernet Transmission clock disabled during CSleep mode
1: Ethernet Transmission clock enabled during CSleep mode (default after reset)

Bit 15 **ETH1MACLPEN**: Ethernet MAC bus interface Clock Enable During CSleep Mode
Set and reset by software.
0: Ethernet MAC bus interface clock disabled during CSleep mode
1: Ethernet MAC bus interface clock enabled during CSleep mode (default after reset)

Bits 14:6 Reserved, must be kept at reset value.

Bit 5 **ADC12LPEN**: ADC1/2 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: ADC1/2 peripheral clocks disabled during CSleep mode
1: ADC1/2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the ADC1 and 2 are: the kernel clock selected by ADCSEL and provided to adc_ker_ck input, and the **rcc_hclk1** bus interface clock.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **DMA2LPEN**: DMA2 Clock Enable During CSleep Mode
Set and reset by software.
0: DMA2 clock disabled during CSleep mode
1: DMA2 clock enabled during CSleep mode (default after reset)

Bit 0 **DMA1LPEN**: DMA1 Clock Enable During CSleep Mode
Set and reset by software.
0: DMA1 clock disabled during CSleep mode
1: DMA1 clock enabled during CSleep mode (default after reset)

7.7.49 RCC AHB2 Sleep Clock Register (RCC_AHB2LPENR)

This register can be accessed via two different offset address.

Table 72. RCC_AHB2LPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB2LPENR	0x104	0xE000 0271
RCC_C1_AHB2LPENR	0x164	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM3LPEN	SRAM2LPEN	SRAM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2LPEN	Res.	Res.	RNGLPEN	HASHLPEN	CRYPTLPEN	Res.	Res.	Res.	CAMITFLPEN
						rw			rw	rw	rw				rw

Bit 31 **SRAM3LPEN**: SRAM3 Clock Enable During CSleep Mode
Set and reset by software.
0: SRAM3 clock disabled during CSleep mode
1: SRAM3 clock enabled during CSleep mode (default after reset)

Bit 30 **SRAM2LPEN**: SRAM2 Clock Enable During CSleep Mode
Set and reset by software.
0: SRAM2 clock disabled during CSleep mode
1: SRAM2 clock enabled during CSleep mode (default after reset)

Bit 29 **SRAM1LPEN**: SRAM1 Clock Enable During CSleep Mode
Set and reset by software.
0: SRAM1 clock disabled during CSleep mode
1: SRAM1 clock enabled during CSleep mode (default after reset)

Bits 28:10 Reserved, must be kept at reset value.

Bit 9 **SDMMC2LPEN**: SDMMC2 and SDMMC2 Delay Clock Enable During CSleep Mode
Set and reset by software.
0: SDMMC2 and SDMMC2 Delay clock disabled during CSleep mode
1: SDMMC2 and SDMMC2 Delay clock enabled during CSleep mode (default after reset)

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RNGLPEN**: RNG peripheral clock enable during CSleep mode
Set and reset by software.
0: RNG peripheral clocks disabled during CSleep mode
1: RNG peripheral clock enabled during CSleep mode (default after reset)
The peripheral clocks of the RNG are: the kernel clock selected by RNGSEL and provided to **rng_ker_ck** input, and the **rcc_hclk2** bus interface clock.

Bit 5 **HASHLPEN**: HASH peripheral clock enable during CSleep mode
Set and reset by software.
0: HASH peripheral clock disabled during CSleep mode
1: HASH peripheral clock enabled during CSleep mode (default after reset)

Bit 4 **CRYPTLPEN**: CRYPT peripheral clock enable during CSleep mode
Set and reset by software.
0: CRYPT peripheral clock disabled during CSleep mode
1: CRYPT peripheral clock enabled during CSleep mode (default after reset)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMILPEN**: DCMI peripheral clock enable during CSleep mode
Set and reset by software.
0: DCMI peripheral clock disabled during CSleep mode
1: DCMI peripheral clock enabled during CSleep mode (default after reset)

7.7.50 RCC AHB4 Sleep Clock Register (RCC_AHB4LPENR)

This register can be accessed via two different offset address.

Table 73. RCC_AHB4LPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_AHB4LPENR	0x108	0x3128 07FF
RCC_C1_AHB4LPENR	0x168	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SRAM4LPEN	BKPRAMLPEN	Res.	Res.	Res.	ADC3LPEN	Res.	Res.	BDMALPEN	Res.	CRCLPEN	Res.	Res.	Res.
		rw	rw				rw			rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **SRAM4LPEN**: SRAM4 Clock Enable During CSleep Mode
Set and reset by software.
0: SRAM4 clock disabled during CSleep mode
1: SRAM4 clock enabled during CSleep mode (default after reset)

Bit 28 **BKPRAMLPEN**: Backup RAM Clock Enable During CSleep Mode
Set and reset by software.
0: Backup RAM clock disabled during CSleep mode
1: Backup RAM clock enabled during CSleep mode (default after reset)

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **ADC3LPEN**: ADC3 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: ADC3 peripheral clocks disabled during CSleep mode
1: ADC3 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the ADC3 are: the kernel clock selected by ADCSEL and provided to adc_ker_ck input, and the **rcc_hclk4** bus interface clock.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **BDMALPEN**: BDMA Clock Enable During CSleep Mode
Set and reset by software.
0: BDMA clock disabled during CSleep mode
1: BDMA clock enabled during CSleep mode (default after reset)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CRCLPEN**: CRC peripheral clock enable during CSleep mode
Set and reset by software.
0: CRC peripheral clock disabled during CSleep mode
1: CRC peripheral clock enabled during CSleep mode (default after reset)

Bits 18:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKLPEN**: GPIOK peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOK peripheral clock disabled during CSleep mode
1: GPIOK peripheral clock enabled during CSleep mode (default after reset)

Bit 9 **GPIOJLPEN**: GPIOJ peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOJ peripheral clock disabled during CSleep mode
1: GPIOJ peripheral clock enabled during CSleep mode (default after reset)

Bit 8 **GPIOILPEN**: GPIOI peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOI peripheral clock disabled during CSleep mode
1: GPIOI peripheral clock enabled during CSleep mode (default after reset)

Bit 7 **GPIOHLPEN**: GPIOH peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOH peripheral clock disabled during CSleep mode
1: GPIOH peripheral clock enabled during CSleep mode (default after reset)

Bit 6 **GPIOGLPEN**: GPIOG peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOG peripheral clock disabled during CSleep mode
1: GPIOG peripheral clock enabled during CSleep mode (default after reset)

Bit 5 **GPIOFLPEN**: GPIOF peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOF peripheral clock disabled during CSleep mode
1: GPIOF peripheral clock enabled during CSleep mode (default after reset)

Bit 4 **GPIOELPEN**: GPIOE peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOE peripheral clock disabled during CSleep mode
1: GPIOE peripheral clock enabled during CSleep mode (default after reset)

Bit 3 **GPIODLPEN**: GPIOD peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOD peripheral clock disabled during CSleep mode
1: GPIOD peripheral clock enabled during CSleep mode (default after reset)

Bit 2 **GPIOCLPEN**: GPIOC peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOC peripheral clock disabled during CSleep mode
1: GPIOC peripheral clock enabled during CSleep mode (default after reset)

Bit 1 **GPIOBLPEN**: GPIOB peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOB peripheral clock disabled during CSleep mode
1: GPIOB peripheral clock enabled during CSleep mode (default after reset)

Bit 0 **GPIOALPEN**: GPIOA peripheral clock enable during CSleep mode
Set and reset by software.
0: GPIOA peripheral clock disabled during CSleep mode
1: GPIOA peripheral clock enabled during CSleep mode (default after reset)

7.7.51 RCC APB3 Sleep Clock Register (RCC_APB3LPENR)

This register can be accessed via two different offset address.

Table 74. RCC_APB3LPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB3LPENR	0x10C	0x0000 0058
RCC_C1_APB3LPENR	0x16C	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1LPEN	Res.	Res.	LTDCLPEN	Res.	Res.	Res.
									rw			rw			

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **WWDG1LPEN**: WWDG1 Clock Enable During CSleep Mode
Set and reset by software.
0: WWDG1 clock disable during CSleep mode
1: WWDG1 clock enabled during CSleep mode (default after reset)

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **LTDCLPEN**: LTDC peripheral clock enable during CSleep mode
Provides the pixel clock (**ltdc_ker_ck**) to the LTDC block.
Set and reset by software.
0: LTDC clock disabled during CSleep mode
1: LTDC clock provided to the LTDC during CSleep mode (default after reset)

Bits 2:0 Reserved, must be kept at reset value.

7.7.52 RCC APB1 Low Sleep Clock Register (RCC_APB1LLPENR)

This register can be accessed via two different offset address.

Table 75. RCC_APB1LLPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB1LLPENR	0x110	0xE8FF CBFF
RCC_C1_APB1LLPENR	0x170	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8LPEN	UART7LPEN	DAC12LPEN	Res.	CECLPEN	Res.	Res.	Res.	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	SPDIFRXLPEN
rw	rw	rw		rw				rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LPEN	SPI2LPEN	Res.	Res.	Res.	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8LPEN**: UART8 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: UART8 peripheral clocks disabled during CSleep mode
1: UART8 peripheral clocks enabled during CSleep mode (default after reset):
The peripheral clocks of the UART8 are: the kernel clock selected by USART234578SEL and provided to **usart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 30 **UART7LPEN**: UART7 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: UART7 peripheral clocks disabled during CSleep mode
1: UART7 peripheral clocks enabled during CSleep mode (default after reset):
The peripheral clocks of the UART7 are: the kernel clock selected by USART234578SEL and provided to **usart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 29 **DAC12LPEN**: DAC1/2 peripheral clock enable during CSleep mode
Set and reset by software.
0: DAC1/2 peripheral clock disabled during CSleep mode
1: DAC1/2 peripheral clock enabled during CSleep mode (default after reset)

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECLPEN**: HDMI-CEC Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: HDMI-CEC peripheral clocks disabled during CSleep mode
1: HDMI-CEC peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the HDMI-CEC are: the kernel clock selected by CECSEL and provided to **cec_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **I2C3LPEN**: I2C3 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: I2C3 peripheral clocks disabled during CSleep mode
1: I2C3 peripheral clocks enabled during CSleep mode (default after reset):
The peripheral clocks of the I2C3 are: the kernel clock selected by I2C123SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 22 **I2C2LPEN**: I2C2 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: I2C2 peripheral clocks disabled during CSleep mode
1: I2C2 peripheral clocks enabled during CSleep mode (default after reset):
The peripheral clocks of the I2C2 are: the kernel clock selected by I2C123SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 21 **I2C1LPEN**: I2C1 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: I2C1 peripheral clocks disabled during CSleep mode
1: I2C1 peripheral clocks enabled during CSleep mode (default after reset):
The peripheral clocks of the I2C1 are: the kernel clock selected by I2C123SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 20 **UART5LPEN**: UART5 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: UART5 peripheral clocks disabled during CSleep mode
1: UART5 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the UART5 are: the kernel clock selected by USART234578SEL and provided to **uart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 19 **UART4LPEN**: UART4 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: UART4 peripheral clocks disabled during CSleep mode
1: UART4 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the UART4 are: the kernel clock selected by USART234578SEL and provided to **uart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 18 **USART3LPEN**: USART3 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: USART3 peripheral clocks disabled during CSleep mode
1: USART3 peripheral clocks enabled during CSleep mode (default after reset):
The peripheral clocks of the USART3 are: the kernel clock selected by USART234578SEL and provided to **usart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 17 **USART2LPEN**: USART2 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: USART2 peripheral clocks disabled during CSleep mode
1: USART2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the USART2 are: the kernel clock selected by USART234578SEL and provided to **usart_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 16 **SPDIFRXLPEN**: SPDIFRX Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SPDIFRX peripheral clocks disabled during CSleep mode
1: SPDIFRX peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPDIFRX are: the kernel clock selected by SPDIFSEL and provided to **spdifrx_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 15 **SPI3LPEN**: SPI3 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SPI3 peripheral clocks disabled during CSleep mode
1: SPI3 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI3 are: the kernel clock selected by I2S123SRC and provided to **spi_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 14 **SPI2LPEN**: SPI2 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SPI2 peripheral clocks disabled during CSleep mode
1: SPI2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI2 are: the kernel clock selected by I2S123SRC and provided to **spi_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: LPTIM1 peripheral clocks disabled during CSleep mode
1: LPTIM1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM1 are: the kernel clock selected by LPTIM1SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 8 **TIM14LPEN**: TIM14 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM14 peripheral clock disabled during CSleep mode
1: TIM14 peripheral clock enabled during CSleep mode (default after reset)

Bit 7 **TIM13LPEN**: TIM13 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM13 peripheral clock disabled during CSleep mode
1: TIM13 peripheral clock enabled during CSleep mode (default after reset)

Bit 6 **TIM12LPEN**: TIM12 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM12 peripheral clock disabled during CSleep mode
1: TIM12 peripheral clock enabled during CSleep mode (default after reset)

Bit 5 **TIM7LPEN**: TIM7 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM7 peripheral clock disabled during CSleep mode
1: TIM7 peripheral clock enabled during CSleep mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM6 peripheral clock disabled during CSleep mode
1: TIM6 peripheral clock enabled during CSleep mode (default after reset)

Bit 3 **TIM5LPEN**: TIM5 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM5 peripheral clock disabled during CSleep mode
1: TIM5 peripheral clock enabled during CSleep mode (default after reset)

Bit 2 **TIM4LPEN**: TIM4 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM4 peripheral clock disabled during CSleep mode
1: TIM4 peripheral clock enabled during CSleep mode (default after reset)

Bit 1 **TIM3LPEN**: TIM3 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM3 peripheral clock disabled during CSleep mode
1: TIM3 peripheral clock enabled during CSleep mode (default after reset)

Bit 0 **TIM2LPEN**: TIM2 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM2 peripheral clock disabled during CSleep mode
1: TIM2 peripheral clock enabled during CSleep mode (default after reset)

7.7.53 RCC APB1 High Sleep Clock Register (RCC_APB1HLPENR)

This register can be accessed via two different offset address.

Table 76. RCC_APB1HLPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB1HLPENR	0x114	0x0000 0136
RCC_C1_APB1HLPENR	0x174	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	MDIOSLPEN	OPAMPLPEN	Res.	SWPLPEN	CRSLPEN	Res.
							rw			rw	rw		rw	rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **FDCANLPEN**: FDCAN Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: FDCAN peripheral clocks disabled during CSleep mode
1: FDCAN peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the FDCAN are: the kernel clock selected by FDCANSEL and provided to **fdcan_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSLPEN**: MDIOS peripheral clock enable during CSleep mode
Set and reset by software.
0: MDIOS peripheral clock disabled during CSleep mode
1: MDIOS peripheral clock enabled during CSleep mode (default after reset)

Bit 4 **OPAMPLPEN**: OPAMP peripheral clock enable during CSleep mode
Set and reset by software.
0: OPAMP peripheral clock disabled during CSleep mode
1: OPAMP peripheral clock enabled during CSleep mode (default after reset)

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SWPLPEN**: SWPMI Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SWPMI peripheral clocks disabled during CSleep mode
1: SWPMI peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SWPMI are: the kernel clock selected by SWPSEL and provided to **swpmi_ker_ck** input, and the **rcc_pclk1** bus interface clock.

Bit 1 **CRSLPEN**: Clock Recovery System peripheral clock enable during CSleep mode
Set and reset by software.
0: CRS peripheral clock disabled during CSleep mode
1: CRS peripheral clock enabled during CSleep mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

7.7.54 RCC APB2 Sleep Clock Register (RCC_APB2LPENR)

This register can be accessed via two different offset address.

Table 77. RCC_APB2LPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB2LPENR	0x118	0x31D7 3033
RCC_C1_APB2LPENR	0x178	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	HRTIMLPEN	DFSDM1LPEN	Res.	Res.	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN	Res.	SPI5LPEN	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN
		rw	rw				rw	rw	rw		rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	USART6LPEN	USART1LPEN	Res.	Res.	TIM8LPEN	TIM1LPEN
		rw	rw							rw	rw			rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **HRTIMLPEN**: HRTIM peripheral clock enable during CSleep mode
Set and reset by software.
0: HRTIM peripheral clock disabled during CSleep mode
1: HRTIM peripheral clock enabled during CSleep mode (default after reset)

Bit 28 **DFSDM1LPEN**: DFSDM1 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: DFSDM1 peripheral clocks disabled during CSleep mode
1: DFSDM1 peripheral clocks enabled during CSleep mode (default after reset)
DFSDM1 peripheral clocks are: the kernel clocks selected by SAI1SEL and DFSDM1SEL and provided to **Aclk** and **clk** inputs respectively, and the **rcc_pclk2** bus interface clock.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **SAI3LPEN**: SAI3 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SAI3 peripheral clocks disabled during CSleep mode
1: SAI3 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SAI3 are: the kernel clock selected by SAI23SEL and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs, and the **rcc_pclk2** bus interface clock.

Bit 23 **SAI2LPEN**: SAI2 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SAI2 peripheral clocks disabled during CSleep mode
1: SAI2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SAI2 are: the kernel clock selected by SAI23SEL and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs, and the **rcc_pclk2** bus interface clock.

Bit 22 **SAI1LPEN**: SAI1 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SAI1 peripheral clocks disabled during CSleep mode
1: SAI1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SAI1 are: the kernel clock selected by SAI1SEL and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs, and the **rcc_pclk2** bus interface clock.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **SPI5LPEN**: SPI5 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SPI5 peripheral clocks disabled during CSleep mode
1: SPI5 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI5 are: the kernel clock selected by SPI45SEL and provided to **spi_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17LPEN**: TIM17 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM17 peripheral clock disabled during CSleep mode
1: TIM17 peripheral clock enabled during CSleep mode (default after reset)

Bit 17 **TIM16LPEN**: TIM16 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM16 peripheral clock disabled during CSleep mode
1: TIM16 peripheral clock enabled during CSleep mode (default after reset)

Bit 16 **TIM15LPEN**: TIM15 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM15 peripheral clock disabled during CSleep mode
1: TIM15 peripheral clock enabled during CSleep mode (default after reset)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SPI4LPEN**: SPI4 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SPI4 peripheral clocks disabled during CSleep mode
1: SPI4 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI4 are: the kernel clock selected by SPI45SEL and provided to **spi_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bit 12 **SPI1LPEN**: SPI1 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SPI1 peripheral clocks disabled during CSleep mode
1: SPI1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI1 are: the kernel clock selected by I2S123SRC and provided to **spi_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **USART6LPEN**: USART6 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: USART6 peripheral clocks disabled during CSleep mode
1: USART6 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the USART6 are: the kernel clock selected by USART16SEL and provided to **usart_ker_ck** input, and the **rcc_pclk2** bus interface clock.

Bit 4 **USART1LPEN**: USART1 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: USART1 peripheral clocks disabled during CSleep mode
1: USART1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the USART1 are: the kernel clock selected by USART16SEL and provided to **usart_ker_ck** inputs, and the **rcc_pclk2** bus interface clock.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8LPEN**: TIM8 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM8 peripheral clock disabled during CSleep mode
1: TIM8 peripheral clock enabled during CSleep mode (default after reset)

Bit 0 **TIM1LPEN**: TIM1 peripheral clock enable during CSleep mode
Set and reset by software.
0: TIM1 peripheral clock disabled during CSleep mode
1: TIM1 peripheral clock enabled during CSleep mode (default after reset)

7.7.55 RCC APB4 Sleep Clock Register (RCC_APB4LPENR)

This register can be accessed via two different offset address.

Table 78. RCC_APB4LPENR address offset and reset value

Register Name	Address Offset	Reset Value
RCC_APB4LPENR	0x11C	0x0421 DEAA
RCC_C1_APB4LPENR	0x17C	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	RTCAPBLPEN
										rw					rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFLPEN	COMP12LPEN	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN	Res.	I2C4LPEN	Res.	SPI6LPEN	Res.	LPUART1LPEN	Res.	SYSCFGLPEN	Res.
rw	rw		rw	rw	rw	rw		rw		rw		rw		rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **SAI4LPEN**: SAI4 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SAI4 peripheral clocks disabled during CSleep mode
1: SAI4 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SAI4 are: the kernel clocks selected by SAI4ASEL and SAI4BSEL, and provided to **sai_a_ker_ck** and **sai_b_ker_ck** inputs respectively, and the **rcc_pclk4** bus interface clock.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **RTCAPBLPEN**: RTC APB Clock Enable During CSleep Mode
Set and reset by software.
0: The register clock interface of the RTC (APB) is disabled during CSleep mode
1: The register clock interface of the RTC (APB) is enabled during CSleep mode (default after reset)

Bit 15 **VREFLPEN**: VREF peripheral clock enable during CSleep mode
Set and reset by software.
0: VREF peripheral clock disabled during CSleep mode
1: VREF peripheral clock enabled during CSleep mode (default after reset)

Bit 14 **COMP12LPEN**: COMP1/2 peripheral clock enable during CSleep mode
Set and reset by software.
0: COMP1/2 peripheral clock disabled during CSleep mode
1: COMP1/2 peripheral clock enabled during CSleep mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5LPEN**: LPTIM5 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: LPTIM5 peripheral clocks disabled during CSleep mode
1: LPTIM5 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM5 are: the kernel clock selected by LPTIM345SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 11 **LPTIM4LPEN**: LPTIM4 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: LPTIM4 peripheral clocks disabled during CSleep mode
1: LPTIM4 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM4 are: the kernel clock selected by LPTIM345SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 10 **LPTIM3LPEN**: LPTIM3 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: LPTIM3 peripheral clocks disabled during CSleep mode
1: LPTIM3 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM3 are: the kernel clock selected by LPTIM345SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 9 **LPTIM2LPEN**: LPTIM2 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: LPTIM2 peripheral clocks disabled during CSleep mode
1: LPTIM2 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPTIM5 are: the kernel clock selected by LPTIM2SEL and provided to **lptim_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4LPEN**: I2C4 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: I2C4 peripheral clocks disabled during CSleep mode
1: I2C4 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the I2C4 are: the kernel clock selected by I2C4SEL and provided to **i2c_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6LPEN**: SPI6 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: SPI6 peripheral clocks disabled during CSleep mode
1: SPI6 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the SPI6 are: the kernel clock selected by SPI6SEL and provided to **spi_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1LPEN**: LPUART1 Peripheral Clocks Enable During CSleep Mode
Set and reset by software.
0: LPUART1 peripheral clocks disabled during CSleep mode
1: LPUART1 peripheral clocks enabled during CSleep mode (default after reset)
The peripheral clocks of the LPUART1 are: the kernel clock selected by LPUART1SEL and provided to **lpuart_ker_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGLPEN**: SYSCFG peripheral clock enable during CSleep mode
Set and reset by software.
0: SYSCFG peripheral clock disabled during CSleep mode
1: SYSCFG peripheral clock enabled during CSleep mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

7.8 RCC register map

Table 79. RCC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**RCC_CR**	Res.	Res.	PLL3RDY	PLL3ON	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	Res.	HSECSSON	HSEBYP	HSERDY	HSEON	D2CKRDY	D1CKRDY	HSI48RDY	HSI48ON	Res.	Res.	CSIKERON	CSIRDY	CSION	Res.	HSIDIVF	HSIDIV[1:0]		HSIRDY	HSIKERON	HSION
	Reset value			0	0	0	0	0	0					0	0	0	0	0	0	0	0			0	0	0		0	0	0	0	0	1
0x004	**RCC_ICSCR**	Res.	CSITRIM[4:0]					CSICAL[7:0]								HSITRIM[5:0]						HSICAL[11:0]											
	Reset value		1	0	0	0	0	-	-	-	-	-	-	-	-	1	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-
0x008	**RCC_CRRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
	Reset value																							-	-	-	-	-	-	-	-	-	-
0x00C	**reserved**	Reserved																															
0x010	**RCC_CFGR**	MCO2[2:0]			MCO2PRE[3:0]				MCO1[2:0]			MCO1PRE[3:0]				Res.	Res.	TIMPRE	HRTIMSEL	RTCPRE[5:0]						STOPKERWUCK	STOPWUCK	SWS[2:0]			SW[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**reserved**	Reserved																															
0x018	**RCC_D1CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D1CPRE[3:0]				Res.	D1PPRE[2:0]			HPRE[3:0]			
	Reset value																					0	0	0	0		0	0	0	0	0	0	0
0x01C	**RCC_D2CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D2PPRE2[2:0]			Res.	D2PPRE1[2:0]			Res.	Res.	Res.	Res.
	Reset value																						0	0	0		0	0	0				
0x020	**RCC_D3CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D3PPRE[2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x024	**reserved**	Reserved																															
0x028	**RCC_ PLLCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	DIVM3[5:0]						Res.	Res.	DIVM2[5:0]						Res.	Res.	DIVM1[5:0]						Res.	Res.	PLLSRC[1:0]	
	Reset value							1	0	0	0	0	0			1	0	0	0	0	0			1	0	0	0	0	0			0	0

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**RCC_PLLCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVR3EN	DIVQ3EN	DIVP3EN	DIVR2EN	DIVQ2EN	DIVP2EN	DIVR1EN	DIVQ1EN	DIVP1EN	Res.	Res.	Res.	Res.	PLL3RGE[1:0]		PLL3VCOSEL	PLL3FRACEN	PLL2RGE[1:0]		PLL2VCOSEL	PLL2FRACEN	PLL1RGE[1:0]		PLL1VCOSEL	PLL1FRACEN
	Reset value								1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x030	**RCC_PLL1DIVR**	Res.	DIVR1[6:0]							Res.	DIVQ1[6:0]							DIVP1[6:0]							DIVN1[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x034	**RCC_PLL1FRACR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACN1[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x038	**RCC_PLL2DIVR**	Res.	DIVR2[6:0]							Res.	DIVQ2[6:0]							DIVP2[6:0]							DIVN2[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x03C	**RCC_PLL2FRACR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACN2[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x040	**RCC_PLL3DIVR**	Res.	DIVR3[6:0]							Res.	DIVQ3[6:0]							DIVP3[6:0]							DIVN3[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x044	**RCC_PLL3FRACR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACN3[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x048	**reserved**	Reserved																															
0x04C	**RCC_D1CCIPR**	Res.	Res.	CKPERSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPISEL[1:0]		Res.	Res.	FMCSEL[1:0]	
	Reset value			0	0												0											0	0			0	0
0x050	**RCC_D2CCIP1R**	SWPSEL	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	DFSDM1SEL	Res.	Res.	SPDIFSEL[1:0]		Res.	SPI45SEL[2:0]			Res.	SPI123SEL[2:0]			Res.	Res.	Res.	SAI23SEL[2:0]			Res.	Res.	Res.	SAI1SEL[2:0]		
	Reset value	0		0	0				0			0	0		0	0	0		0	0	0				0	0	0				0	0	0
0x054	**RCC_D2CCIP2R**	Res.	LPTIM1SEL[2:0]			Res.	Res.	Res.	Res.	CECSEL[1:0]		USBSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	I2C123SEL[1:0]		Res.	Res.	RNGSEL[1:0]		Res.	Res.	USART16SEL[2:0]			USART234578SEL[2:0]		
	Reset value		0	0	0					0	0	0	0							0	0			0	0			0	0	0	0	0	0

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	RCC_D3CCIPR	Res.	SPI6SEL[2:0]			Res.	SAI4BSEL[2:0]			SAI4ASEL[2:0]			Res.	Res.	Res.	ADCSEL[1;0]		LPTIM345SEL[2:0]			LPTIM2SEL[2:0]			I2C4SEL[1:0]		Res.	Res.	Res.	Res.	Res.	LPUART1SEL[2:0]		
	Reset value		0	0	0		0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0						0	0	0
0x05C	reserved	Reserved																															
0x060	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSIE	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x064	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x068	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	CSIRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x06C	reserved	Reserved																															
0x070	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	Reset value																0	0						0	0		0	0	0	0	0	0	0
0x074	RCC_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value																															0	0
0x078	reserved	Reserved																															
0x07C	RCC_ AHB3RSTR	CPURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1RST	Res.	QSPIRST	Res.	FMCRST	Res.	Res.	Res.	Res.	Res.	Res.	JPGDECRST	DMA2DRST	Res.	Res.	Res.	MDMARST
	Reset value	0															0		0		0							0	0				0
0x080	RCC_ AHB1RSTR	Res.	Res.	Res.	Res.	USB2OTGRST	Res.	USB1OTGRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1MACRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12RST	Res.	Res.	DMA2RST	DMA1RST
	Reset value					0		0										0										0				0	0

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084	**RCC_ AHB2RSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2RST	Res.	Res.	RNGRST	HASHRST	CRYPTRST	Res.	Res.	Res.	CAMITFRST
	Reset value																							0			0	0	0				0
0x088	**RCC_ AHB4RSTR**	Res.	Res.	Res.	Res.	Res.	Res.	HSEMRST	ADC3RST	Res.	Res.	BDMARST	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value							0	0			0		0								0	0	0	0	0	0	0	0	0	0	0
0x08C	**RCC_ APB3RSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCRST	Res.	Res.	Res.
	Reset value																													0			
0x090	**RCC_ APB1LRSTR**	UART8RST	UART7RST	DAC12RST	Res.	HDMICECRST	Res.	Res.	Res.	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	SPDIFRXRST	SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value	0	0	0		0				0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0
0x094	**RCC_ APB1HRSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	MDIOSRST	OPAMPRST	Res.	RST	CRSRST	Res.
	Reset value																								0			0	0		0	0	
0x098	**RCC_ APB2RSTR**	Res.	Res.	HRTIMRST	DFSDM1RST	Res.	Res.	Res.	SAI3RST	SAI2RST	SAI1RST	Res.	SPI5RST	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	Res.	SPI4RST	SPI1RST	Res.	Res.	Res.	Res.	Res.	Res.	USART6RST	USART1RST	Res.	Res.	TIM8RST	TIM1RST
	Reset value			0	0				0	0	0		0		0	0	0			0	0							0	0			0	0
0x09C	**RCC_ APB4RSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4RST	Res.	Res.	Res.	Res.	Res.	VREFRST	COMP12RST	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST	Res.	Res.	I2C4RST	Res.	SPI6RST	Res.	LPUART1RST	Res.	SYSCFGRST	Res.
	Reset value											0						0	0		0	0	0	0			0		0		0		0	
0x0A0	**RCC_GCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WW1RSC
	Reset value																																0
0x0A4	**reserved**	Reserved																															

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A8	**RCC_D3AMR**	Res.	Res.	SRAM4AMEN	BKPRAMAMEN	Res.	Res.	Res.	ADC3AMEN	Res.	Res.	SAI4AMEN	Res.	CRCAMEN	Res.	Res.	RTCAMEN	VREFAMEN	COMP12AMEN	Res.	LPTIM5AMEN	LPTIM4AMEN	LPTIM3AMEN	LPTIM2AMEN	Res.	I2C4AMEN	Res.	SPI6AMEN	Res.	LPUART1AMEN	Res.	Res.	BDMAAMEN
	Reset value			0	0				0			0		0			0	0	0		0	0	0	0		0		0		0			0
0x0AC to 0x0CC	**reserved**	Reserved																															
0x0D0	**RCC_RSR**	Res.	LPWRRSTF	Res.	WWDG1RSTF	Res.	IWDG1RSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	D2RSTF	D1RSTF	Res.	CPURSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0		0		0		0	1	1	1	1	1		1	0																
0x0D4	**RCC_ AHB3ENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN	Res.	QSPIEN	Res.	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	JPGDECEN	DMA2DEN	Res.	Res.	Res.	MDMAEN
	Reset value																0		0		0							0	0				0
0x0D8	**RCC_ AHB1ENR**	Res.	Res.	Res.	Res.	USB2OTGHSEN	USB1OTGHSULPIEN	USB1OTGHSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXEN	ETH1TXEN	ETH1MACEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12EN	Res.	Res.	DMA2EN	DMA1EN
	Reset value					0	0	0								0	0	0										0				0	0
0x0DC	**RCC_ AHB2ENR**	SRAM3EN	SRAM2EN	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2EN	Res.	Res.	RNGEN	HASHEN	CRYPTEN	Res.	Res.	Res.	DCMIEN
	Reset value	0	0	0																			0			0	0	0				0
0x0E0	**RCC_ AHB4ENR**	Res.	Res.	Res.	BKPRAMEN	Res.	Res.	HSEMEN	ADC3EN	Res.	Res.	BDMAEN	Res.	CRCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value				0			0	0			0		0								0	0	0	0	0	0	0	0	0	0	0	0
0x0E4	**RCC_ APB3ENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1EN	Res.	Res.	LTDCEN	Res.	Res.	Res.
	Reset value																										0			0			

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0E8	**RCC_ APB1LENR**	UART8EN	UART7EN	DAC12EN	Res.	HDMICECEN	Res.	Res.	Res.	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	SPDIFRXEN	SPI3EN	SPI2EN	Res.	Res.	Res.	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0	0		0				0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0
0x0EC	**RCC_ APB1HENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	MDIOSEN	OPAMPEN	Res.	SWPEN	CRSEN	Res.
	Reset value																								0			0	0		0	0	
0x0F0	**RCC_ APB2ENR**	Res.	Res.	HRTIMEN	DFSDM1EN	Res.	Res.	Res.	SAI3EN	SAI2EN	SAI1EN	Res.	SPI5EN	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	Res.	SPI4EN	SPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	USART6EN	USART1EN	Res.	Res.	TIM8EN	TIM1EN
	Reset value			0	0				0	0	0		0		0	0	0			0	0							0	0			0	0
0x0F4	**RCC_ APB4ENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	RTCAPBEN	VREFEN	COMP12EN	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN	Res.	I2C4EN	Reserved	SPI6EN	Res.	LPUART1EN	Res.	SYSCFGEN	Res.
	Reset value											0					1	0	0		0	0	0	0		0	0	0		0		0	
0x0F8	**reserved**	Reserved																															
0x0FC	**RCC_ AHB3LPENR**	AXISRAMLPEN	ITCMLPEN	DTCM2LPEN	DTCM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1LPEN	Res.	QSPILPEN	Res.	FMCLPEN	Res.	Res.	Res.	FLASHLPEN	Res.	Res.	JPGDECLPEN	DMA2DLPEN	Res.	Res.	Res.	MDMALPEN
	Reset value	1	1	1	1												1		1		1				1			1	1				1
0x100	**RCC_ AHB1LPENR**	Res.	Res.	Res.	Res.	USB2OTGHSLPEN	USB1OTGHSULPILPEN	USB1OTGHSLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXLPEN	ETH1TXLPEN	ETH1MACLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPEN	Res.	Res.	Res.	DMA2LPEN	DMA1LPEN
	Reset value					1	1	1								1	1	1										1				1	1
0x104	**RCC_ AHB2LPENR**	SRAM3LPEN	SRAM2LPEN	SRAM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2LPEN	Res.	Res.	RNGLPEN	HASHLPEN	CRYPTLPEN	Res.	Res.	Res.	CAMITFLPEN
	Reset value	1	1	1																			1			1	1	1				1

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x108	**RCC_ AHB4LPENR**	Res.	Res.	SRAM4LPEN	BKPRAMLPEN	Res.	Res.	Res.	ADC3LPEN	Res.	Res.	BDMALPEN	Res.	CRCLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value			1	1				1			1		1									1	1	1	1	1	1	1	1	1	1	1
0x10C	**RCC_ APB3LPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1LPEN	Res.	Res.	LTDCLPEN	Res.	Res.	Res.
	Reset value																										1			1			
0x110	**RCC_ APB1LLPENR**	UART8LPEN	UART7LPEN	DAC12LPEN	Res.	CECLPEN	Res.	Res.	Res.	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	SPDIFRXLPEN	SPI3LPEN	SPI2LPEN	Res.	Res.	Res.	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1	1	1		1				1	1	1	1	1	1	1	1	1	1					1	1	1	1	1	1	1	1	1	1
0x114	**RCC_ APB1HLPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	MDIOSLPEN	OPAMPLPEN	Res.	SWPLPEN	CRSLPEN	Res.
	Reset value																								1			1	1		1	1	
0x118	**RCC_ APB2LPENR**	Res.	Res.	HRTIMLPEN	DFSDM1LPEN	Res.	Res.	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN	Res.	SPI5LPEN	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	Res.	Res.	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	USART6LPEN	USART1LPEN	Res.	Res.	TIM8LPEN	TIM1LPEN
	Reset value			1	1				1	1	1		1		1	1	1			1	1							1	1			1	1
0x11C	**RCC_ APB4LPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	RTCAPBLPEN	VREFLPEN	COMP12LPEN	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN	Res.	I2C4LPEN	Res.	SPI6LPEN	Res.	LPUART1LPEN	Res.	SYSCFGLPEN	Res.
	Reset value											1					1	1	1		1	1	1	1		1		1		1		1	
0x120 to 0x130	**reserved**	Reserved																															
0x134	**RCC_C1_ AHB3ENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN	Res.	QSPIEN	Res.	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	JPGDECEN	DMA2DEN	Res.	Res.	Res.	MDMAEN
	Reset value																0		0		0							0	0				0

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x138	**RCC_C1_ AHB1ENR**	Res.	Res.	Res.	Res.	USB2OTGHSEN	USB1OTGHSULPIEN	USB1OTGHSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXEN	ETH1TXEN	ETH1MACEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12EN	Res.	Res.	Res.	DMA2EN	DMA1EN
	Reset value					0	0	0								0	0	0										0				0	0
0x13C	**RCC_C1_ AHB2ENR**	SRAM3EN	SRAM2EN	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2EN	Res.	Res.	RNGEN	HASHEN	CRYPTEN	Res.	Res.	Res.	DCMIEN
	Reset value	0	0	0																				0			0	0	0				0
0x140	**RCC_C1_ AHB4ENR**	Res.	Res.	Res.	BKPRAMEN	Res.	Res.	HSEMEN	ADC3EN	Res.	Res.	DMA1EN	Res.	CRCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value				0			0	0			0		0									0	0	0	0	0	0	0	0	0	0	0
0x144	**RCC_C1_ APB3ENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1EN	Res.	Res.	LTDCEN	Res.	Res.	Res.
	Reset value																										0			0			
0x148	**RCC_C1_ APB1LENR**	UART8EN	UART7EN	DAC12EN	Res.	HDMICECEN	Res.	Res.	Res.	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	SPDIFRXEN	SPI3EN	SPI2EN	Res.	Res.	Res.	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0	0		0				0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0
0x14C	**RCC_C1_ APB1HENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	MDIOSEN	OPAMPEN	Res.	SWPEN	CRSEN	Res.
	Reset value																								0			0	0		0	0	
0x150	**RCC_C1_ APB2ENR**	Res.	Res.	HRTIMEN	DFSDM1EN	Res.	Res.	Res.	SAI3EN	SAI2EN	SAI1EN	Res.	SPI5EN	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	Res.	SPI4EN	SPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	USART6EN	USART1EN	Res.	Res.	Res.	TIM8EN	TIM1EN
	Reset value			0	0				0	0	0		0		0	0	0			0	0							0	0				0	0
0x154	**RCC_C1_ APB4ENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	RTCAPBEN	VREFEN	COMP12EN	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN	Res.	Res.	I2C4EN	Res.	SPI6EN	Res.	LPUART1EN	Res.	SYSCFGEN	Res.
	Reset value											0					1	0	0		0	0	0	0			0		0		0		0	
0x158	**reserved**	Reserved																															

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x15C	**RCC_C1_ AHB3LPENR**	AXISRAMLPEN	ITCMLPEN	DTCM2LPEN	DTCM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1LPEN	Res.	QSPILPEN	Res.	FMCLPEN	Res.	Res.	Res.	FLASHLPEN	Res.	Res.	JPGDECLPEN	DMA2DLPEN	Res.	Res.	Res.	MDMALPEN
	Reset value	1	1	1	1												1		1		1				1			1	1				1
0x160	**RCC_C1_ AHB1LPENR**	Res.	Res.	Res.	Res.	USB2OTGHSLPEN	USB1OTGHSULPILPEN	USB1OTGHSLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXLPEN	ETH1TXLPEN	ETH1MACLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPEN	Res.	Res.	DMA2LPEN	DMA1LPEN
	Reset value					1	1	1								1	1	1										1				1	1
0x164	**RCC_C1_ AHB2LPENR**	SRAM3LPEN	SRAM2LPEN	SRAM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2LPEN	Res.	Res.	RNGLPEN	HASHLPEN	CRYPTLPEN	Res.	Res.	Res.	CAMITFLPEN
	Reset value	1	1	1																			1			1	1	1				1
0x168	**RCC_C1_ AHB4LPENR**	Res.	Res.	SRAM4LPEN	BKPRAMLPEN	Res.	Res.	Res.	ADC3LPEN	Res.	Res.	DMA1LPEN	Res.	CRCLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value			1	1				1			1		1									1	1	1	1	1	1	1	1	1	1	1
0x16C	**RCC_C1_ APB3LPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1LPEN	Res.	Res.	LTDCLPEN	Res.	Res.	Res.
	Reset value																										1			1			
0x170	**RCC_C1_ APB1LLPENR**	UART8LPEN	UART7LPEN	DAC12LPEN	Res.	CECLPEN	Res.	Res.	Res.	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	SPDIFRXLPEN	SPI3LPEN	SPI2LPEN	Res.	Res.	Res.	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1	1	1		1				1	1	1	1	1	1	1	1	1	1					1	1	1	1	1	1	1	1	1	1
0x174	**RCC_C1_ APB1HLPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	MDIOSLPEN	OPAMPLPEN	Res.	SWPLPEN	CRSLPEN	Res.
	Reset value																								1			1	1		1	1	

Table 79. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x178	**RCC_C1_ APB2LPENR**	Res.	Res.	HRTIMLPEN	DFSDM1LPEN	Res.	Res.	Res.	SA3LPEN	SAI2LPEN	SAI1LPEN	Res.	SPI5LPEN	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	Res.	Res.	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	USART6LPEN	USART1LPEN	Res.	Res.	TIM8LPEN	TIM1LPEN
	Reset value			1	1				1	1	1		1		1	1	1			1	1							1	1			1	1
0x17C	**RCC_C1_ APB4LPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	RTCAPBLPEN	VREFLPEN	COMP12LPEN	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN	Res.	I2C4LPEN	Res.	SPI6LPEN	Res.	LPUART1LPEN	Res.	SYSCFGLPEN	Res.
	Reset value											1					1	1	1		1	1	1	1		1		1		1		1	
0x180 to 0x1FC	**reserved**	Reserved																															

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

8 Clock recovery system (CRS)

8.1 Introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means for oscillator output frequency evaluation, based on comparison with a selectable synchronization signal. It is capable of doing automatic adjustment of oscillator trimming based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In such case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, which is sent by a USB host at precise 1-ms intervals.

The synchronization signal can also be derived from the LSE oscillator output or it can be generated by user software.

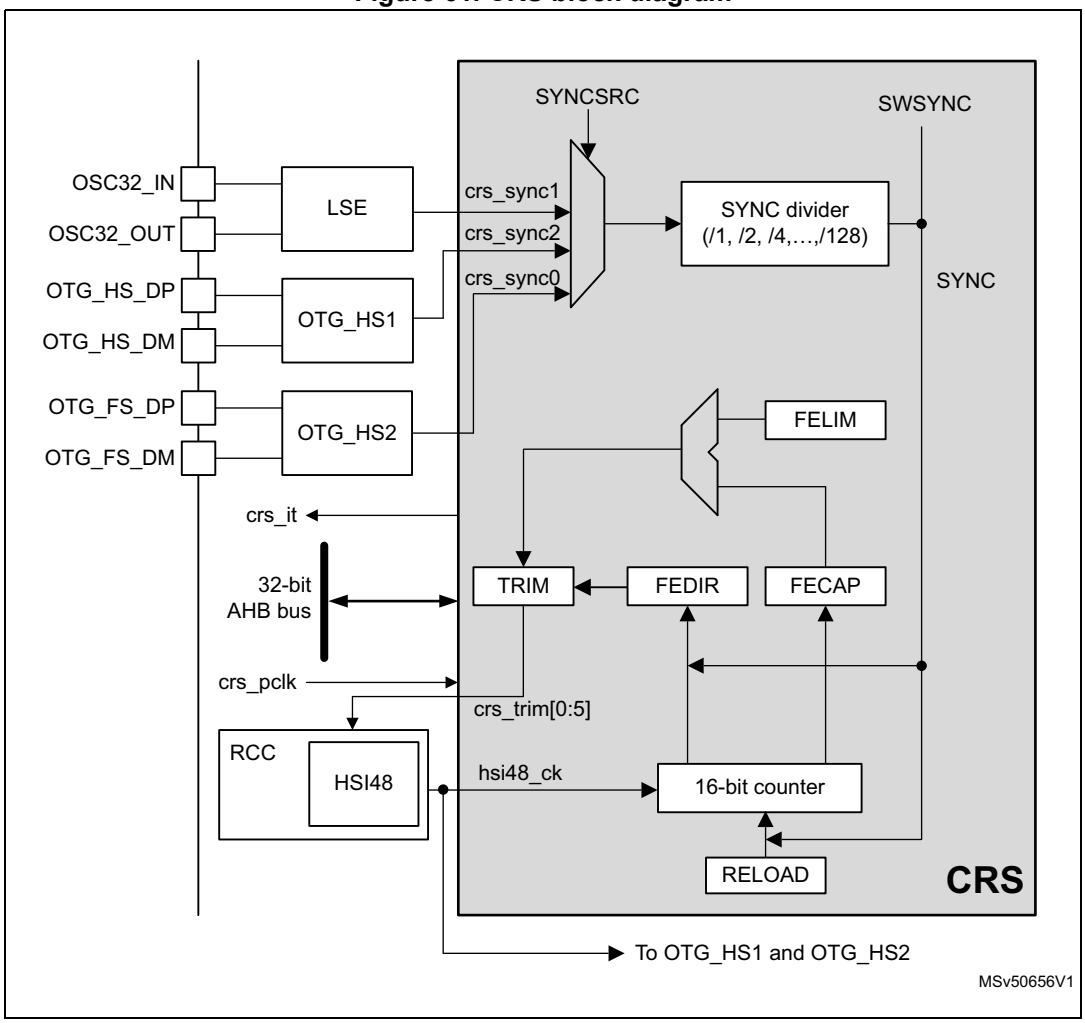
8.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - USB2 SOF packet reception
  - LSE oscillator output
  - USB1 SOF packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster start-up convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

8.3 CRS functional description

8.3.1 CRS block diagram

Figure 61. CRS block diagram



8.4 CRS internal signals

Table 80 gives the list of CRS internal signals.

Table 80. CRS internal input/output signals

Signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
hsi48_ck	Digital input	HSI48 oscillator clock

Table 80. CRS internal input/output signals (continued)

Signal name	Signal type	Description
crs_trim[0:5]	Digital output	HSI48 oscillator smooth trimming value
crs_sync0 crs_sync1 crs_sync2	Digital input	SYNC signal source selection (OTG_HS1, LSE or OTG_HS2)

8.4.1 Synchronization input

The CRS synchronization (SYNC) source, selectable through the CRS_CFGR register, can be the signal from an external signal (SYNC), the LSE clock, the OTG HS1 SOF signal, or the OTG HS2 SOF signal. This source signal also has a configurable polarity and can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to *Section 8.7.2: CRS configuration register (CRS_CFGR)*.

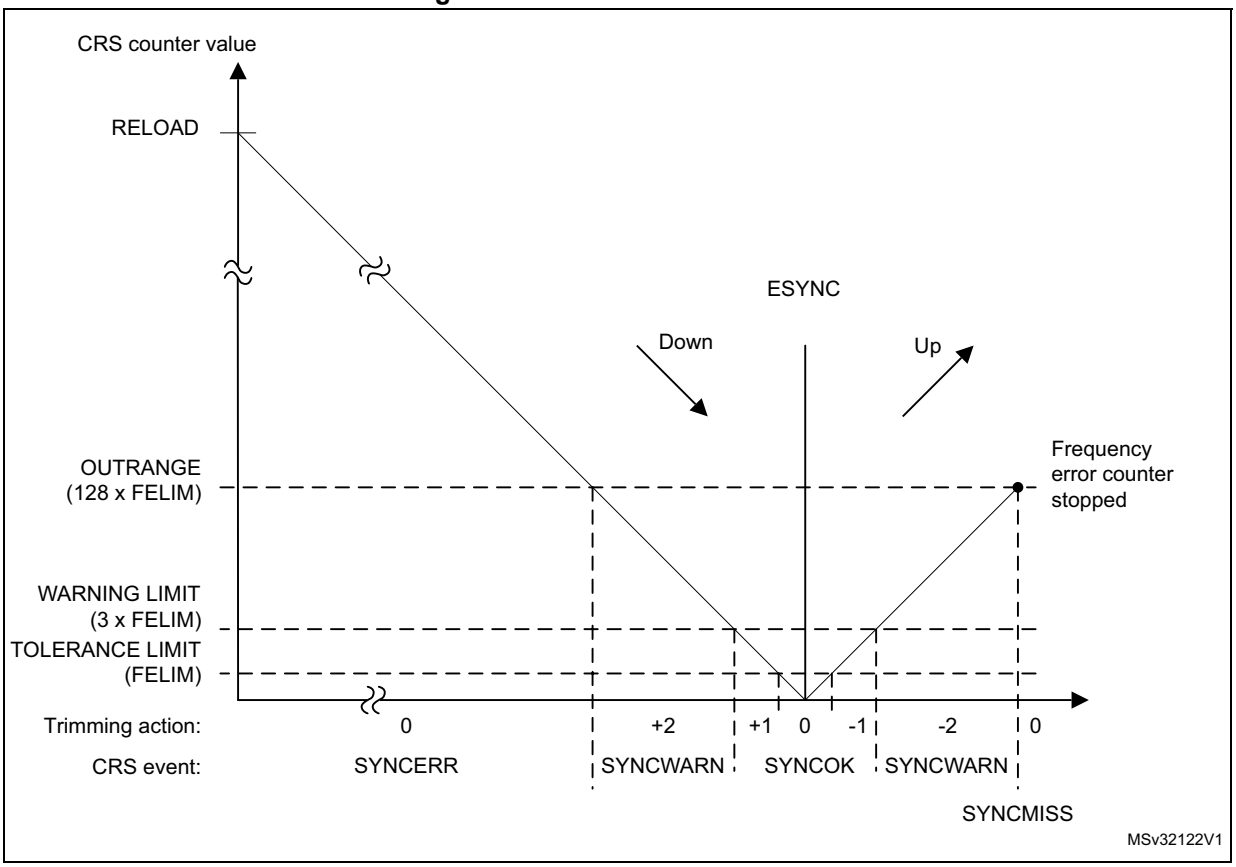
It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

8.4.2 Frequency error measurement

The frequency error counter is a 16-bit down/up counter which is reloaded with the RELOAD value on each SYNC event. It starts counting down till it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit where it eventually stops (if no SYNC event is received) and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the downcounting phase (before reaching the zero value), it means that the actual frequency is lower than the target (and so, that the TRIM value should be incremented), while when it is detected during the upcounting phase it means that the actual frequency is higher (and that the TRIM value should be decremented).

Figure 62. CRS counter behavior



8.4.3 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 * FELIM value
- OUTRANGE (error limit), defined as 128 * FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one and that then, no trimming action is necessary.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated
  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, it means that a stronger trimming action is necessary, and there is a risk that the optimal TRIM value will not be reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, it means that the frequency is out of the trimming range. This can also happen when the SYNC input is not clean or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is so close to its limits that the automatic trimming would force it to overflow or underflow, then the TRIM value is set just to the limit and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

8.4.4 CRS initialization and configuration

RELOAD value

The RELOAD value should be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. It is then decreased by one in order to reach the expected synchronization on the zero value. The formula is the following:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of HSI48 oscillator clock ticks. The following formula can be used:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result should be always rounded up to the nearest integer value in order to obtain the best trimming response. If frequent trimming actions are not wanted in the application, the trimming hysteresis can be increased by increasing slightly the FELIM value.

The reset value of the FELIM field corresponds to (f_{TARGET} / f_{SYNC}) = 48000 and to a typical trimming step size of 0.14%.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields which can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

8.5 CRS low-power modes

Table 81. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop or Standby mode is exited and the HSI48 oscillator restarted.
Standby	

8.6 CRS interrupts

Table 82. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

8.7 CRS registers

Refer to *Section 1.2 on page 97* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

8.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERRIE	SYNC WARNIE	SYNC OKIE
		rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI48 oscillator.

The default value is 32, which corresponds to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 8.4.3: Frequency error evaluation and automatic trimming* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable
0: Expected SYNC (ESYNCF) interrupt disabled
1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable
0: Synchronization or trimming error (ERRF) interrupt disabled
1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable
0: SYNC warning (SYNCWARNF) interrupt disabled
1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable
0: SYNC event OK (SYNCOKF) interrupt disabled
1: SYNC event OK (SYNCOKF) interrupt enabled

8.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]							
rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SYNCPOL**: SYNC polarity selection
This bit is set and cleared by software to select the input polarity for the SYNC signal source.
0: SYNC active on rising edge (default)
1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection
These bits are set and cleared by software to select the SYNC signal source.
00: USB2 SOF selected as SYNC signal source
01: LSE selected as SYNC signal source
10: OTG HS1 SOF selected as SYNC signal source (default)
11: Reserved
When using USB LPM (Link Power Management) and the device is in Sleep mode, the periodic USB SOF will not be generated by the host. No SYNC signal will therefore be provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin should be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.
000: SYNC not divided (default)
001: SYNC divided by 2
010: SYNC divided by 4
011: SYNC divided by 8
100: SYNC divided by 16
101: SYNC divided by 32
110: SYNC divided by 64
111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 8.4.3: Frequency error evaluation and automatic trimming* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 8.4.2: Frequency error measurement* for more details about counter behavior.

8.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 8.4.3: Frequency error evaluation and automatic trimming* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.
0: Upcounting direction, the actual frequency is above the target.
1: Downcounting direction, the actual frequency is below the target.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
0: No trimming error signalized
1: Trimming error signalized

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reached value FELIM * 128 and no SYNC was detected, meaning either that a SYNC pulse was missed or that the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, and that some other action should be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC) and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signalized

1: SYNC missed error signalized

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action should be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signalized

1: SYNC error signalized

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signalized

1: Expected SYNC signalized

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signalized

1: Synchronization or trimming error signalized

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signalized

1: SYNC warning signalized

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signalized

1: SYNC event OK signalized

8.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNC WARNC	SYNC OKC
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

8.7.5 CRS register map

Table 83. CRS register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CRS_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value																			1	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**CRS_CFGR**	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	**CRS_ISR**	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	**CRS_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

9 Hardware semaphore (HSEM)

9.1 Hardware semaphore introduction

The hardware semaphore block provides 32 (32-bit) register based semaphores.

The semaphores can be used to ensure synchronization between different processes running on the core. The HSEM provides a non blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Locking a semaphore can be done in two ways:
  - 2-step lock: by writing MasterID and ProcessID to the semaphore, followed by a Read check
  - 1-step lock: by reading the MasterID from the semaphore
- Interrupt generation when a semaphore is freed
  - Each semaphore may generate an interrupt
- Semaphore clear protection
  - A semaphore will only be cleared when MasterID and ProcessID match
- Global semaphore clear per MasterID

9.2 Hardware semaphore main features

The HSEM includes the following features:

- 32 (32-bit) semaphores
- 8-bit ProcessID
- 4-bit MasterID
- 1 interrupt line
- Lock indication

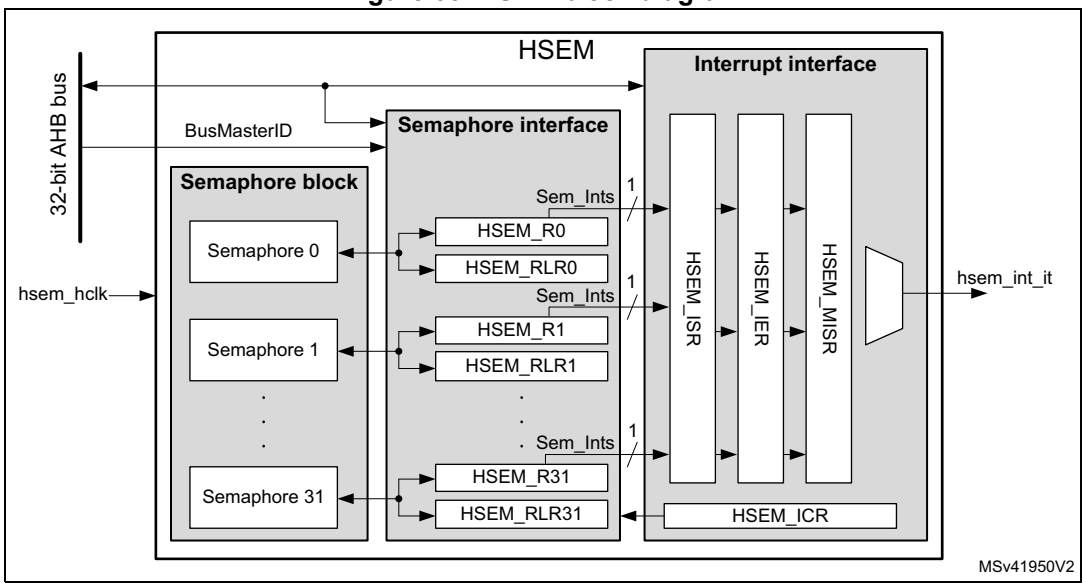
9.3 HSEM functional description

9.3.1 HSEM block diagram

As shown in *Figure 63*, the HSEM is based on three sub-blocks:

- The Semaphore block containing the semaphore status and IDs
- The Semaphore Interface block providing AHB access to the Semaphore via the HSEM_R and HSEM_RLR registers
- The Interrupt interface block providing control for the interrupts via the HSEM_CnISR, HSEM_CnIER, HSEM_CnMISR, and HSEM_CnICR registers

Figure 63. HSEM block diagram



9.3.2 HSEM internal signals

Table 84. HSEM internal input/output signals

Signal name	Signal type	Description
hsem_hclk	Digital input	AHB clock
hsem_int_it	Digital output	Interrupt line

9.3.3 HSEM lock procedures

There are two lock procedures:

- 2-step (Write) lock
- 1-step (Read) lock

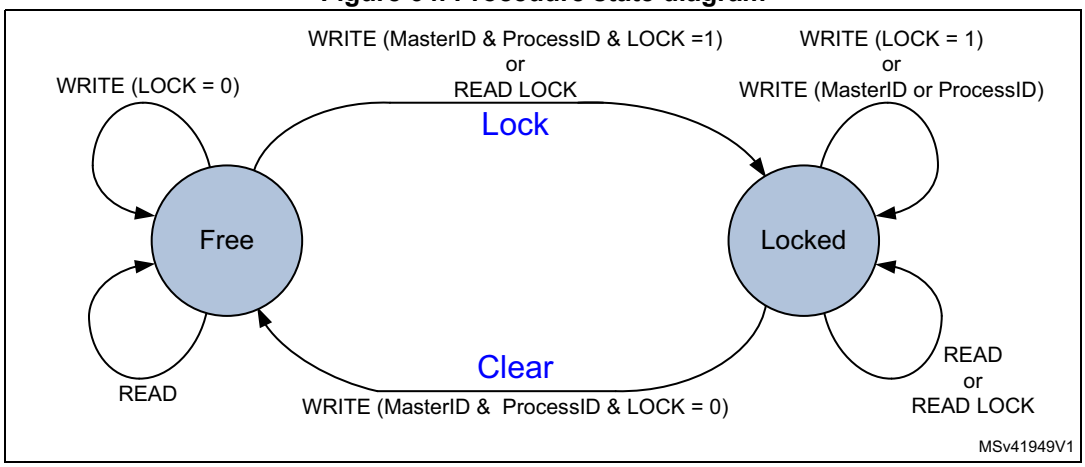
The semaphore is free when its lock bit is ‘0’, in this case the MasterID and ProcessID are also ‘0. When the lock bit is ‘1’ the semaphore is locked and the MasterID indicates which AHB bus master has locked it. The ProcessID indicates which process of that AHB bus master has locked the semaphore.

When Write locking a semaphore, the MasterID is taken from the bus master ID and the ProcessID is taken from the Write data. When Read locking the semaphore, the MasterID is taken from the master ID, and the ProcessID will be zero. There are no ProcessID available with the 1-step (Read) lock.

The MasterID is taken from the AHB bus master ID. The ProcessID is written by the firmware of that AHB bus master. Each AHB bus master process must have a unique ProcessID. ProcessID is only available in the 2-step lock procedure.

The two procedures (1-step and 2-step) can be used concurrently.

Figure 64. Procedure state diagram



2-step (Write) lock procedure

The 2-step lock procedure consists in a Write to lock the semaphore, followed by a Read to check if the lock has been successful, carried out from the HSEM_Rn register

- Write semaphore with ProcessID and MasterID, and LOCK bit = 1
  (Lock will be put in place when semaphore is free at Write time)
- Read-back the semaphore
  (FW checks lock status, if ProcessID and MasterID match, then lock is confirmed).
- Else retry (the semaphore has been locked by another AHB bus master or process)

A semaphore can only be locked when it is free.

A semaphore can be locked when the ProcessID is ‘0’.

Consecutive write attempts with the lock bit = 1 to a locked semaphore are ignored.

1-step (Read) lock procedure

The 1-step procedure consists in a Read to lock and check the semaphore in a single step from the HSEM_RLRn register.

- Read Lock semaphore with MasterID.
- If Read MasterID matches and ProcessID = 0, then lock is put in place.
  (if MasterID matches and ProcessID is not ‘0’, this means that another process from the same MasterID has locked the semaphore with a 2-step (Write) procedure.
- Else retry (the semaphore has been locked by another AHB bus master or process)

A semaphore can only be locked when it is free. When Read locking a free semaphore the ProcessID will be ‘0’. Read locking a locked semaphore will return the MasterID and ProcessID that locked it. All Read locks, including the first one which locks the semaphore, will return the MasterID that locks or has locked the semaphore.

If multiple processes of the same AHB bus master use the 1-step procedure, all processes using the same semaphore will read the same status. When only one process locks the semaphore, each process of that AHB bus master will read the semaphore as locked by itself with the MasterID.

9.3.4 HSEM Write/Read/ReadLock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 0x80.

In the first register address bank the semaphore can be written (locked/cleared) and read through the HSEM_R registers.

In the second register address bank the semaphore can be read (locked) through the HSEM_RLR registers.

9.3.5 HSEM Clear procedures

Clearing a semaphore is a protected process, to prevent accidental clearing by a AHB bus master or by a process that does not have the semaphore lock right. The semaphore Clear procedure consists in writing to the semaphore with the corresponding MasterID and ProcessID and the lock bit = 0. When cleared, the semaphore lock bit, the MasterID, and the ProcessID are all ‘0’.

When cleared, an interrupt may be generated to signal the event. To this end, the semaphore interrupt shall be enabled.

The Clear procedure consists in a Write to the semaphore HSEM_R register

- Write semaphore with ProcessID and MasterID, lock bit = 0
- If ProcessID and MasterID match, semaphore is freed, and an interrupt may be generated when enabled
- Else Write is ignored, semaphore remains locked and no interrupts are generated (the semaphore is locked by another AHB bus master or process)

If multiple processes of the same AHB bus master use the 1-step lock procedure (ProcessID = 0), all processes using the same semaphore will clear the semaphore also for the other processes of that AHB bus master.

9.3.6 HSEM MasterID semaphore clear

All semaphores locked by a AHB bus master can be cleared all at once by using the HSEM_CR register.

The procedure to clear all semaphores locked by a AHB bus master is the following:

- Write MasterID and correct KEY value. All locked semaphore with a matching MasterID are cleared (set to free), and may generate an interrupt when enabled.

This procedure may be used in case of an incorrect functioning AHB bus master, where the AHB bus master can free the locked semaphores by writing the MasterID into the HSEM_CR register with the correct KEY value. This will clear all locked semaphores with a matching MasterID.

An interrupt may be generated for the semaphore(s) that become free. To this end, the semaphore interrupt shall be enabled in the HSEM_CnIER registers.

9.3.7 HSEM interrupts

hsem_int_it interrupt line allows each of the 32 semaphores to generate an interrupt.

This interrupt line provides the following features:

- Interrupt enable per semaphore
- Interrupt clear per semaphore
- Interrupt status per semaphore
- Masked interrupt status per semaphore

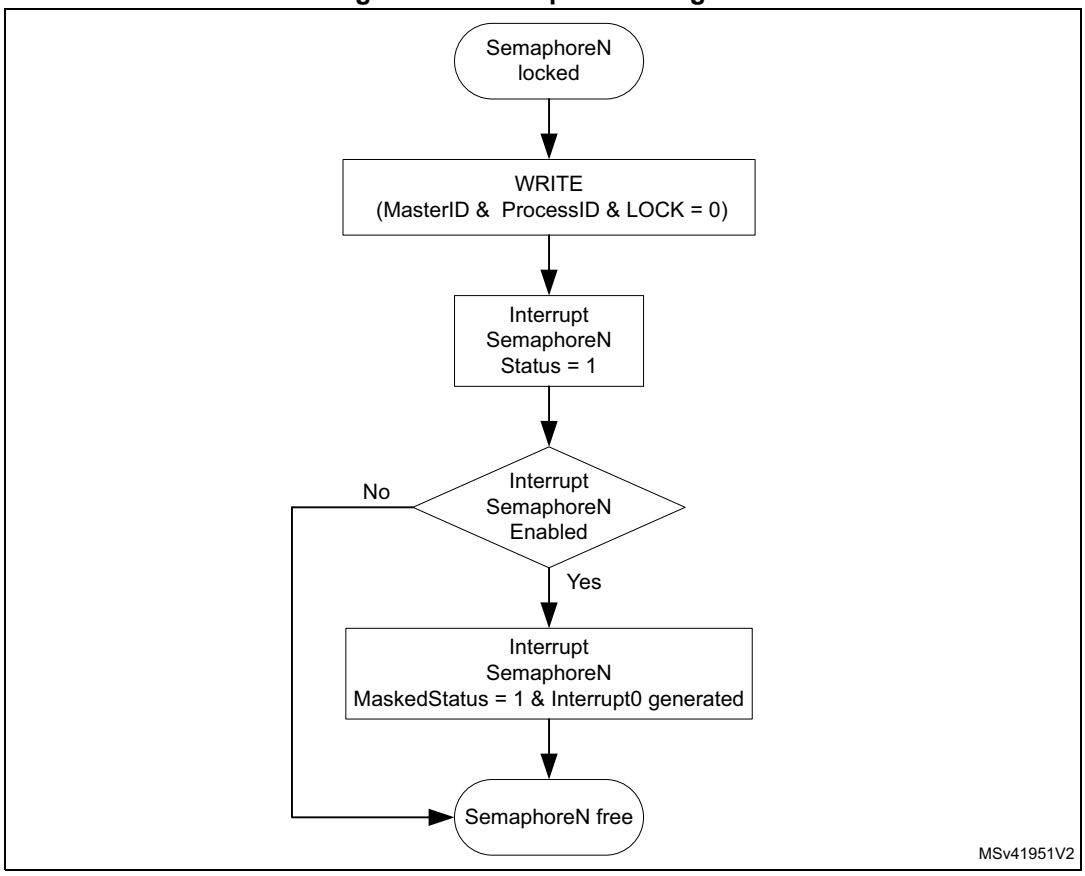
With the Interrupt enable (HSEM_CnIER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts will not set the masked interrupt status for that semaphore, and will not generate an interrupt on the interrupt line.

The Interrupt clear (HSEM_CnICR) will clear the Interrupt status and Masked interrupt status of the associated semaphore for the interrupt line.

The Interrupt status (HSEM_CnISR) mirrors the semaphore Interrupt status of the interrupt line before the Enable.

The Masked interrupt status (HSEM_CnMISR) only mirrors the semaphore Interrupt status of the enabled semaphore interrupts on the interrupt line. All Masked interrupt status of the enabled semaphore need to be cleared in order to clear the interrupt line.

Figure 65. Interrupt state diagram



The procedure to get an interrupt when a semaphore becomes free is described hereafter.

Try to lock the semaphore N

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
  Clear pending semaphore N interrupt status for the interrupt line hsem_intn_it in HSEM_CnICR.
  Re-try to lock the semaphore N again:
  - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock it and clear semaphore interrupt status).
  - If the semaphore lock fails, enable the semaphore N interrupt for the interrupt line hsem_intnn_it in HSEM_CnIER.

On the semaphore N free interrupt, try to lock the semaphore N

- If the semaphore lock is obtained:
  disable the semaphore N interrupt for the interrupt line hsem_intn_it in HSEM_CnIER, and clear pending semaphore N interrupt status for the interrupt line hsem_intn_it in HSEM_CnICR.
- If the semaphore N lock fails:
  Clear pending semaphore N Interrupt status for the interrupt line hsem_intn_it in HSEM_CnICR.
  Try again to lock the semaphore N:
  - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore Interrupt status clear):
    Disable the semaphore interrupt for the interrupt line hsem_intn_it in HSEM_CnIER.
  - If the semaphore lock failed, wait for semaphore free interrupt.

Note: An interrupt will not lock the semaphore. After an interrupt either the AHB bus master or the process still have to perform the lock procedure to lock the semaphore.

9.3.8 AHB bus master ID verification

The HSEM allows only authorized AHB bus master IDs to lock and unlock semaphores.

- The AHB bus master 2-step lock Write access to the semaphore HSEM_Rn register is checked against the valid bus master IDs.
  - Accesses from unauthorized AHB bus master IDs are discarded and will not lock the semaphore.
- The AHB bus master 1-step lock Read access from the semaphore HSEM_RLRn register is checked against the valid bus master IDs.
  - An unauthorized AHB bus master ID read from HSEM_RLRn will return the following Read data depending on the semaphore status:
    - when the semaphore is free it will return all ‘0’s
    - when the semaphore has been locked before it will return the HSEM_RLRn data.
- The MasterID semaphore clears Write access to the HSEM_CR register is checked against the valid bus master IDs. Only the valid bus master IDs can write to the HSEM_CR register and clear any of the MasterID semaphores.
  - Accesses from unauthorized AHB bus master IDs are discarded and will not clear the MasterID semaphore.

Table 85 details the relation between bus master/CPU and MASTERID.

Table 85. Authorized AHB bus master ID

Bus master 0 (CPU)
MASTERID = 3

Note: Accesses from unauthorized AHB bus master IDs to other registers are granted.

9.4 HSEM registers

Registers shall be accessed using Word format. Byte and Half Word access are ignored and will have no effect on the semaphores. Byte and Half Word Read accesses will always return 0. Byte and Half Word accesses will not generate a bus error.

9.4.1 HSEM register semaphore x (HSEM_Rx)

Address offset: 0x000 + 0x4 * x, (x = 0 to 31)

Reset value: 0x0000 0000

The HSEM_R0 - HSEM_R31 shall be used to perform a 2-step Write lock and Read back. Only Write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock indication.
This bit can be written and read by firmware.
0: On Write free semaphore (only when MASTERID and PROCID match), on Read semaphore is free.
1: On Write try to lock semaphore, on Read semaphore is locked.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]:** Semaphore MasterID.
Written by firmware, when the semaphore is free and the LOCK bit is at the same time written to 1, the MASTERID will be written only when the bus ID of the AHB bus master writing the semaphore matches.
When the semaphore is cleared (LOCK bit written to 0 and AHB bus master ID matched MASTERID), the MasterID will be cleared to 0.
When the semaphore is cleared (LOCK bit written to 0 and AHB bus master ID does not match MASTERID), the MasterID will not be affected.
Write when LOCK bit is already 1 (semaphore locked), the MasterID will not be affected.
A Read will return the stored MasterID value.

Bits 7:0 **PROCID[7:0]:** Semaphore ProcessID.
Written by firmware, when the semaphore is free and the lock bit is written to 1, the ProcessID will be set to the written data.
When the semaphore is cleared, (LOCK bit written to 0), the ProcessID will be cleared to 0.
Write when LOCK bit is already 1 (semaphore locked), the ProcessID will not be affected.
A Read will return the programmed ProcessID value.

9.4.2 HSEM read lock register semaphore x (HSEM_RLRx)

Address offset: 0x080 + 0x4 * x, (x = 0 to 31)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM_R0 - HSEM_R31. The HSEM_RLR0 - HSEM_RLR31 shall be used to perform a 1-step Read lock. Only Read accesses with authorized AHB bus master IDs are granted. Read accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **LOCK:** Lock indication.

This bit is read only by firmware at this address. A Read with a valid bus master ID will always return 1.

When the semaphore is free and the firmware performs a Read, hardware will set the semaphore to locked.

When the semaphore is locked and the firmware performs a Read the LOCK bit is not effected.

0: Semaphore is free.

1: Semaphore is locked.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]:** Semaphore MasterID.

This field is read only by firmware at this address.

On a Read, when the semaphore is free, hardware will set the MasterID to the AHB bus master ID reading the semaphore. The MasterID of the AHB bus master locking the semaphore will be read.

On a Read when the semaphore is locked, will return the MasterID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]:** Semaphore ProcessID.

This field is read only by firmware at this address.

On a Read when the semaphore is free will return 0.

On a Read when the semaphore is locked will return the ProcessID of the process that has locked the semaphore.

9.4.3 HSEM interrupt enable register (HSEM_CnIER)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ISE[31:0]:** Interrupt semaphore n enable bit.
This bit is read and written by firmware.
0: Interrupt generation for semaphore n is disabled (masked).
1: Interrupt generation for semaphore n is enabled (not masked).

9.4.4 HSEM interrupt clear register (HSEM_CnICR)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISC[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISC[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ISC[31:0]:** Interrupt semaphore n clear bit.
This bit is written by firmware, and will always read 0.
0: Interrupt semaphore n status and masked status not affected.
1: Interrupt semaphore n status and masked status cleared.

9.4.5 HSEM interrupt status register (HSEM_CnISR)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ISF[31:0]:** Interrupt semaphore n status bit before enable (mask).
This bit is set by hardware and read only by firmware.
Bit will be cleared by firmware writing the corresponding HSEM_CnICR bit.
0: Interrupt semaphore n status, no interrupt pending.
1: Interrupt semaphore n status, interrupt pending.

9.4.6 HSEM masked interrupt status register (HSEM_CnMISR)

Address offset: 0x10C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MISF[31:0]:** masked interrupt semaphore n status bit after enable (mask).
This bit is set by hardware and read only by firmware.
Bit will be cleared by firmware writing the corresponding HSEM_CnICR bit.
0: Interrupt semaphore n status after masking not pending.
1: Interrupt semaphore n status after masking pending.

9.4.7 HSEM clear register (HSEM_CR)

Address offset: 0x140

Reset value: 0x0000 0000

Only Write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MASTERID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				w	w	w	w								

Bits 31:16 **KEY[15:0]:** Semaphore clear Key.
This bit can be written by firmware. Will always read 0.
Key value not matching HSEM_KEYR.KEY, semaphores not effected.
Key value matching HSEM_KEYR.KEY, all semaphores matching the MASTERID will be cleared to the free state.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **MASTERID[3:0]:** MasterID of semaphores to be cleared.
This field can be written by firmware, will always read 0.
Indicates the MasterID for which the semaphores will be cleared when writing the HSEM_CR.

Bits 7:0 Reserved, must be kept at reset value.

9.4.8 HSEM interrupt clear register (HSEM_KEYR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **KEY[15:0]:** Semaphore Clear Key.
This bit can be written and read by firmware.
Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.

9.4.9 HSEM register map

Table 86. HSEM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**HSEM_R0**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**HSEM_R1**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x07C	**HSEM_R31**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	**HSEM_RLR0**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**HSEM_RLR1**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x0FC	**HSEM_RLR31**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID[3:0]				PROCID[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	**HSEM_C1IER**	ISE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**HSEM_C1ICR**	ISC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	**HSEM_C1ISR**	ISF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	**HSEM_C1MISR**	MISF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	**HSEM_CR**	KEY																Res.	Res.	Res.	Res.	MASTERID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144	**HSEM_KEYR**	KEY																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

10 General-purpose I/Os (GPIO)

10.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

10.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

10.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 66 and *Figure 67* show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. *Table 87* gives the possible port bit configurations.

Figure 66. Basic structure of an I/O port bit

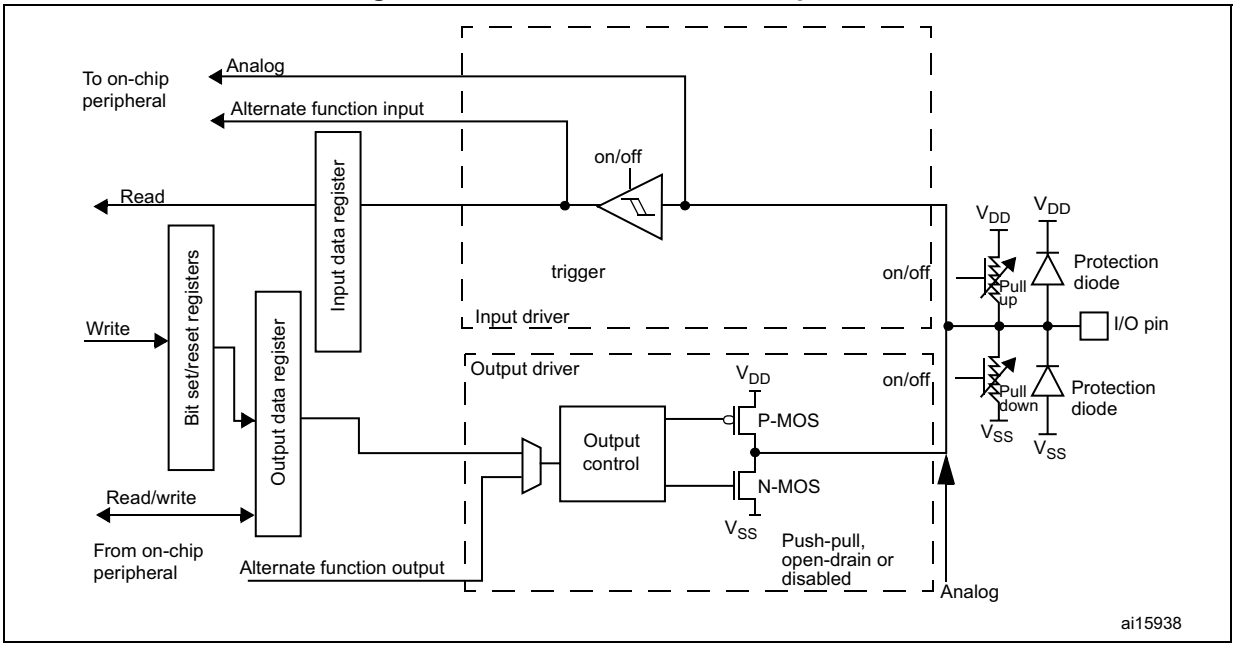
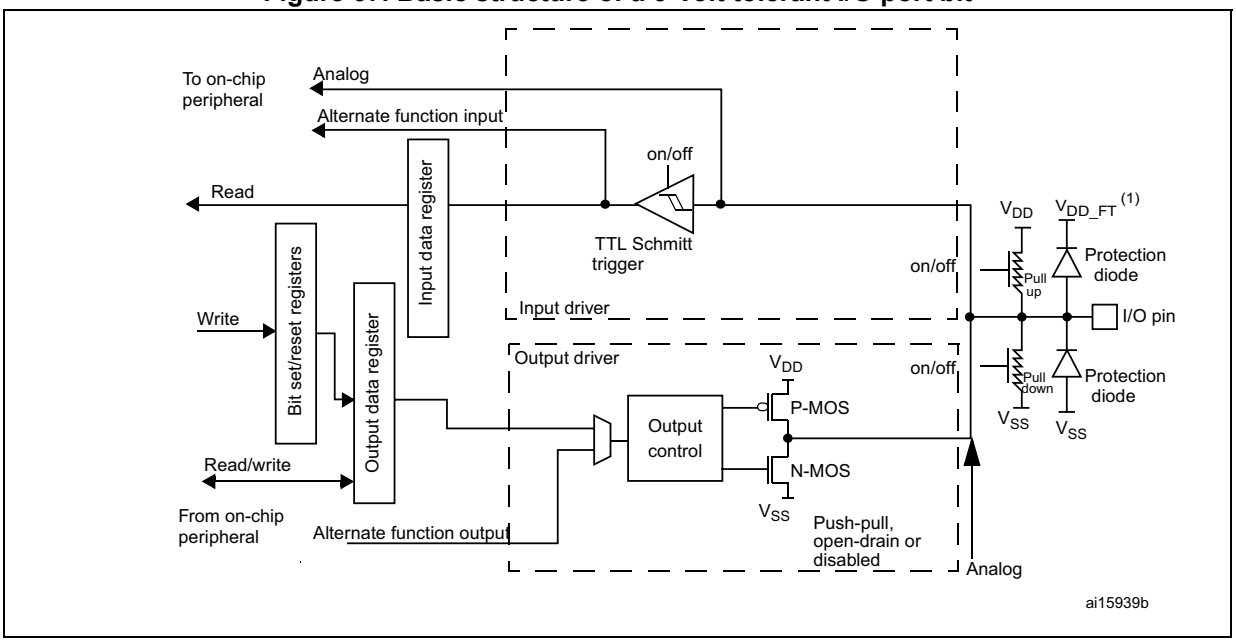


Figure 67. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 87. Port bit configuration table(1)

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

10.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in input floating mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDAT in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO in floating state

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

10.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.
- Cortex-M7 with FPU EVENTOUT is mapped on AF15

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- **Debug function:** after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **System function:** MCOx pins have to be configured in alternate function mode.
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function:**
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.

  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For the ADC and DAC, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and DAC registers. For the additional functions like RTC_OUT, RTC_TS, RTC_TAMPx, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers. For details about I/O control by the RTC, refer to *Section 45.3: RTC functional description on page 1890*.
- EVENTOUT
  - Configure the I/O pin used to output the core EVENTOUT signal by connecting it to AF15.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

10.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

10.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

See *Section 10.4.5: GPIO port input data register (GPIOx_IDR) (x = A to K)* and *Section 10.4.6: GPIO port output data register (GPIOx_ODR) (x = A to K)* for the register descriptions.

10.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

10.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence (refer to *Section 10.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to K)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in *Section 10.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to K)*.

10.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

10.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. Refer to *Section 19: Extended interrupt and event controller (EXTI)Section 19.3: EXTI functional description*.

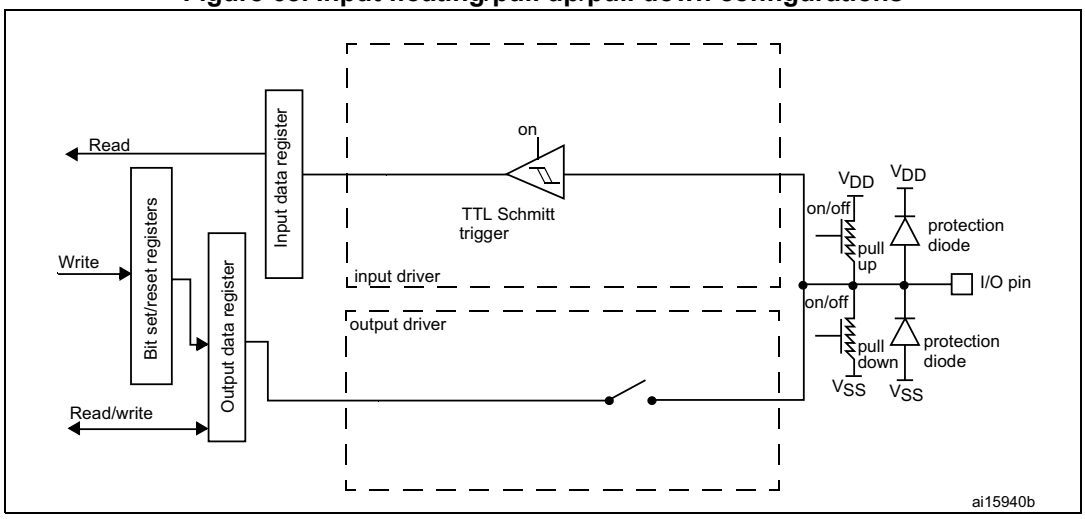
10.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

Figure 68 shows the input configuration of the I/O port bit.

Figure 68. Input floating/pull up/pull down configurations



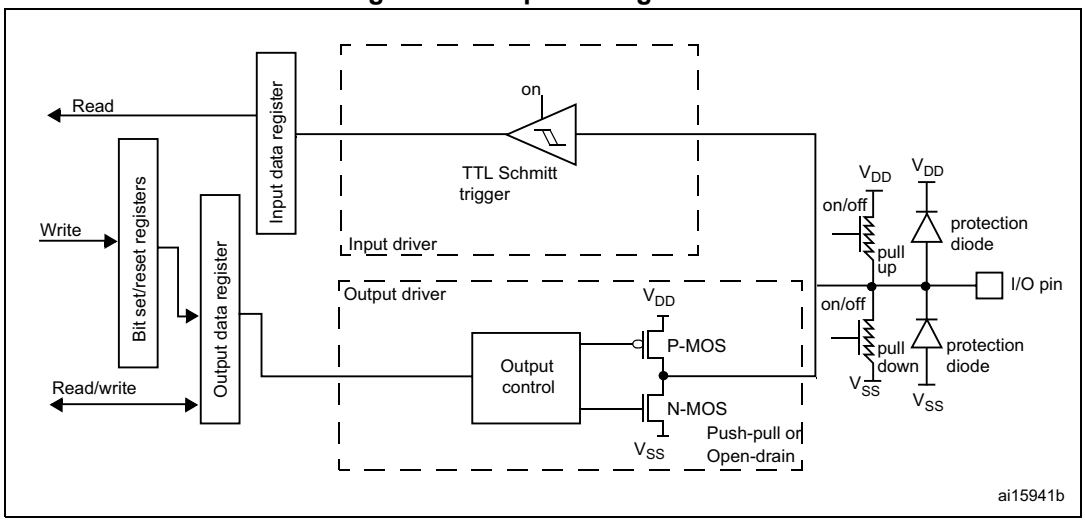
10.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 69 shows the output configuration of the I/O port bit.

Figure 69. Output configuration



10.3.11 I/O compensation cell

This cell is used to control the I/O commutation slew rate (t_{fall} / t_{rise}) to reduce the I/O noise on power supply.

The cell is split into two blocks:

- The first block provides an optimal code for the current PVT. The code stored in this block can be read when the READY flag of the SYSCFG_CCSR is set.
- The second block controls the I/O slew rate.The user selects the code to be applied and programs it by software.

The I/O compensation cell features 2 voltage ranges: 1.62 to 2.0 V and 2.7 to 3.6 V.

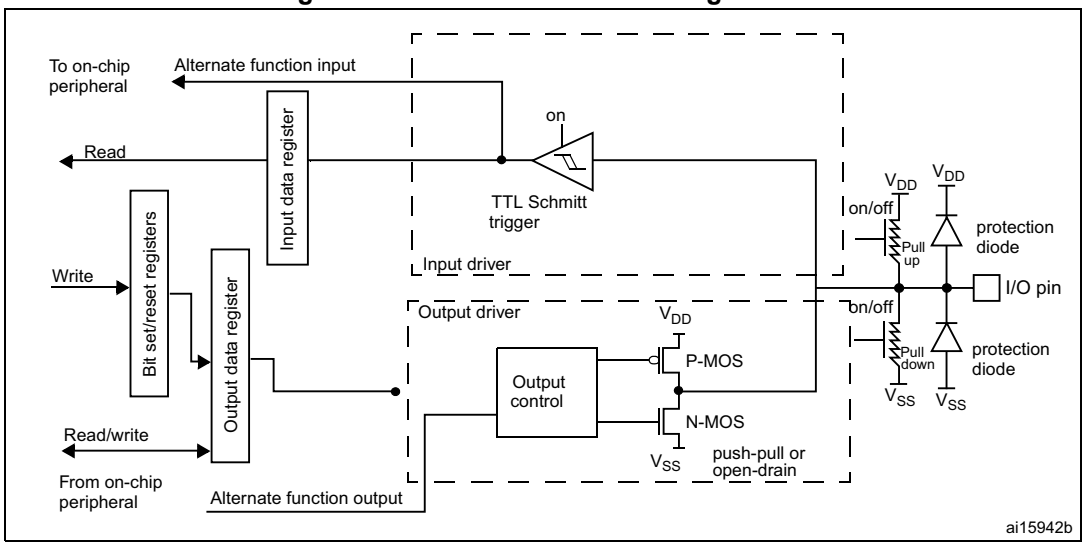
10.3.12 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 70 shows the Alternate function configuration of the I/O port bit.

Figure 70. Alternate function configuration



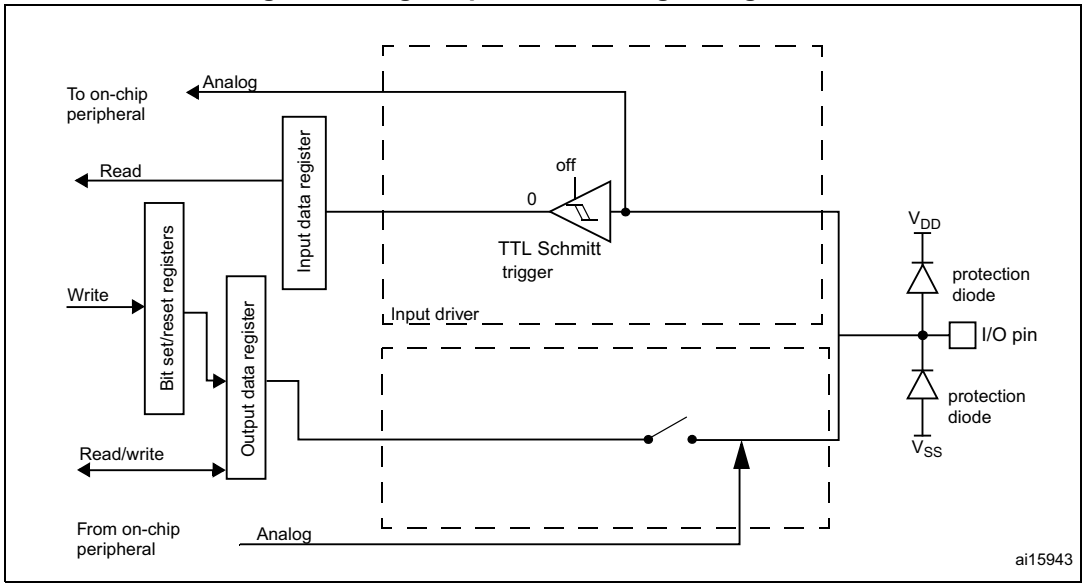
10.3.13 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware
- Read access to the input data register gets the value “0”

Figure 71 shows the high-impedance, analog-input configuration of the I/O port bits.

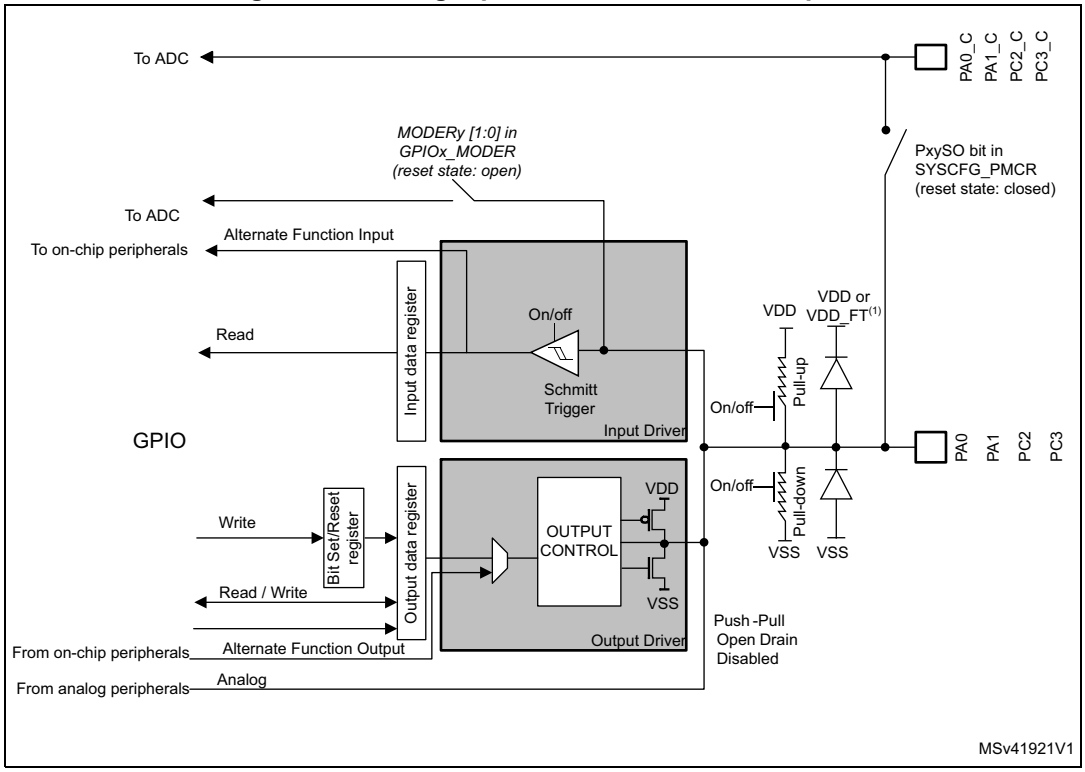
Figure 71. High impedance-analog configuration



Some pins/balls are directly connected to PA0_C, PA1_C, PC2_C and PC3_C ADC analog inputs (see *Figure 72*): there is a direct path between Pxy_C and Pxy pins/balls, through an

analog switch (refer to *Section 11.3.1: SYSCFG peripheral mode configuration register (SYSCFG_PMCR)* for details on how to configure analog switches).

Figure 72. Analog inputs connected to ADC inputs



1. VDD_FT is a potential specific to 5V tolerant I/Os. It is distinct from VDD.

10.3.14 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the OSC_IN or OSC32_IN pin is reserved for clock input and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

10.3.15 Using the GPIO pins in the backup supply domain

The PC13/PC14/PC15/PI8 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

10.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to *Table 88*.

The peripheral registers can be written in word, half word or byte mode.

10.4.1 GPIO port mode register (GPIOx_MODER) (x =A to K)

Address offset:0x00

Reset value:

- 0xABFF FFFF for port A
- 0xFFFF FEBF for port B
- 0xFFFF FFFF for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODER[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

10.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to K)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OT[15:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

10.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to K)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3 [1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDR[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very high speed

Note: Refer to the product datasheets for the values of OSPEEDRy bits versus V_{DD} range and external load.

10.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to K)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDR[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

10.4.5 GPIO port input data register (GPIOx_IDR) (x = A to K)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDR[15:0]:** Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

10.4.6 GPIO port output data register (GPIOx_ODR) (x = A to K)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODR[15:0]:** Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.
Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..F).

10.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to K)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit
Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

10.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to K)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = '1' + LCKR[15:0]

WR LCKR[16] = '0' + LCKR[15:0]

WR LCKR[16] = '1' + LCKR[15:0]

RD LCKR

RD LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit will return '1' until the next MCU reset or peripheral reset.

Bits 15:0 **LCK[15:0]:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is '0.

0: Port configuration not locked

1: Port configuration locked

10.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to K)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFR[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

10.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to J)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFR[15:8][3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

10.4.11 GPIO register map

The following table gives the GPIO register map and reset values.

Table 88. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOA_MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x00	**GPIOB_MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
0x00	**GPIOx_MODER** (where x = **C..K**)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER[1:0]		MODER0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**GPIOx_OTYPER** (where x = **A..K**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOA_OSPEEDR**	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOB_OSPEEDR**	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x08	**GPIOx_OSPEEDR** (where x = **C..K**)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 88. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	**GPIOB_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x0C	**GPIOx_PUPDR** (where x = **C..K**)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = **A..I/J/K**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = **A..K**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = **A..I/J/K**)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = **A..K**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = **A..K**)	AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]				AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH** (where x = **A..K**)	AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]				AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

11 System configuration controller (SYSCFG)

11.1 Introduction

The devices feature a set of configuration registers. The objectives of this section is to describe in details the system configuration controller.

11.2 SYSCFG main features

The system configuration controller main functions are the following:

- Analog switch configuration management
- I2C Fm+ configuration
- Selection of the Ethernet PHY interface.
- Management of the external interrupt line connection to the GPIOs
- Management of the I/O compensation cell
- Getting readout protection and Flash memory bank swap informations
- Management of boot sequences and boot addresses
- Management BOR reset level
- Management of Flash memory secured and protected sector status
- Management Flash memory write protections status
- Management of DTCM secured section status
- Management of independent watchdog behavior (hardware or software / freeze)
- Reset generation in Stop and Standby mode
- Secure mode enabling/disabling status.
  On STM32H750xB devices, only the statuses of Bank 1 option bytes are applicable.

11.3 SYSCFG registers

11.3.1 SYSCFG peripheral mode configuration register (SYSCFG_PMCR)

Address offset: 0x04

Reset value: 0x0X00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PC3SO	PC2SO	PA1SO	PA0SO	EPIS[2:0]			Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw	rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTE	PB9 FMP	PB8 FMP	PB7 FMP	PB6 FMP	I2C4 FMP	I2C3 FMP	I2C2 FMP	I2C1 FMP
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PC3SO**: PC3 Switch Open
This bit controls the analog switch between PC3 and PC3_C (dual pad)
0: Analog switch closed (pads are connected through the analog switch)
1: Analog switch open (2 separated pads)

Bit 26 **PC2SO**: PC2 Switch Open
This bit controls the analog switch between PC2 and PC2_C (dual pad)
0: Analog switch closed (pads are connected through the analog switch)
1: Analog switch open (2 separated pads)

Bit 25 **PA1SO**: PA1 Switch Open
This bit controls the analog switch between PA1 and PA1_C (dual pad)
0: Analog switch closed (pads are connected through the analog switch)
1: Analog switch open (2 separated pads)

Bit 24 **PA0SO**: PA0 Switch Open
This bit controls the analog switch between PA0 and PA0_C (dual pad)
0: Analog switch closed (pads are connected through the analog switch)
1: Analog switch open (2 separated pads)

Bits 23:21 **EPIS[2:0]**: Ethernet PHY Interface Selection
These bits select the Ethernet PHY interface.
000: MII
001: Reserved
010: Reserved
011: Reserved
100: RMII
101: Reserved
110: Reserved
111: Reserved

Bits 20:9 Reserved, must be kept at reset value.

Bit 8 **BOOSTE**: Booster Enable
This bit enables the booster to reduce the total harmonic distortion of the analog switch when the supply voltage is lower than 2.7 V.
Activating the booster allows to guaranty the analog switch AC performance when the supply voltage is below 2.7 V: in this case, the analog switch performance is the same on the full voltage range.
0: Booster disabled
1: Booster enabled

Bit 7 **PB9FMP**: PB(9) Fm+
This bit enables I2C Fm+ on PB(9).
0: Fm+ disabled
1: Fm+ enabled

Bit 6 **PB8FMP**: PB(8) Fm+
This bit enables I2C Fm+ on PB(8).
0: Fm+ disabled
1: Fm+ enabled

Bit 5 **PB7FMP**: PB(7) Fm+
this bit enables I2C Fm+ on PB(7).
0: Fm+ disabled
1: Fm+ enabled

Bit 4 **PB6FMP**: PB(6) Fm+
This bit enables I2C Fm+ on PB(6).
0: Fm+ disabled
1: Fm+ enabled

Bit 3 **I2C4FMP**: I2C4 Fm+
This bit enables Fm+ on I2C4.
0: Fm+ disabled
1: Fm+ enabled

Bit 2 **I2C3FMP**: I2C3 Fm+
This bit enables Fm+ on I2C3.
0: Fm+ disabled
1: Fm+ enabled

Bit 1 **I2C2FMP**: I2C2 Fm+
This bit enables Fm+ on I2C2.
0: Fm+ disabled
1: Fm+ enabled

Bit 0 **I2C1FMP**: I2C1 Fm+
This bit enables Fm+ on I2C1.
0: Fm+ disabled
1: Fm+ enabled

11.3.2 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)

These bits are written by software to select the source input for the EXTI input for external interrupt / event detection.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin
1010: PK[x] pin
Other configurations: reserved

11.3.3 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 4 to 7)

These bits are written by software to select the source input for the EXTI input for external interrupt / event detection.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin
1010: PK[x] pin
Other configurations: reserved

11.3.4 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 8 to 11)

These bits are written by software to select the source input for the EXTI input for external interrupt / event detection.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001: PJ[x] pin
1010: PK[x] pin
Other configurations: reserved

Note: PK[11:8] are not used

11.3.5 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 12 to 15)

These bits are written by software to select the source input for the EXTI input for external interrupt / event detection.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1001: PJ[x] pin
1010: PK[x] pin
Other configurations: reserved

Note: PK[15:12] are not used.

11.3.6 SYSCFG configuration register (SYSCFG_CFGR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AXI SRAML	ITCML	DTCML	SRAM1L	SRAM2L	SRAM3L	SRAM4L	Res.	BK RAML	CM7L	Res.	Res.	FLASHL	PVDL	Res.	Res.
rs	rs	rs	rs	rs	rs	rs		rs	rs			rs	rs		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **AXISRAML**: D1 AXI-SRAM double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the AXI-SRAM double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: AXI-SRAM double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: AXI-SRAM double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs.

Bit 14 **ITCML**: D1 ITCM double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the ITCM double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: ITCM double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: ITCM double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 13 **DTCML**: D1 DTCM double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the DTCM double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: DTCM double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: DTCM double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 12 **SRAM1L**: D2 SRAM1 double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the D2 SRAM1 double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: D2 SRAM1 double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: D2 SRAM1 double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 11 **SRAM2L**: D2 SRAM2 double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the D2 SRAM2 double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: D2 SRAM2 double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: D2 SRAM2 double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 10 **SRAM3L**: D2 SRAM3 double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the D2 SRAM3 double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: D2 SRAM3 double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: D2 SRAM3 double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 9 **SRAM4L**: D3 SRAM4 double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the D3 SRAM4 double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: D3 SRAM4 double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: D3 SRAM4 double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BKRAML**: Backup SRAM double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Backup SRAM double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: Backup SRAM double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: Backup SRAM double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 6 **CM7L**: Arm® Cortex®-M7 LOCKUP (HardFault) output enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the connection of the Arm® Cortex®-M7 LOCKUP (HardFault) output to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: Arm® Cortex®-M7 LOCKUP output disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: Arm® Cortex®-M7 LOCKUP output connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **FLASHL**: FLASH double ECC error lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the FLASH double ECC error flag connection to TIM1/8/15/16/17 and HRTIMER Break inputs.

0: FLASH double ECC error flag disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: FLASH double ECC error flag connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/8/15/16/17 and HRTIMER Break inputs, as well as the PVDE and PLS[2:0] in the PWR_CR1 register.

0: PVD signal disconnected from TIM1/8/15/16/17/HRTIMER Break inputs

1: PVD signal connected to TIM1/8/15/16/17/HRTIMER Break inputs

Bits 1:0 Reserved, must be kept at reset value.

11.3.7 SYSCFG compensation cell control/status register (SYSCFG_CCCSR)

Address offset: 0x20

Reset value: 0x0000 0000

Refer to *Section 10.3.11: I/O compensation cell* for a detailed description of I/O compensation mechanism.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLV
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	Res.	Res.	Res.	Res.	Res.	Res.	CS	EN
							r							rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **HSLV:** High-speed at low-voltage

This bit is written by software to optimize the I/O speed when the product voltage is low.

This bit is active only if IO_HSLV user option bit is set. It must be used only if the product supply voltage is below 2.7 V. Setting this bit when V_{DD} is higher than 2.7 V might be destructive.

0: No I/O speed optimization

1: I/O speed optimization

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **READY:** Compensation cell ready flag

This bit provides the status of the compensation cell.

0: I/O compensation cell not ready

1: I/O compensation cell ready

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CS:** Code selection

This bit selects the code to be applied for the I/O compensation cell.

0: Code from the cell (available in the SYSCFG_CCVR)

1: Code from the SYSCFG compensation cell code register (SYSCFG_CCCR)

Bit 0 **EN:** Enable

This bit enables the I/O compensation cell.

0: I/O compensation cell disabled

1: I/O compensation cell enabled

11.3.8 SYSCFG compensation cell value register (SYSCFG_CCVR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV[3:0]				NCV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **PCV[3:0]:** PMOS compensation value

This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when the CS bit of the SYSCFG_CCCSR is reset.

Bits 3:0 **NCV[3:0]:** NMOS compensation value

This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when the CS bit of the SYSCFG_CCCSR is reset.

11.3.9 SYSCFG compensation cell code register (SYSCFG_CCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC[3:0]				NCC[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **PCC[3:0]:** PMOS compensation code

These bits are written by software to define an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when the CS bit of the SYSCFG_CCCSR is set.

Bits 3:0 **NCC[3:0]:** NMOS compensation code

These bits are written by software to define an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when the CS bit of the SYSCFG_CCCSR is set.

11.3.10 SYSCFG package register (SYSCFG_PKGR)

Address offset: 0x124

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[3:0			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PKG[3:0]:** Package

These bits indicate the device package.
0000: LQFP100
0010: TQFP144
0101: TQFP176/UFBGA176
1000: LQFP208/TFBGA240
Other configurations: all pads enabled

11.3.11 SYSCFG user register 0 (SYSCFG_UR0)

Address offset: 0x300

Reset value: 0x00XX 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDP[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKS
															r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **RDP[7:0]:** Readout protection

These bits indicate the readout protection level.

0xAA: Level 0 (no protection)

0xCC: Level 2 (Flash memory readout protected, full debug features, boot from SRAM and boundary scan disabled)

Other configurations: Level 1 (Flash memory readout protected, limited debug features and boundary scan enabled)

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **BKS:** Bank Swap

This bit indicates Flash memory bank mapping.

0: Flash memory bank addresses are inverted

1: Flash memory banks are mapped to their original addresses

11.3.12 SYSCFG user register 2 (SYSCFG_UR2)

Address offset: 0x308

Reset value: 0xXXXX 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOOT_ADD0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BORH[1:0]	
														r	r

Bits 31:16 **BOOT_ADD0[15:0]:** Boot Address 0
These bits define the MSB of the core boot address when BOOT pin is low.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **BORH[1:0]:** BOR_LVL Brownout Reset Threshold Level
These bits indicate the Brownout reset high level.
0x11: BOR Level 3
0x10: BOR Level 2
0x01: BOR Level 1
0x00: BOR OFF (Level 0)

11.3.13 SYSCFG user register 3 (SYSCFG_UR3)

Address offset: 0x30C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT_ADD1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BOOT_ADD1[15:0]:** Boot Address 1
These bits define the MSB of the core boot address when BOOT pin is high.

11.3.14 SYSCFG user register 4 (SYSCFG_UR4)

Address offset: 0x310

Reset value: 0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEPAD_1
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **MEPAD_1:** Mass Erase Protected Area Disabled for bank 1
This bit indicates if the flash protected area (Bank 1) is affected by a mass erase.
0: When a mass erase occurs the protected area is erased
1: When a mass erase occurs the protected area is not erased

Bits 15:0 Reserved, must be kept at reset value.

11.3.15 SYSCFG user register 5 (SYSCFG_UR5)

Address offset: 0x314

Reset value: 0x00XX 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPS_1[7:0]							
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MESAD_1
															r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRPN_1[7:0]:** Write protection for flash bank 1
WRPN[i] bit indicates if the sector i of the Flash memory bank 1 is protected.
0: Write protection is active on sector i
1: Write protection is not active on sector i

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MESAD_1:** Mass erase secured area disabled for bank 1
This bit indicates if the flash secured area (bank 1) is affected by a mass erase.
0: When a mass erase occurs the secured area is erased
1: When a mass erase occurs the secured area is not erased

11.3.16 SYSCFG user register 6 (SYSCFG_UR6)

Address offset: 0x318

Reset value: 0x0XXX 0XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PA_END_1[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PA_BEG_1[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 23:16 **PA_END_1[11:0]:** Protected area end address for bank 1
End address for bank 1 protected area.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PA_BEG_1[11:0]:** Protected area start address for bank 1
Start address for bank 1 protected area.

11.3.17 SYSCFG user register 7 (SYSCFG_UR7)

Address offset: 0x31C

Reset value: 0x0XXX 0XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SA_END_1[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SA_BEG_1[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 23:16 **SA_END_1[11:0]:** Secured area end address for bank 1
End address for bank 1 secured area.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SA_BEG_1[11:0]:** Secured area start address for bank 1
Start address for bank 1 secured area.

11.3.18 SYSCFG user register 8 (SYSCFG_UR8)

Address offset: 0x320

Reset value: 0x000X 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MESAD_2
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEPAD_2
															r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **MESAD_2:** Mass erase secured area disabled for bank 2
This bit indicates if the Flash memory secured area (Bank 2) is affected by a mass erase.
0: When a mass erase occurs the secured area is erased
1: When a mass erase occurs the secured area is not erased

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MEPAD_2:** Mass erase protected area disabled for bank 2
This bit indicates if the Flash memory protected area (Bank 2) is affected by a mass erase.
0: When a mass erase occurs the protected area is erased
1: When a mass erase occurs the protected area is not erased

11.3.19 SYSCFG user register 9 (SYSCFG_UR9)

Address offset: 0x324

Reset value: 0x0XXX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PA_BEG_2[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPS_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **PA_BEG_2[11:0]:** Protected area start address for bank 2
Start address for bank 2 protected area.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPN_2[7:0]:** Write protection for flash bank 2
WRPN[i] bit indicates if the sector i of the Flash memory bank 2 is protected.
0: Write protection is active on sector i
1: Write protection is not active on sector i

11.3.20 SYSCFG user register 10 (SYSCFG_UR10)

Address offset: 0x328

Reset value: 0x0XXX 0XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SA_BEG_2[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PA_END_2[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SA_BEG_2[11:0]:** Secured area start address for bank 2
Start address for bank 2 secured area.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PA_END_2[11:0]:** Protected area end address for bank 2
End address for bank 2 protected area.

11.3.21 SYSCFG user register 11 (SYSCFG_UR11)

Address offset: 0x32C

Reset value: 0x000X 0XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG1M
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SA_END_2[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IWDG1M:** Independent Watchdog 1 mode
This bit indicates the control mode of the Independent Watchdog 1 (IWDG1).
0: IWDG1 controlled by hardware
1: IWDG1 controlled by software

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **SA_END_2[11:0]:** Secured area end address for bank 2
End address for bank 2 secured area.

11.3.22 SYSCFG user register 12 (SYSCFG_UR12)

Address offset: 0x330

Reset value: 0x000X 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECURE
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
															r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SECURE:** Secure mode
This bit indicates the Secure mode status.
0: Secure mode disabled
1: Secure mode enabled

Bits 15:0 Reserved, must be kept at reset value.

11.3.23 SYSCFG user register 13 (SYSCFG_UR13)

Address offset: 0x334

Reset value: 0x000X 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D1SBRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDRS[1:0]	
														r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **D1SBRST:** D1 Standby reset

This bit indicates if a reset is generated when D1 domain enters DStandby mode.

0: A reset is generated by entering D1 Standby mode

1: D1 Standby mode is entered without reset generation

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **SDRS[1:0]:** Secured DTCM RAM Size

These bits indicates the size of the secured DTCM RAM.

00: 2 Kbytes

01: 4 Kbytes

10: 8 Kbytes

11: 16 Kbytes

11.3.24 SYSCFG user register 14 (SYSCFG_UR14)

Address offset: 0x338

Reset value: 0x000X 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D1STPRST
															rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **D1STPRST:** D1 Stop Reset

This bit indicates if a reset is generated when D1 domain enters in DStop mode.
0: A reset is generated entering D1 Stop mode
1: D1 Stop mode is entered without reset generation

11.3.25 SYSCFG user register 15 (SYSCFG_UR15)

Address offset: 0x33C

Reset value: 0x000X 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZIWDGSTB
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **FZIWDGSTB:** Freeze independent watchdog in Standby mode
This bit indicates if the independent watchdog is frozen in Standby mode.
0: Independent Watchdog frozen in Standby mode
1: Independent Watchdog running in Standby mode

Bits 15:0 Reserved, must be kept at reset value.

11.3.26 SYSCFG user register 16 (SYSCFG_UR16)

Address offset: 0x340

Reset value: 0x000X 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKP
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZIWDG STP
															r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PKP:** Private key programmed
This bit indicates if the device private key is programmed.
0: Private key not programmed
1: Private key programmed

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **FZIWDGSTP:** Freeze independent watchdog in Stop mode
This bit indicates if the independent watchdog is frozen in Stop mode.
0: Independent Watchdog frozen in Stop mode
1: Independent Watchdog running in Stop mode

11.3.27 SYSCFG user register 17 (SYSCFG_UR17)

Address offset: 0x344

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IO_HSLV
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **IO_HSLV:** I/O high speed / low voltage
This bit indicates that the IOHSLV option bit is set.
0: Product is working on the full voltage range
1: Product is working below 2.7 V

11.3.28 SYSCFG register maps

The following table gives the SYSCFG register map and the reset values.

Table 89. SYSCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	Reserved	Reserved																															
0x04	SYSCFG_PMCR	Res.	Res.	Res.	Res.	PC3SO	PC2SO	PA1SO	PA0SO	EPIS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTE	PB9FMP	PB8FMP	PB7FMP	PB6FMP	I2C4FMP	I2C3FMP	I2C2FMP	I2C1FMP
	Reset value					X	X	X	0	0	0	0													0	0	0	0	0	0	0	0	0
0x08	SYSCFG_ EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_ EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_ EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_ EXTICR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SYSCFG_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXISRAML	ITCML	DTCML	SRAM1L	SRAM2L	SRAM3L	SRAM4L	Res.	BKRAML	CM7L	Res.	Res.	FLASHL	PVDL	Res.	Res.
	Reset value																	0	0	0	0	0	0	0		0	0			0	0		
0x20	SYSCFG_CCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	Res.	Res.	Res.	Res.	Res.	CS	EN
	Reset value																0									0						0	0
0x24	SYSCFG_CCVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV[3:0]				NCV[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x28	SYSCFG_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC[3:0]				NCC[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x2C - 0x120	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x124	SYSCFG_PKGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[3:0]			
	Reset value																													X	X	X	X
0x128 - 0x2FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x300	SYSCFG_UR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDP[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKS
	Reset value									x	x	x	x	x	x	x	x																x
0x304	Reserved	Reserved																															

Table 89. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	SYSCFG_UR2	BOOT_ADD0[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BORH[1:0]	
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X															X	X
0x30C	SYSCFG_UR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_ADD1[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x310	SYSCFG_UR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEPAD_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																X																
0x314	SYSCFG_UR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPN_1[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MESAD_1
	Reset value									X	X	X	X	X	X	X	X																X
0x318	SYSCFG_UR6	Res.	Res.	Res.	Res.	PA_END_1[11:0]												Res.	Res.	Res.	Res.	PA_BEG_1[11:0]											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x31C	SYSCFG_UR7	Res.	Res.	Res.	Res.	SA_END_1[11:0]												Res.	Res.	Res.	Res.	SA_BEG_1[11:0]											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x320	SYSCFG_UR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MESAD_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEPAD_2	
	Reset value																x																x
0x324	SYSCFG_UR9	Res.	Res.	Res.	Res.	PA_END_2[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPN_2[7:0]							
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X
0x328	SYSCFG_UR10	Res.	Res.	Res.	Res.	SA_BEG_2[11:0]												Res.	Res.	Res.	Res.	PA_END_2[11:0]											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x32C	SYSCFG_UR11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG1M	Res.	Res.	Res.	Res.	SA_END_2[11:0]											
	Reset value																X					X	X	X	X	X	X	X	X	X	X	X	X
0x330	SYSCFG_UR12	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECURE	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	Res.
	Reset value																X																X
0x334	SYSCFG_UR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D1SBRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDRS[1:0]	
	Reset value																X															X	X

Table 89. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x338	SYSCFG_UR14	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	D1STPRST
	Reset value																X																X
0x33C	SYSCFG_UR15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZIWDGSTB	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	Res.
	Reset value																X																X
0x340	SYSCFG_UR16	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKP	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	FZIWDGST
	Reset value																X																X
0x344	SYSCFG_UR17	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	IO_HSLV
	Reset value																																X

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

12 Block interconnect

12.1 Peripheral interconnect

12.1.1 Introduction

Several peripherals have direct connections between them.

This enables autonomous communication and synchronization between peripherals, thus saving CPU resources and power consumption.

These hardware connections remove software latency, allow the design of a predictable system and result in a reduction of the number of pins and GPIOs.

12.1.2 Connection overview

There are several types of connections.

- Asynchronous connections (A)

  The source output signal is sampled by the destination clock, leading to introduction of a possible jitter in the latency between the source output event and the destination event detection
- Synchronous connections (S)

  Both source and destination are synchronous (they run on the same clock), and the latency from the source to the destination is deterministic. No jitter is introduced.
- Immediate connections (I)

  Either the source or the destination is an analog signal.
- Break/fault connection for TIM/HRTIM outputs (B)

  The source output signal disables the timer outputs through a pure combinational logic path, without any latency.

Table 90. Peripherals interconnect matrix (D2 domain) (1) (2)

Source			Destination																										
			D2 domain																				D3 domain						
			APB1									APB2							AHB1			AHB4	APB4						
			TIM2	TIM3	TIM4	TIM5	TIM12	LPTIM1	DAC	CRS	CAN	TIM1	TIM8	TIM15	TIM16	TIM17	DFSDM1	HRTIM	ADC1	ADC2	ETHERNET	ADC3	LPTIM2	LPTIM3	LPTIM4	LPTIM5	COMP1	COMP2
D2 domain	APB1	TIM2	-	S	S	-	-	-	S	-	A	S	S	S	-	-	-	S	S	S	A	S	-	-	-	-	I	I
		TIM3	S	-	S	S	-	-	-	-	A	S		S	-	-	S	S	S	S	A	S	-	-	-	-	I	I
		TIM4	S	S	-	S	S	-	S	-	-	S	S	S	-	-	S	-	S	S	-	S	-	-	-	-	-	-
		TIM5	-	-	-	-	S	-	S	-	-		S		-	-	-	-	-	-	-	S	-	-	-	-	-	-
		TIM6	-	-	-	-			S	-	-	-	-	-	-	-	S	S	S	S	-	S	-	-	-	-	-	-
		TIM7	-	-	-	-			S	-	-	-	-	-	-	-	S	S	-	-	-	-	-	-	-	-	-	-
		TIM13	-	-	-	-	S			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		TIM14	-	-	-	-	S			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		LPTIM1	-	-	-	-	-	-	A	-	-	-	-	-	-	-	A	A	A	A		A	-	-	-	-	-	-
		SPDIFRX	-	-	-	-	-	-		-	-	-	-	-	-	S	-	-	-	-	-	-	-	-	-	-	-	-
		OPAMP	-	-	-	-	-	-		-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-
		CAN	-	-	-	A	-	-		-	-	-	-	-	-	-	-	A	-	-	A	-	-	-	-	-	-	-
	APB2	TIM1	S	S	S	S	-	-	S	-	-	-	S	S	-	-	S	S	S	S	-	S	-	-	-	-	I	I
		TIM8	S	-	S	S	-	-	S	-	-	-	-	-	-	-	S		S	S	-	S	-	-	-	-	I	I
		TIM15	-	S	-	-	-	-	S	-	-	S	-	-	-	-		S	S	S	-	S	-	-	-	-	I	I
		TIM16	-	-	-	-	-	-	-	-	-	-	-	S	-	-	S	S	-	-	-	-	-	-	-	-	-	-
		TIM17	-	-	-	-	-	-	-	-	-	-	-	S	-	-		S	-	-	-	-	-	-	-	-	-	-
		SAI1	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-	-	-	-
		SAI2	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-	-	-
		DFSDM1	-	-	-	-	-	-	-	-	-	B	B	B	B	B	-	-	-	-	-	-	-	-	-	-	-	-
		HRTIM	-	-	-	-	-	-	A	-	A	-	-	-	-	-	S	-	A	A	A	A	-	-	-	-	-	-
	AHB1	ADC1	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-
		ADC2	-	-	-	-	-	-	-	-	-	-	A	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-
		ETH	A	A	-	-	-	-	-	-	A	-	-	-	-	-	-		-	-	-	-	-	-	-	-	-	-
		USB1 (OTG_HS1)	A			A	-	-		A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		USB2 (OTG_HS2)	A			A	-	-		A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. Letters in the table correspond to the type of connection described in *Section 12.1.2: Connection overview*
2. The "-" symbol in a gray cell means no interconnect.

Table 91. Peripherals interconnect matrix (D3 domain) (1) (2)

Source			Destination																							
			D2 domain																		D3 domain AHB4 APB4				
			APB1									APB2						AHB1								
			TIM2	TIM3	TIM4	TIM5	TIM12	LPTIM1	DAC	CRS	CAN	TIM1	TIM8	TIM15	TIM16	TIM17	DFSDM1	HRTIM	ADC1	ADC2	ETHERNET	ADC3	LPTIM2	LPTIM3	LPTIM4	LPTIM5
D3 Domain	APB4	EXTI	-	-	-	-	-	-	A	-	-	-	-	-	-	-	A	-	A	A	-	-	-	-	-	-
		LPTIM2	-	-	-	-	-	-	A	-	-	-	-	-	-	-	A	A	A	A	-	A	-	A	A	A
		LPTIM3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-	A	A	-	A	-	-	A	A
		LPTIM4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-	A
		LPTIM5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	A	-
		COMP1	A	A	-	-	-	A	-	-	-	A/B	A/B	B	B	B		A/B	-	-	-	-	A	-	-	-
		COMP2	A	A	-	-	-	A	-	-	-	A/B	A/B	B	B	B		A/B	-	-	-	-	A	-	-	-
		SAI4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-	A
		RTC	-	-	-	-	-	A	-	-	-	-	-	-	A	-	-	-	-	-	-	-	A	-	-	-
	AHB4	ADC3	-	-	-	-	-	-	-	-	-	A	A	-	-	-	-	-	-	-	-	-	-	-	-	-
		RCC	A	-	-	-	-	-	-	A	-	-	-	A	A	A	-	-	-	-	-	-	-	-	-	-

1. Letters in the table correspond to the type of connection described in *Section 12.1.2: Connection overview*.
2. The "-" symbol in a gray cell means no interconnect.

Table 92. Peripherals interconnect matrix details(1)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	TRGO	ITR0	TIM2	APB1	D2	S	-
		TIM8	TRGO	ITR1				S	-
	APB1	TIM3	TRGO	ITR2				S	-
		TIM4	TRGO	ITR3				S	-
	AHB1	ETH	PPS	ITR4				S	-
		USB1	SOF	ITR5				S	-
		USB2	SOF	ITR6				S	-
D3	APB4	COMP1	comp1_out	ETR1				I	-
		COMP2	comp2_out	ETR2				I	-
		RCC	lse_ck	ETR3				A	-
D2	APB2	SAI1	SAI1_FS_A	ETR4				A	-
		SAI1	SAI1_FS_B	ETR5				A	-
D3	APB4	COMP1	comp1_out	TI4_1				I	-
		COMP2	comp2_out	TI4_2				I	-
		COMP1 or COMP2(2)	comp1_out or comp2_out	TI4_3				I	-
D2	APB2	TIM1	TRGO	ITR0	TIM3	APB1	D2	S	-
	APB1	TIM2	TRGO	ITR1				S	-
	APB2	TIM15	TRGO	ITR2				S	-
	APB1	TIM4	TRGO	ITR3				S	-
	AHB1	ETH	PPS	ITR4				S	-
D3	APB4	COMP1	comp1_out	ETR1				I	-
		COMP1	comp1_out	TI1_1				I	-
		COMP2	comp2_out	TI1_2				I	-
		COMP1 or COMP2(2)	comp1_out or comp2_out	TI1_3				I	-
D2	APB2	TIM1	TRGO	ITR0	TIM4	APB1	D2	S	-
	APB1	TIM2	TRGO	ITR1				S	-
		TIM3	TRGO	ITR2				S	-
	APB2	TIM8	TRGO	ITR3				S	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	TRGO	ITR0	TIM5	APB1	D2	S	-
		TIM8	TRGO	ITR1				S	-
	APB1	TIM3	TRGO	ITR2				S	-
		TIM4	TRGO	ITR3				S	-
		CAN	SOC	ITR6				S	-
	AHB1	USB1	SOF	ITR7				S	-
		USB2	SOF	ITR8				S	-
	APB2	SAI2	SAI2_FS_A	ETR1				A	-
		SAI2	SAI2_FS_B	ETR2				A	-
	APB1	CAN	TMP	TI1_1				A	-
		CAN	RTP	TI1_2				A	-
D2	APB1	TIM4	TRGO	ITR0	TIM12	APB1	D2	S	-
		TIM5	TRGO	ITR1				S	-
		TIM13	OC1	ITR2				S	-
		TIM14	OC1	ITR3				S	-
	AHB1	USB1	SOF	crs_sync2	CRS	APB1	D2	A	-
		USB2	SOF	crs_sync3	CRS	APB1	D2	A	-
D3	AHB4	RCC	lse_ck	crs_sync1	CRS	APB1	D2	A	-

Table 92. Peripherals interconnect matrix details[1] (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM15	TRGO	ITR0	TIM1	APB2	D2	S	-
	APB1	TIM2	TRGO	ITR1				S	-
		TIM3	TRGO	ITR2				S	-
		TIM4	TRGO	ITR3				S	-
D3	APB4	COMP1	comp1_out	ETR1				I	-
		COMP2	comp2_out	ETR2				I	-
D2	AHB1	ADC1	adc1_awd1	ETR3				A	-
		ADC1	adc1_awd2	ETR4				A	-
		ADC1	adc1_awd3	ETR5				A	-
D3	AHB4	ADC3	adc3_awd1	ETR6				A	-
		ADC3	adc3_awd2	ETR7				A	-
		ADC3	adc3_awd3	ETR8				A	-
	APB4	COMP1	comp1_out	TI1_1				I	-
		COMP1	comp1_out	BRK_1				B	-
		COMP2	comp2_out	BRK_2				B	-
D2	APB2	DFSDM1	dfsdm1_ break0	BRK_3				B	-
D3	APB4	COMP1	comp1_out	BRK2_1				B	-
		COMP2	comp2_out	BRK2_2				B	-
D2	APB2	DFSDM1	dfsdm1_ break1	BRK2_3				B	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	TRGO	ITR0	TIM8	APB2	D2	S	-
	APB1	TIM2	TRGO	ITR1				S	-
		TIM4	TRGO	ITR2				S	-
		TIM5	TRGO	ITR3				S	-
D3	APB4	COMP1	comp1_out	ETR1				I	-
		COMP2	comp2_out	ETR2				I	-
D2	AHB1	ADC2	adc2_awd1	ETR3				A	-
		ADC2	adc2_awd2	ETR4				A	-
		ADC2	adc2_awd3	ETR5				A	-
D3	AHB4	ADC3	adc3_awd1	ETR6				A	-
		ADC3	adc3_awd2	ETR7				A	-
		ADC3	adc3_awd3	ETR8				A	-
	APB4	COMP2	comp2_out	TI1_1				I	-
		COMP1	comp1_out	BRK_1				B	-
		COMP2	comp2_out	BRK_2				B	-
D2	APB2	DFSDM1	dfsdm1_ break2	BRK_3				B	-
D3	APB4	COMP1	comp1_out	BRK2_1				B	-
		COMP2	comp2_out	BRK2_2				B	-
D2	APB2	DFSDM1	dfsdm1_ break3	BRK2_3				B	-

Table 92. Peripherals interconnect matrix details[1] (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	TRGO	ITR0	TIM15	APB2	D2	S	-
	APB1	TIM3	TRGO	ITR1				S	-
	APB2	TIM16	OC1	ITR2				S	-
		TIM17	OC1	ITR3				S	-
	APB1	TIM2	CH1	TI1_1				A	-
		TIM3	CH1	TI1_2				A	-
		TIM4	CH1	TI1_3				A	-
D3	AHB4	RCC	lse_ck	TI1_4				A	-
		RCC	csi_ck	TI1_5				A	-
		RCC	MCO2	TI1_6				A	-
D2	APB1	TIM2	CH2	TI2_1				A	-
		TIM3	CH2	TI2_2				A	-
		TIM4	CH2	TI2_3				A	-
D3	APB4	COMP1	comp1_out	BRK_1				B	-
		COMP2	comp2_out	BRK_2				B	-
D2	APB2	DFSDM1	dfsdm_ break0	BRK_3				B	-
D3	AHB4	RCC	lsi_ck	TI1_1	TIM16	APB2	D2	A	-
		RCC	lse_ck	TI1_2				A	-
	APB4	RTC	WKUP_IT	TI1_3				A	-
		COMP1	comp1_out	BRK_1				B	-
		COMP2	comp2_out	BRK_2				B	-
D2	APB2	DFSDM1	dfsdm_ break1	BRK_3				B	-
D2	APB1	SPDIFRX	spdifrx_frame_ sync	TI1_1	TIM17	APB2	D2	A	-
D3	AHB4	RCC	HSE_1MHZ	TI1_2				A	-
		RCC	MCO1	TI1_3				A	-
	APB4	COMP1	comp1_out	BRK_1				B	-
		COMP2	comp2_out	BRK_2				B	-
D2	APB2	DFSDM1	dfsdm_ break2	BRK_3				B	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D3	APB4	COMP1	comp1_out	hrtim_evt11	HRTIM	APB2	D2	B	-
D2	APB2	TIM1	TRGO	hrtim_evt12				B	-
	AHB1	ADC1	adc1_awd1	hrtim_evt13				B	-
D3	APB4	COMP2	OUT	hrtim_evt21				B	-
D2	APB2	TIM2	TRGO	hrtim_evt22				B	-
	AHB1	ADC1	adc1_awd2	hrtim_evt23				B	-
	NC	NC	NC	hrtim_evt31				B	-
	APB2	TIM3	TRGO	hrtim_evt32				B	-
	AHB1	ADC1	adc1_awd3	hrtim_evt33				B	-
	APB1	OPAMP1	opamp1_out	hrtim_evt41				B	-
		TIM7	TRGO	hrtim_evt42				B	-
	AHB1	ADC2	adc2_awd1	hrtim_evt43				B	-
	NC	NC	NC	hrtim_evt51				B	-
	APB1	LPTIM1	lptim1_out	hrtim_evt52				B	-
	AHB1	ADC2	adc2_awd2	hrtim_evt53				B	-
D3	APB4	COMP1	comp1_out	hrtim_evt61				I	-
D2	APB1	TIM6	TRGO	hrtim_evt62				S	-
	AHB1	ADC2	adc2_awd3	hrtim_evt63				A	-
D3	APB4	COMP2	comp2_out	hrtim_evt71				I	-
D2	APB1	TIM7	TRGO	hrtim_evt72				S	-
	NC	NC	NC	hrtim_evt73				-	-
				hrtim_evt81				-	-
	APB1	TIM6	TRGO	hrtim_evt82				S	-
	APB1	CAN	TTCAN_TMP	hrtim_evt83				A	-
		OPAMP1	opamp1_out	hrtim_evt91				I	-
	APB2	TIM15	TRGO	hrtim_evt92				S	-
	APB1	CAN	TTCAN_RTP	hrtim_evt93				A	-
	NC	NC	NC	hrtim_ evt101					-
D3	APB4	LPTIM2	lptim2_out	hrtim_ evt102				A	-
D2	APB1	CAN	TTCAN_SOC	hrtim_ evt103				A	-

Table 92. Peripherals interconnect matrix details[1] (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D3	APB4	COMP1	comp1_out	hrtim_in_flt1	HRTIM	APB2	D2	B	-
		COMP2	comp2_out	hrtim_in_flt2				B	-
D2	APB2	TIM16	OC	hrtim_upd_en1				S	-
		TIM17	OC	hrtim_upd_en2				S	-
	APB1	TIM6	TRGO	hrtim_upd_en3				S	-
		TIM7	TRGO	hrtim_bm_trg				S	-
	APB2	TIM16	OC	hrtim_bm_ck1				S	-
		TIM17	OC	hrtim_bm_ck2				S	-
	APB1	TIM7	TRGO	hrtim_bm_ck3				S	-
D3	APB4	RTC	rtc_alarm_a_evt	lptim1_ext_trg0	LPTIM1	APB1	D2	A	-
		RTC	rtc_alarm_b_evt	lptim1_ext_trg1				A	-
		RTC	rtc_tamp1_evt	lptim1_ext_trg2				A	-
		RTC	rtc_tamp2_evt	lptim1_ext_trg3				A	-
		RTC	rtc_tamp3_evt	lptim1_ext_trg4				A	-
		COMP1	comp1_out	lptim1_ext_trg5				I	-
		COMP2	comp2_out	lptim1_ext_trg6				I	-
		COMP1	comp1_out	lptim1_in1_mux1				I	-
		COMP2	comp2_out	lptim1_in2_mux2				I	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D3	APB4	RTC	rtc_alarm_a_evt	lptim2_ext_trg0	LPTIM2	APB4	D3	A	-
		RTC	rtc_alarm_b_evt	lptim2_ext_trg1				A	-
		RTC	rtc_tamp1_evt	lptim2_ext_trg2				A	-
		RTC	rtc_tamp2_evt	lptim2_ext_trg3				A	-
		RTC	rtc_tamp3_evt	lptim2_ext_trg4				A	-
		COMP1	comp1_out	lptim2_ext_trg5				I	-
		COMP2	comp2_out	lptim2_ext_trg6				I	-
		COMP1	comp1_out	lptim2_in1_mux1				I	-
		COMP2	comp2_out	lptim2_in1_mux2				I	-
		COMP1 or COMP2 (2)	comp1_out or comp2_out	lptim2_in1_mux3				I	-
		COMP2	comp2_out	lptim2_in2_mux1				I	-
D3	APB4	LPTIM2	lptim2_out	lptim3_ext_trg0	LPTIM3	APB4	D3	S	If same kernel clock source
		NC	NC	lptim3_ext_trg1				-	-
		LPTIM4	lptim4_out	lptim3_ext_trg2				S	If same kernel clock source
		LPTIM5	lptim5_out	lptim3_ext_trg3				S	If same kernel clock source
D2	APB2	SAI1	SAI1_FS_A	lptim3_ext_trg4				A	-
		SAI1	SAI1_FS_B	lptim3_ext_trg5				A	-
		SAI1	SAI1_FS_A	lptim3_in1_mux1				A	-
		SAI1	SAI1_FS_B	lptim3_in1_mux2				A	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D3	APB4	LPTIM2	lptim2_out	lptim4_ext_trg 0	LPTIM4	APB4	D3	S	If same kernel clock source
		LPTIM3	lptim3_out	lptim4_ext_trg 1				S	If same kernel clock source
		NC	NC	lptim4_ext_trg 2					-
		LPTIM5	lptim5_out	lptim4_ext_trg 3				S	If same kernel clock source
D2	APB2	SAI2	SAI2_FS_A	lptim4_ext_trg 4				A	-
		SAI2	SAI2_FS_B	lptim4_ext_trg 5				A	-
D3	APB4	LPTIM2	lptim2_out	lptim5_ext_trg 0	LPTIM5	APB4	D3	S	If same kernel clock source
		LPTIM3	lptim3_out	lptim5_ext_trg 1				S	If same kernel clock source
		LPTIM4	lptim4_out	lptim5_ext_trg 2				S	If same kernel clock source
		SAI4	SAI4_FS_A	lptim5_ext_trg 3				A	-
		SAI4	SAI4_FS_B	lptim5_ext_trg 4				A	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	TRGO	dac_ch1/2_trg0	DAC channel 1/channel 2	APB1	D2	S	-
	APB1	TIM2	TRGO	dac_ch1/2_trg1				S	-
		TIM4	TRGO	dac_ch1/2_trg02				S	-
		TIM5	TRGO	dac_ch1/2_trg3				S	-
		TIM6	TRGO	dac_ch1/2_trg4				S	-
		TIM7	TRGO	dac_ch1/2_trg5				S	-
	APB2	TIM8	TRGO	dac_ch1/2_trg6				S	-
		TIM15	TRGO	dac_ch1/2_trg7				S	-
		HRTIM1	hrtim_dac_trg1	dac_ch1/2_trg8				S	-
			hrtim_dac_trg2	dac_ch1/2_trg9				S	-
	APB1	LPTIM1	lptim1_out	dac_ch1/2_trg10				S	-
		LPTIM2	lptim2_out	dac_ch1/2_trg11				S	-
D3	APB4	SYSCFG	EXTI9	dac_ch1/2_trg12				S	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	TRGO	TRG0	DFSDM1	APB2	D2	S	-
		TIM1	TRGO2	TRG1				S	-
		TIM8	TRGO	TRG2				S	-
		TIM8	TRGO2	TRG3				S	-
	APB1	TIM3	TRGO	TRG4				S	-
		TIM4	TRGO	TRG5				S	-
	APB2	TIM16	OC1	TRG6				S	-
	APB1	TIM6	TRGO	TRG7				S	-
		TIM7	TRGO	TRG8				S	-
	APB2	HRTIM1	hrtim_adc_trg1	TRG9				S	-
		HRTIM1	hrtim_adc_trg3	TRG10				S	-
D3	APB4	SYSCFG	EXTI11	TRG24				A	-
		SYSCFG	EXTI15	TRG25				A	-
D2	APB1	LPTIM1	lptim1_out	TRG26				A	-
D3	APB4	LPTIM2	lptim2_out	TRG27				A	-
		LPTIM3	lptim3_out	TRG28				A	-
D2	APB2	TIM1	CC1	adc_ext_trg0	ADC1 / ADC2	AHB1	D2	S	-
		TIM1	CC2	adc_ext_trg1				S	-
		TIM1	CC3	adc_ext_trg2				S	-
	APB1	TIM2	CC2	adc_ext_trg3				S	-
		TIM3	TRGO	adc_ext_trg4				S	-
		TIM4	CC4	adc_ext_trg5				S	-
D3	APB4	SYSCFG	EXTI11	adc_ext_trg6				A	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM8	TRGO	adc_ext_trg7	ADC1 / ADC2	AHB1	D2	S	-
		TIM8	TRGO2	adc_ext_trg8				S	-
		TIM1	TRGO	adc_ext_trg9				S	-
		TIM1	TRGO2	adc_ext_trg10				S	-
	APB1	TIM2	TRGO	adc_ext_trg11				S	-
		TIM4	TRGO	adc_ext_trg12				S	-
		TIM6	TRGO	adc_ext_trg13				S	-
	APB2	TIM15	TRGO	adc_ext_trg14				S	-
	APB1	TIM3	CC4	adc_ext_trg15				S	-
	APB2	HRTIM1	hrtim_adc_trg1	adc_ext_trg16				A	-
		HRTIM1	hrtim_adc_trg3	adc_ext_trg17				A	-
		LPTIM1	lptim1_out	adc_ext_trg18				A	-
D3	APB4	LPTIM2	lptim2_out	adc_ext_trg19				A	-
		LPTIM3	lptim3_out	adc_ext_trg20				A	-
D2	APB2	TIM1	TRGO	adc_jext_trg0	ADC1 / ADC2	AHB1	D2	S	-
		TIM1	CC4	adc_jext_trg1				S	-
	APB1	TIM2	TRGO	adc_jext_trg2				S	-
		TIM2	CC1	adc_jext_trg3				S	-
		TIM3	CC4	adc_jext_trg4				S	-
		TIM4	TRGO	adc_jext_trg5				S	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D3	APB4	SYSCFG	EXTI15	adc_jext_trg6	ADC1 / ADC2	AHB1	D2	A	-
D2	APB2	TIM8	CC4	adc_jext_trg7				S	-
		TIM1	TRGO2	adc_jext_trg8				S	-
		TIM8	TRGO	adc_jext_trg9				S	-
		TIM8	TRGO2	adc_jext_trg10				S	-
	APB1	TIM3	CC3	adc_jext_trg11				S	-
		TIM3	TRGO	adc_jext_trg12				S	-
		TIM3	CC1	adc_jext_trg13				S	-
		TIM6	TRGO	adc_jext_trg14				S	-
	APB2	TIM15	TRGO	adc_jext_trg15				S	-
		HRTIM1	hrtim_adc_trg2	adc_jext_trg16				A	-
		HRTIM1	hrtim_adc_trg4	adc_jext_trg17				A	-
	APB1	LPTIM1	lptim1_out	adc_jext_trg18				A	-
D3	APB4	LPTIM2	lptim2_out	adc_jext_trg19				A	-
		LPTIM3	lptim2_out	adc_jext_trg20				A	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	CC1	adc_ext_trg0	ADC3	AHB4	D3	S	-
		TIM1	CC2	adc_ext_trg1				S	-
		TIM1	CC3	adc_ext_trg2				S	-
	APB1	TIM2	CC2	adc_ext_trg3				S	-
		TIM3	TRGO	adc_ext_trg4				S	-
		TIM4	CC4	adc_ext_trg5				S	-
D3	APB4	SYSCFG	EXTI11	adc_ext_trg6				A	-
D2	APB2	TIM8	TRGO	adc_ext_trg7				S	-
		TIM8	TRGO2	adc_ext_trg8				S	-
		TIM1	TRGO	adc_ext_trg9				S	-
		TIM1	TRGO2	adc_ext_trg10				S	-
	APB1	TIM2	TRGO	adc_ext_trg11				S	-
		TIM4	TRGO	adc_ext_trg12				S	-
		TIM6	TRGO	adc_ext_trg13				S	-
	APB2	TIM15	TRGO	adc_ext_trg14				S	-
	APB1	TIM3	CC4	adc_ext_trg15				S	-
	APB2	HRTIM1	hrtim_adc_trg1	adc_ext_trg16				A	-
		HRTIM1	hrtim_adc_trg3	adc_ext_trg17				A	-
		LPTIM1	lptim1_out	adc_ext_trg18				A	-
D3	APB4	LPTIM2	lptim2_out	adc_ext_trg19				A	-
		LPTIM3	lptim3_out	adc_ext_trg20				A	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB2	TIM1	TRGO	adc_jext_trg0	ADC3	AHB4	D3	SI	-
		TIM1	CC4	adc_jext_trg1				S	-
	APB1	TIM2	TRGO	adc_jext_trg2				S	-
		TIM2	CC1	adc_jext_trg3				S	-
		TIM3	CC4	adc_jext_trg4				S	-
		TIM4	TRGO	adc_jext_trg5				S	-
D3	APB4	SYSCFG	EXTI15	adc_jext_trg6				A	-
D2	APB2	TIM8	CC4	adc_jext_trg7				S	-
		TIM1	TRGO2	adc_jext_trg8				S	-
		TIM8	TRGO	adc_jext_trg9				S	-
		TIM8	TRGO2	adc_jext_trg10				S	-
	APB1	TIM3	CC3	adc_jext_trg11				S	-
		TIM3	TRGO	adc_jext_trg12				S	-
		TIM3	CC1	adc_jext_trg13				S	-
		TIM6	TRGO	adc_jext_trg14				S	-
	APB2	TIM15	TRGO	adc_jext_trg15				S	-
		HRTIM1	hrtim_adc_trg2	adc_jext_trg16				A	-
		HRTIM1	hrtim_adc_trg4	adc_jext_trg17				A	-
	APB1	LPTIM1	OUT	adc_jext_trg18				A	-
D3	APB4	LPTIM2	OUT	adc_jext_trg19				A	-
		LPTIM3	OUT	adc_jext_trg20				A	-
D2	APB2	TIM1	OC5	comp_blk1	COMP1 / COMP2	APB4	D3	I	-
		TIM1	OC3	comp_blk2				I	-
	APB1	TIM3	OC3	comp_blk3				I	-
		TIM3	OC4	comp_blk4				I	-
	APB2	TIM8	OC5	comp_blk5				I	-
		TIM15	OC1	comp_blk6				I	-

Table 92. Peripherals interconnect matrix details(1) (continued)

Source				Destination				Type	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain		
D2	APB1	TIM2	TRGO	SWT0	FDCAN	APB1	D2	A	-
		TIM3	TRGO	SWT1				A	-
	AHB1	ETH	PPS	SWT2				A	-
	APB2	HRTIM1	hrtim_dac_trg1	SWT3				A	-
	APB1	TIM2	TRGO	EVT0				A	-
		TIM3	TRGO	EVT1				A	-
	AHB1	ETH	PPS	EVT2				A	-
	APB2	HRTIM1	hrtim_dac_trg1	EVT3				A	-
	APB1	TIM2	TRGO	PTP0	ETH	AHB1	D2	A	-
		TIM3	TRGO	PTP1				A	-
	APB2	HRTIM1	hrtim_dac_trg2	PTP2				A	-
	APB1	CAN	TMP	PTP3				A	-

1. Letters in the table correspond to the type of connection described in *Section 12.1.2: Connection overview*.
2. comp1_out and comp2_out are connected to the inputs of an OR gate. The output of this OR gate is connected to the The lptim2_in1_mux3 input.

12.2 Wakeup from low power modes

The Extended interrupt and event controller module (EXTI) allows to wake up the system from Stop mode and/or a CPU from CStop mode. Wakeup events are coming from peripherals.

These events are handled by the EXTI either as Configurable events (**C**), or as Direct events (**D**). See *Type* column in *Table 93*. Refer to *Section 19: Extended interrupt and event controller (EXTI)* for further details.

Three types of peripheral output signals are connected to the EXTI input events:

- The wake up signals. These signals can be generated by the peripheral without any bus interface clock, they are referred to as xxx_wkup in *Table 93*. Some peripherals do not have this capability.
- The interrupt signals. These signals can be generated only if the peripheral bus interface clock is running. These interrupt signals are generally directly connected to the NVIC of CPU. They are referred to as xxx_it.
- The signals, i.e. the pulses generated by the peripheral. Once a peripheral has generated a signal, no action (flag clearing) is required at peripheral level.

Each EXTI input event has a different wakeup capability or possible target (see *Target* column in *Table 93*):

- CPU wakeup (**CPU**): the input event can be enabled to wake up the CPU
- CPU and D3 domain wakeup for autonomous Run mode (**ANY**): the input event can be enabled to wake up the CPU or the D3 domain only for an autonomous Run mode phase.

Table 93. EXTI wakeup inputs(1)

Source				Destination		Type	Target	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral			
D3	APB4	SYSCFG	exti0_wkup	WKUP0	EXTI	C	ANY	-
			exti1_wkup	WKUP1				-
			exti2_wkup	WKUP2				-
			exti3_wkup	WKUP3				-
			exti4_wkup	WKUP4				-
			exti5_wkup	WKUP5				-
			exti6_wkup	WKUP6				-
			exti7_wkup	WKUP7				-
			exti8_wkup	WKUP8				-
			exti9_wkup	WKUP9				-
			exti10_wkup	WKUP10				-
			exti11_wkup	WKUP11				-
			exti12_wkup	WKUP12				-
			exti13_wkup	WKUP13				-
			exti14_wkup	WKUP14				-
			exti15_wkup	WKUP15				-
D3	AHB4	PWR	pvd_avd_wkup	WKUP16		C	CPU	-
D3	APB4	RTC	ALARMS	WKUP17		D	CPU	-
D3	APB4	RTC	TAMPER TIMESTAMP	WKUP18		C	CPU	-
D3	AHB4	RCC	CSS_LSE					-
D3	APB4	RTC	WKUP	WKUP19		C	ANY	-
D3	APB4	COMP1	comp1_out	WKUP20		C	ANY	-
D3	APB4	COMP2	comp2_out	WKUP21		C	ANY	-
D2	APB1	I2C1	i2c1_wkup	WKUP22		C	CPU	-
D2	APB1	I2C2	i2c2_wkup	WKUP23		D	CPU	-
D2	APB1	I2C3	i2c3_wkup	WKUP24		D	CPU	-
D2	APB1	I2C4	i2c4_wkup	WKUP25		D	ANY	-
D2	APB2	USART1	usart1_wkup	WKUP26		D	CPU	-
D2	APB1	USART2	usart2_wkup	WKUP27		D	CPU	-

Table 93. EXTI wakeup inputs[1] (continued)

Source				Destination		Type	Target	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral			
D2	APB1	USART3	usart3_wkup	WKUP28	EXTI	D	CPU	-
D2	APB2	USART6	usart6_wkup	WKUP29		D	CPU	-
D2	APB1	UART4	uart4_wkup	WKUP30		D	CPU	-
D2	APB1	UART5	uart5_wkup	WKUP31		D	CPU	-
D2	APB1	UART7	uart7_wkup	WKUP32		D	CPU	-
D2	APB1	UART8	uart8_wkup	WKUP33		D	CPU	-
D3	APB4	LPUART	lpuart_rx_wkup	WKUP34		D	ANY	-
D3	APB4	LPUART	lpuart_tx_wkup	WKUP35		D	ANY	-
D2	APB2	SPI1	spi1_wkup	WKUP36		D	CPU	-
D2	APB1	SPI2	spi2_wkup	WKUP37		D	CPU	-
D2	APB1	SPI3	spi3_wkup	WKUP38		D	CPU	-
D2	APB2	SPI4	spi4_wkup	WKUP39		D	CPU	-
D2	APB2	SPI5	spi5_wkup	WKUP40		D	CPU	-
D3	APB4	SPI6	spi6_wkup	WKUP41		D	ANY	-
D2	APB1	MDIOS	mdios_wkup	WKUP42		D	CPU	-
D2	AHB1	USB1	usb1_wkup	WKUP43		D	CPU	-
D2	AHB1	USB2	usb2_wkup	WKUP44		D	CPU	-
-	-	NC	NC	WKUP45		-	-	-
D2	APB1	LPTIM1	lptim1_wkup	WKUP47		D	CPU	-
D3	APB4	LPTIM2	lptim2_wkup	WKUP48		D	ANY	-
D3	APB4	LPTIM2	lptim2_out	WKUP49		C	ANY	(2)
D3	APB4	LPTIM3	lptim3_wkup	WKUP50		D	ANY	-
D3	APB4	LPTIM3	lptim3_out	WKUP51		C	ANY	(2)
D3	APB4	LPTIM4	lptim4_wkup	WKUP52		D	ANY	-
D3	APB4	LPTIM5	lptim5_wkup	WKUP53		D	ANY	-
D2	APB1	SWPMI	swpmi_wkup	WKUP54		D	CPU	-

Table 93. EXTI wakeup inputs(1) (continued)

Source				Destination		Type	Target	Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral			
D3	AHB4	PWR	pwr_wkup1_wkup	WKUP55	EXTI	D	CPU	-
			pwr_wkup2_wkup	WKUP56				-
			pwr_wkup3_wkup	WKUP57				-
			pwr_wkup4_wkup	WKUP58				-
			pwr_wkup5_wkup	WKUP59				-
			pwr_wkup6_wkup	WKUP60				-
D3	AHB4	RCC	rcc_it	WKUP61		D	CPU	-
D3	APB4	I2C4	i2c4_ev_it	WKUP62		D	CPU	(1)
		I2C4	i2c4_err_it	WKUP63		D	CPU	(1)
D3	APB4	LPUART1	lpuart1_it	WKUP64		D	CPU	(1)
D3	APB4	SPI6	spi6_it	WKUP64		D	CPU	(1)
D3	AHB4	BDMA	bdma_ch0_it	WKUP66		D	CPU	(1)
			bdma_ch1_it	WKUP67		D	CPU	(1)
			bdma_ch2_it	WKUP68		D	CPU	(1)
			bdma_ch3_it	WKUP69		D	CPU	(1)
			bdma_ch4_it	WKUP70		D	CPU	(1)
			bdma_ch5_it	WKUP71		D	CPU	(1)
			bdma_ch6_it	WKUP72		D	CPU	(1)
			bdma_ch7_it	WKUP73		D	CPU	(1)
D3	AHB4	DMAMUX2	dmamux2_it	WKUP74			CPU	(1)
D3	AHB4	ADC3	adc3_it	WKUP75		D	CPU	(1)
D3	APB4	SAI4	sai4_gbl_it	WKUP76		D	CPU	(1)
D3	AHB4	HSEM	hsem_int_it	WKUP77		D	CPU	(1)
-	-	NC	NC	WKUP81		-	-	-
D1	APB3	WWDG1	wwdg1_out_rst	WKUP82		C	CPU	(1)
-	-	NC	NC	WKUP83		-	-	-
D1	APB1	CEC	cec_wkup	WKUP85		C	CPU	-
D2	AHB1	ETH	eth	WKUP86		C	CPU	-
D3	AHB4	RCC	hse_css_rcc_wkup	WKUP87		D	CPU	-

1. The source peripheral needs its bus clock in order to generate the event. This is either PCLK4 or HCLK4 in D3 domain, PCLK3 in D1 domain.
2. The source peripheral signal is not connected to the NVIC.

The Extended Interrupt and Event Controller (EXTI) module event inputs able to wake up the D3 domain for autonomous Run mode have a pending request logic that can be cleared by 4 different input sources (*Table 94*). Refer to *Section 19: Extended interrupt and event controller (EXTI)* for further details.

Table 94. EXTI pending requests clear inputs

Source				Destination				Comment
Domain	**Bus**	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**	**Domain**	
D3	AHB4	DMAMUX2	dmamux2_evt6	d3_pendclear_in[0]	EXTI	APB4	D3	-
			dmamux2_evt7	d3_pendclear_in[1]				-
	APB4	LPTIM4	lptim4_out	d3_pendclear_in[2]				-
		LPTIM5	lptim5_out	d3_pendclear_in[3]				-

12.3 DMA

In D1 domain, the MDMA allows the memory to transfer data. It can be triggered by software or by hardware, according to the connections described in *Section 12.3.1*.

DMA Multiplexer in D2 domain (DMAMUX1) allows to map any peripheral DMA request to any stream of the DMA1 or the DMA2. In addition to this, The DMAMUX provides two other functionalities:

- It's possible to synchronize a peripheral DMA request with a timer, with an external pin or with a DMA transfer complete of another stream.
- DMA requests can be generated on a stream by the DMAMUX1 itself. This event can be triggered by a timer, by an external pin event, or by a DMA transfer complete of another stream. The number of DMA requests generated is configurable.

The connections on DMAMUX1 and DMA1/DMA2 are described in *Section 16: DMA request multiplexer (DMAMUX)*, *Section 14: Direct memory access controller (DMA)* and *Section 15: Basic direct memory access controller (BDMA)*.

DMA Multiplexer in D3 domain (DMAMUX2) has the same functionality of DMAMUX1, it is connected to the basic DMA (BDMA).

The connections on DMAMUX2 and BDMA are described in *Section 12.3.3: DMAMUX2, BDMA (D3 domain)*. Refer to *Section 12.3.3: DMAMUX2, BDMA (D3 domain)* and *Section 15: Basic direct memory access controller (BDMA)* for more details.

12.3.1 MDMA (D1 domain)

Table 95. MDMA

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D2	AHB1	DMA1	dma1_tcif0	mdma_str0	MDMA	AXI	D1	DMA1 stream 0 transfer complete
			dma1_tcif1	mdma_str1				DMA1 stream 1 transfer complete
			dma1_tcif2	mdma_str2				DMA1 stream 2 transfer complete
			dma1_tcif3	mdma_str3				DMA1 stream 3 transfer complete
			dma1_tcif4	mdma_str4				DMA1 stream 4 transfer complete
			dma1_tcif5	mdma_str5				DMA1 stream 5 transfer complete flag
			dma1_tcif6	mdma_str6				DMA1 stream 6 transfer complete
			dma1_tcif7	mdma_str7				DMA1 stream 7 transfer complete
D2	AHB1	DMA2	dma2_tcif0	mdma_str8				DMA2 stream 0 transfer complete
			dma2_tcif1	mdma_str9				DMA2 stream 1 transfer complete
			dma2_tcif2	mdma_str10				DMA2 stream 2 transfer complete
			dma2_tcif3	mdma_str11				DMA2 stream 3 transfer complete
			dma2_tcif4	mdma_str12				DMA2 stream 4 transfer complete
			dma2_tcif5	mdma_str13				DMA2 stream 5 transfer complete
			dma2_tcif6	mdma_str14				DMA2 stream 6 transfer complete
			dma2_tcif7	mdma_str15				DMA2 stream 7 transfer complete
D1	APB3	LTDC	ltdc_li_it	mdma_str16				LTDC line interrupt

Table 95. MDMA (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D1	AHB3	JPEG	jpeg_ift_trg	mdma_str17	MDMA	AXI	D1	JPEG input FIFO threshold
			jpeg_ifnt_trg	mdma_str18				JPEG input FIFO not full
			jpeg_oft_trg	mdma_str19				JPEG output FIFO threshold
			jpeg_ofne_trg	mdma_str20				JPEG output FIFO not empty
			jpeg_oec_trg	mdma_str21				JPEG end of conversion
D1	AHB3	QUADSPI	quadspi_ft_trg	mdma_str22				QUADSPI FIFO threshold
			quadspi_tc_trg	mdma_str23				QUADSPI transfer complete
D1	AHB3	DMA2D	dma2d_clut_trg	mdma_str24				DMA2D CLUT transfer complete
			dma2d_tc_trg	mdma_str25				DMA2D transfer complete
			dma2d_tw_trg	mdma_str26				DMA2D transfer watermark
D1	AHB3	SDMMC1	sdmmc1_ dataend_trg	mdma_str29				End of data

12.3.2 DMAMUX1, DMA1 and DMA2 (D2 domain)

Table 96. DMAMUX1, DMA1 and DMA2 connections(1)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D3	AHB4	dmamux1 internal (Request generator)		dmamux1_req_in1	DMAMUX1	AHB1	D2	Requests
				dmamux1_req_in2				
				dmamux1_req_in3				
				dmamux1_req_in4				
				NC				
				NC				
				NC				
				NC				
D2	AHB1	ADC1	adc1_dma	dmamux1_req_in9				
D2	AHB1	ADC2	adc2_dma	dmamux1_req_in10				
D2	APB2	TIM1	tim1_ch1_dma	dmamux1_req_in11				
			tim1_ch2_dma	dmamux1_req_in12				
			tim1_ch3_dma	dmamux1_req_in13				
			tim1_ch4_dma	dmamux1_req_in14				
			tim1_up_dma	dmamux1_req_in15				
			tim1_trig_dma	dmamux1_req_in16				
			tim1_com_dma	dmamux1_req_in17				
D2	APB1	TIM2	tim2_ch1_dma	dmamux1_req_in18				
			tim2_ch2_dma	dmamux1_req_in19				
			tim2_ch3_dma	dmamux1_req_in20				
			tim2_ch4_dma	dmamux1_req_in21				
			tim2_up_dma	dmamux1_req_in22				
D2	APB1	TIM3	tim3_ch1_dma	dmamux1_req_in23				
			tim3_ch2_dma	dmamux1_req_in24				
			tim3_ch3_dma	dmamux1_req_in25				
			tim3_ch4_dma	dmamux1_req_in26				
			tim3_up_dma	dmamux1_req_in27				
			tim3_trig_dma	dmamux1_req_in28				
D2	APB1	TIM4	tim4_ch1_dma	dmamux1_req_in29				
			tim4_ch2_dma	dmamux1_req_in30				
			tim4_ch3_dma	dmamux1_req_in31				
			tim4_up_dma	dmamux1_req_in32				

Table 96. DMAMUX1, DMA1 and DMA2 connections(1) (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D2	APB1	I2C1	i2c1_rx_dma	dmamux1_req_in33	DMAMUX1	AHB1	D2	Requests
			i2c1_tx_dma	dmamux1_req_in34				
D2	APB1	I2C2	i2c2_rx_dma	dmamux1_req_in35				
			i2c2_tx_dma	dmamux1_req_in36				
D2	APB2	SPI1	spi1_rx_dma	dmamux1_req_in37				
			spi1_tx_dma	dmamux1_req_in38				
D2	APB1	SPI2	spi2_rx_dma	dmamux1_req_in39				
			spi2_tx_dma	dmamux1_req_in40				
D2	APB2	USART1	usart1_rx_dma	dmamux1_req_in41				
			usart1_tx_dma	dmamux1_req_in42				
D2	APB1	USART2	usart2_rx_dma	dmamux1_req_in43				
			usart2_tx_dma	dmamux1_req_in44				
D2	APB1	USART3	usart3_rx_dma	dmamux1_req_in45				
			usart3_tx_dma	dmamux1_req_in46				
D2	APB2	TIM8	tim8_ch1_dma	dmamux1_req_in47				
			tim8_ch2_dma	dmamux1_req_in48				
			tim8_ch3_dma	dmamux1_req_in49				
			tim8_ch4_dma	dmamux1_req_in50				
			tim8_up_dma	dmamux1_req_in51				
			tim8_trig_dma	dmamux1_req_in52				
			tim8_com_dma	dmamux1_req_in53				
-	-	NC	NC	NC				
D1	APB1	TIM3	tim5_ch1_dma	dmamux1_req_in55				
			tim5_ch2_dma	dmamux1_req_in56				
			tim5_ch3_dma	dmamux1_req_in57				
			tim5_ch4_dma	dmamux1_req_in58				
			tim5_up_dma	dmamux1_req_in59				
			tim5_trig_dma	dmamux1_req_in60				
D2	APB1	SPI3	spi3_rx_dma	dmamux1_req_in61				
			spi3_tx_dma	dmamux1_req_in62				
D1	APB1	UART4	uart4_rx_dma	dmamux1_req_in63				
			uart4_tx_dma	dmamux1_req_in64				

Table 96. DMAMUX1, DMA1 and DMA2 connections(1) (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D1	APB1	UART5	uart5_rx_dma	dmamux1_req_in65	DMAMUX1	AHB1	D2	Requests
			uart5_tx_dma	dmamux1_req_in66				
D2	APB1	DAC1	dac_ch1_dma	dmamux1_req_in67				
D2	APB1	DAC2	dac_ch2_dma	dmamux1_req_in68				
D2	APB1	TIM6	tim6_up_dma	dmamux1_req_in69				
D2	APB1	TIM7	tim7_up_dma	dmamux1_req_in70				
D2	APB2	USART6	usart6_rx_dma	dmamux1_req_in71				
			usart6_tx_dma	dmamux1_req_in72				
D2	APB1	I2C3	i2c3_rx_dma	dmamux1_req_in73				
			i2c3_tx_dma	dmamux1_req_in74				
D2	AHB2	DCMI	dcmi_dma	dmamux1_req_in75				
D2	AHB2	CRYP	cryp_in_dma	dmamux1_req_in76				
			cryp_out_dma	dmamux1_req_in77				
D2	AHB2	HASH	hash_in_dma	dmamux1_req_in78				
D2	APB1	UART7	uart7_rx_dma	dmamux1_req_in79				
			uart7_tx_dma	dmamux1_req_in80				
D2	APB1	UART8	uart8_rx_dma	dmamux1_req_in81				
			uart8_tx_dma	dmamux1_req_in82				
D2	APB2	SPI4	spi4_rx_dma	dmamux1_req_in83				
			spi4_tx_dma	dmamux1_req_in84				
D2	APB2	SPI5	spi5_rx_dma	dmamux1_req_in85				
			spi5_tx_dma	dmamux1_req_in86				
D2	APB2	SAI1	sai1_a_dma	dmamux1_req_in87				
			sai1_b_dma	dmamux1_req_in88				
D2	APB2	SAI2	sai2_a_dma	dmamux1_req_in89				
			sai2_b_dma	dmamux1_req_in90				
D2	APB1	SWPMI	swpmi_rx_dma	dmamux1_req_in91				
			swpmi_tx_dma	dmamux1_req_in92				
D2	APB1	SPDIFRX	spdifrx_dt_dma	dmamux1_req_in93				
			spdifrx_cs_dma	dmamux1_req_in94				

Table 96. DMAMUX1, DMA1 and DMA2 connections(1) (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D2	APB2	HRTIM1	hrtim_dma1	dmamux1_req_in95	DMAMUX1	AHB1	D2	Requests
			hrtim_dma2	dmamux1_req_in96				
			hrtim_dma3	dmamux1_req_in97				
			hrtim_dma4	dmamux1_req_in98				
			hrtim_dma5	dmamux1_req_in99				
			hrtim_dma6	dmamux1_req_in100				
D2	APB2	DFSDM1	dfsdm1_dma0	dmamux1_req_in101				
			dfsdm1_dma1	dmamux1_req_in102				
			dfsdm1_dma2	dmamux1_req_in103				
			dfsdm1_dma3	dmamux1_req_in104				
D2	APB2	TIM15	tim15_ch1_dma	dmamux1_req_in105				
			tim15_up_dma	dmamux1_req_in106				
			tim15_trig_dma	dmamux1_req_in107				
			tim15_com_dma	dmamux1_req_in108				
D2	APB2	TIM16	tim16_ch1_dma	dmamux1_req_in109				
			tim16_up_dma	dmamux1_req_in110				
D2	APB2	TIM17	tim17_ch1_mda	dmamux1_req_in111				
			tim17_up_dma	dmamux1_req_in112				
D2	APB2	SAI3	sai3_a_dma	dmamux1_req_in113				
			sai3_b_dma	dmamux1_req_in114				
D3	AHB4	ADC3	adc3_dma	dmamux1_req_in115				
D2	AHB1	DMAMUX1	dmamux1_evt0	dmamux1_gen0	DMAMUX1	AHB1	D2	Request generation
			dmamux1_evt1	dmamux1_gen1				
			dmamux1_evt2	dmamux1_gen2				
D2	APB1	LPTIM1	lptim1_out	dmamux1_gen3				
D2	APB1	LPTIM2	lptim2_out	dmamux1_gen4				
D2	APB1	LPTIM3	lptim3_out	dmamux1_gen5				
D3	APB4	EXTI	exti_exti0_it	dmamux1_gen6				
D2	APB1	TIM12	tim12_trgo	dmamux1_gen7				

Table 96. DMAMUX1, DMA1 and DMA2 connections(1) (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D2	AHB1	DMAMUX1	dmamux1_evt0	dmamux1_trg0	DMAMUX1	AHB1	D2	Triggers
			dmamux1_evt1	dmamux1_trg1				
			dmamux1_evt2	dmamux1_trg2				
D2	APB1	LPTIM1	lptim1_out	dmamux1_trg3				
D2	APB1	LPTIM2	lptim2_out	dmamux1_trg4				
D2	APB1	LPTIM3	lptim3_out	dmamux1_trg5				
D3	APB4	EXTI	exti_exti0_it	dmamux1_trg6				
D2	APB1	TIM12	tim12_trgo	dmamux1_trg7				
D2	AHB1	DMAMUX1	dmamux1_req_out0	dma1_str0	DMA1	AHB1	D2	Requests out
			dmamux1_req_out1	dma1_str1				
			dmamux1_req_out2	dma1_str2				
			dmamux1_req_out3	dma1_str3				
			dmamux1_req_out4	dma1_str4				
			dmamux1_req_out5	dma1_str5				
			dmamux1_req_out6	dma1_str6				
			dmamux1_req_out7	dma1_str7				
			dmamux1_req_out8	dma2_str0	DMA2	AHB1	D2	
			dmamux1_req_out9	dma2_str1				
			dmamux1_req_out10	dma2_str2				
			dmamux1_req_out11	dma2_str3				
			dmamux1_req_out12	dma2_str4				
			dmamux1_req_out13	dma2_str5				
			dmamux1_req_out14	dma2_str6				
			dmamux1_req_out15	dma2_str7				

1. The “-” symbol in grayed cells means no interconnect.

12.3.3 DMAMUX2, BDMA (D3 domain)

Table 97. DMAMUX2 and BDMA connections

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D3	AHB4	dmamux2 internal (Request generator)		dmamux2_req_in1	DMAMUX2	AHB4	D3	Requests
				dmamux2_req_in2				
				dmamux2_req_in3				
				dmamux2_req_in4				
				NC				
				NC				
				NC				
				NC				
D3	APB4	LPUART	dma_rx_lpuart	dmamux2_req_in9				
			dma_tx_lpuart	dmamux2_req_in10				
D3	APB4	SPI6	dma_rx_spi6	dmamux2_req_in11				
			dma_tx_spi6	dmamux2_req_in12				
D2	APB1	I2C4	dma_rx_i2c4	dmamux2_req_in13				
			dma_tx_i2c4	dmamux2_req_in14				
D3	APB4	SAI4	dma_a_sai4	dmamux2_req_in15				
			dma_b_sai4	dmamux2_req_in16				
D3	APB4	ADC3	dma_adc3	dmamux2_req_in17				

Table 97. DMAMUX2 and BDMA connections (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D3	AHB4	DMAMUX2	dmamux2_evt0	dmamux2_gen0	DMAMUX2	AHB4	D3	Request generation
			dmamux2_evt1	dmamux2_gen1				
			dmamux2_evt2	dmamux2_gen2				
			dmamux2_evt3	dmamux2_gen3				
			dmamux2_evt4	dmamux2_gen4				
			dmamux2_evt5	dmamux2_gen5				
			dmamux2_evt6	dmamux2_gen6				
D3	APB4	EXTI	exti_lpuart_rx_it	dmamux2_gen7				
			exti_lpuart_tx_it	dmamux2_gen8				
			exti_lptim2_wkup	dmamux2_gen9				
			exti_lptim2_out	dmamux2_gen10				
			exti_lptim3_wkup	dmamux2_gen11				
			exti_lptim3_out	dmamux2_gen12				
			exti_lptim4_wkup	dmamux2_gen13				
			exti_lptim5_wkup	dmamux2_gen14				
			exti_i2c4_wkup	dmamux2_gen15				
			exti_spi6_wkup	dmamux2_gen16				
			exti_comp1_out	dmamux2_gen17				
			exti_comp2_out	dmamux2_gen18				
			exti_rtc_wkup	dmamux2_gen19				
			exti_syscfg_exti0	dmamux2_gen20				
			exti_syscfg_exti2	dmamux2_gen21				
D3	APB4	I2C4	it_evt_i2c4	dmamux2_gen22				
D3	APB4	SPI6	it_spi6	dmamux2_gen23				
D3	APB4	LPUART	it_tx_lpuart1	dmamux2_gen24				
			it_rx_lpuart1	dmamux2_gen25				
D3	AHB4	ADC3	it_adc3	dmamux2_gen26				
			out_awd1_adc3	dmamux2_gen27				
D3	AHB4	BDMA	it_ch0_bdma	dmamux2_gen28				
			it_ch1_bdma	dmamux2_gen29				

Table 97. DMAMUX2 and BDMA connections (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
D3	AHB4	DMAMUX2	dmamux2_evt0	dmamux2_trg0	DMAMUX2	AHB4	D3	Triggers
			dmamux2_evt1	dmamux2_trg1				
			dmamux2_evt2	dmamux2_trg2				
			dmamux2_evt3	dmamux2_trg3				
			dmamux2_evt4	dmamux2_trg4				
			dmamux2_evt5	dmamux2_trg5				
D3	APB4	EXTI	it_exti_tx_lpuart1	dmamux2_trg6				
			it_exti_rx_lpuart1	dmamux2_trg7				
			it_exti_out_lptim2	dmamux2_trg8				
			it_exti_out_lptim3	dmamux2_trg9				
			it_exti_wkup_i2c4	dmamux2_trg10				
			it_exti_wkup_spi6	dmamux2_trg11				
			it_exti_out_comp1	dmamux2_trg12				
			it_exti_wkup_rtc	dmamux2_trg13				
			it_exti_exti0_syscfg	dmamux2_trg14				
			it_exti_exti2_syscfg	dmamux2_trg15				
D3	AHB4	DMAMUX2	dmamux1_req_out0	bdma_ch0	BDMA	AHB4	D3	Requests out
			dmamux1_req_out1	bdma_ch1				
			dmamux1_req_out2	bdma_ch2				
			dmamux1_req_out3	bdma_ch3				
			dmamux1_req_out4	bdma_ch4				
			dmamux1_req_out5	bdma_ch5				
			dmamux1_req_out6	bdma_ch6				
			dmamux1_req_out7	bdma_ch7				

13 MDMA controller (MDMA)

13.1 MDMA introduction

The master direct memory access (MDMA) is used in order to provide high-speed data transfer between memory and memory or between peripherals and memory. Data can be quickly moved by the MDMA without any CPU action. This keeps the CPU resources free for other operations.

The MDMA controller provides a master AXI interface for main memory and peripheral registers access (system access port) and a master AHB interface only for Cortex-M7 TCM memory access (TCM access port).

The MDMA works in conjunction with the standard DMA controllers (DMA1 or DMA2). It offers up to 16 channels, each dedicated to manage memory access requests from one of the DMA stream memory buffer or other peripherals (w/ integrated FIFO).

13.2 MDMA main features

- AXI/AHB master bus architecture, one dedicated to main memory/peripheral accesses and one dedicated to Cortex-M7 AHBS port (only for TCM accesses).
- 16 channels
- Up to 32 hardware trigger sources
- Each channel request can be selected among any of the request sources. This selection is software-configurable and allows several peripherals to initiate DMA requests. The trigger selection can be automatically changed at the end of one block transfer.
- All the channels are identical and can be connected either to a standard DMA or a peripheral request (acknowledge by data read/write) system
- Each channel also supports software trigger
- One 256-level memory buffer, split in two 128-level first-in, first-out (FIFO), that will be used to store temporary the data to be transferred (in burst or single transfer mode), for 1 or 2 consecutive buffers. The FIFO will store the data that will be transferred during the current channel block transfer (up to the block transfer size). The 2nd FIFO can be used for the next buffer to be transferred, either for the same channel or for the next channel transfer.
- The priorities between the DMA channels are software-programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (channel 0 has priority over channel 1, etc.)
- Independent source and destination transfer width (byte, half-word, word, double-word): when the data widths of the source and destination are not equal, the MDMA can pack/unpack the necessary data to optimize the bandwidth.
- The size and address increment for both source and destination can be independently selected.

Note: Based on this separation, some more advanced packing/unpacking operations are available at software level. As an example, 2 x 16-bit data blocks can be interleaved together using two MDMA channels, in the destination memory, by simply programming the 2 channels with an increment step of 4 bytes and a data size of 16-bit + a start address shifted by 2 between the 2 channels.

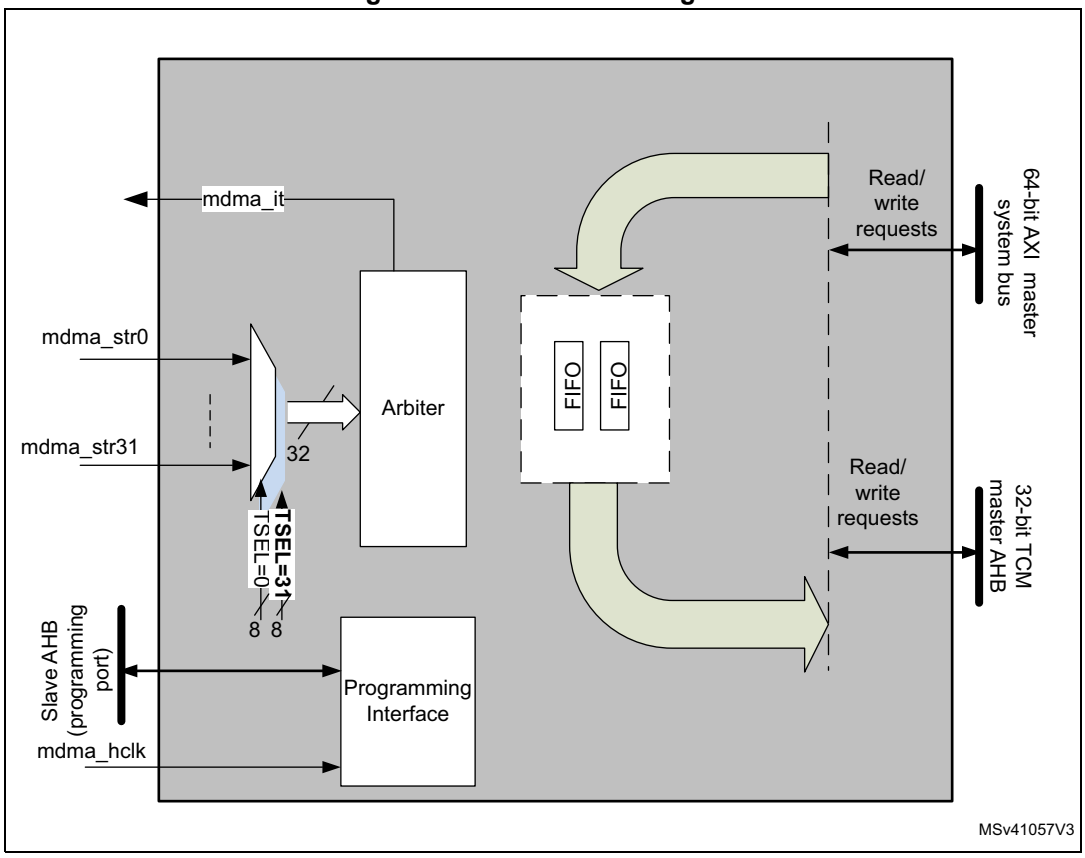
- Incrementing, decrementing or non incrementing/fixed addressing for source and destination
- Data packing/unpacking is always done respecting the little endian convention: lower address in a data entity (double word, word or half word) contains always the lowest significant byte. This is independent of the address increment/decrement mode of both source and destination.
- Supports incremental burst transfers. The size of the burst is software-configurable, up to 128 bytes. For larger data sizes the burst length is limited, as to respect the maximum 128 bytes data burst size (e.g. 16x64-bit or 32x32-bit).
- For the TCM memory accesses, the burst access is only allowed when the increment and data size are identical and lower than or equal to 32-bit.
- 5 event flags (MDMA Channel Transfer Complete, MDMA Block Transfer complete, MDMA Block Repeat Transfer Complete, MDMA buffer transfer Complete, MDMA Transfer Error) are available and can generate interrupts.

13.3 MDMA functional description

13.3.1 MDMA block diagram

Figure 73 shows the block diagram of the MDMA.

Figure 73. MDMA block diagram



13.3.2 MDMA internal signals

Table 98 shows the internal MDMA signals.

Table 98. MDMA internal input/output signals

Signal name	Signal type	Description
mdma_hclk	Digital input	MDMA AHB clock
mdma_it	Digital output	MDMA interrupt
mdma_str[0:31]	Digital input	MDMA stream request

13.3.3 MDMA overview

The MDMA controller performs a direct memory transfer: as an AXI/AHB master, it can take the control of the AXI/AHB bus matrix to initiate AXI/AHB transactions.

It can carry out the following transactions:

- memory-to-memory (software triggered)
- peripheral-to-memory
- memory-to-peripheral

For the last two transaction types, the memory can also be replaced by a memory-mapped peripheral, which has no control over the MDMA flow. When these types of transaction are used and the request is coming from a standard DMA (DMA1 or DMA2), the peripheral register access is replaced by a memory access to the memory buffer used by this DMA.

Note: Non-incrementing/decrementing mode will not be used for memory accesses.

The source and destination are simply defined by the address (peripherals being memory mapped also).

The AHB slave port is used to program the MDMA controller (it supports 8/16/32-bit accesses).

The size of the data array to be transferred for a single request will be one of the following:

1. The buffer transfer size
2. The block size
3. Repeated block
4. Complete channel data (until the linked list pointer for the channel is null)

The choice of the size is done through the TRGM[1:0] (Trigger mode) selection field.

The user must choose one of them based on the data array size available (usually in the DMA1/2 memory buffer) and the “real time” requirements for other MDMA channels (knowing that a buffer transfer is the minimum data aggregate to be transferred by the MDMA without doing a new arbitration between MDMA channel requests).

For each channel, there are three key data array sizes:

1. Burst size: this is the length of the data transfer which can be performed in burst mode. This burst length defines the maximum transfer length which cannot be interrupted at bus arbitration level and can block other masters from accessing the bus)
2. Buffer transfer size: this is the length of the data array to be transferred, on a channel, before checking for MDMA requests on other channels. This is the data array transfer lengths which cannot be interrupted at MDMA level (from other channel requests).
3. Block size: this value has two meanings which can be used together:
   a) main: this is length of the data block which is described in a block structure of the MDMA linked list (corresponds to one entry in the linked list)
   b) selectable: when TRGM[1:0] equals 01, this is the length of the data array which is transferred on a single MDMA request activation (for the respective channel)

13.3.4 MDMA channel

Each of the DMA controller channel provides a unidirectional transfer link between a source and a destination.

Each channel can perform:

- Single block transfer: one block is transferred. At the end of the block, the DMA channel is disabled and an End of Channel Transfer interrupt is generated.
- Repeated block transfer: a number of blocks is transferred before disabling the channel
- Linked list transfer: when the transfer of the current data block (or last block in a repeat) is completed, a new block control structure is loaded from memory and a new block transfer is started.

The minimum amount of data to be transferred for each request (buffer size, up to 128-bytes) is programmable. The total amount of data in a block, is programmable up to 64 Kbytes. This value is decremented after each transfer. When this counter reaches 0, the end of the block is reached and an action is taken based on the repeat counter (for repeated block transfer) and/or linked list structure value.

Note: If the block length is not a multiple of the buffer length, the last buffer transfer in the block will be shorter, covering the remaining bytes to be transferred in the current block.

If the link structure address points to a valid memory address, the MDMA will reload the whole channel descriptor structure register contents from memory at this address. Then, a new block transfer will then be executed (on the next MDMA channel request) based on this information.

If the link structure address is 0x0, at the end of the current/repeated block transfer, the MDMA channel will be disabled and the end of channel transfer interrupt will be generated.

13.3.5 Source, destination and transfer modes

Both the source and destination transfers can address peripherals and memories in the entire 4-Gbyte area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The source/destination addresses can be fixed (e.g. FIFO/single data register peripherals) or incremented/decremented. The transfer can be done in single access or in burst mode (programmable).

13.3.6 Pointer update

The source and destination memory pointers can optionally be automatically post-incremented/decremented or kept constant after each transfer depending on the SINC[1:0] and DINC[1:0] bits in the MDMA_CxCR register.

Disabling the increment mode is useful when the peripheral source or destination data are accessed through a single register/FIFO mode.

If the increment/decrement mode is enabled, the address of the next data transfer will be the address of the previous one incremented/decrement by 1, 2, 4 or 8 depending on the increment size programmed in the SINCOS[1:0] or DINCOS[1:0] bits in the MDMA_CxCR register.

In order to optimize the packing operation, the increment offset size and the data size are programmable independently.

13.3.7 MDMA buffer transfer

This is the minimum logical amount of data (up to 128 bytes) which is transferred on an MDMA request event, on one channel.

An MDMA buffer transfer consists of a sequence of a given number of data transfers (done as single or burst data transfers). The number of data items to be transferred and their width (8-bit, 16-bit, 32-bit or 64-bit) are software programmable. The length of the burst used for data transfers is also programmable, independently.

After an event requiring a data array to be transferred, the DMA/peripheral sends a request signal to the MDMA controller. The MDMA controller serves the request depending on the channel priorities.

The request is acknowledged by writing the mask data value to the address given mask address, when these registers are set.

If the mask address register is not set (0x00 value), the request can be reset by simply reading/writing the data to the peripheral. In this case, if the request is done by a destination peripheral, the write must be set as non bufferable, in order to avoid a false new MDMA request.

The total amount of data to be transferred, on the current channel, following a MDMA request, is determined by the TRGM[1:0] field.

If TRGM[1:0] equals 00, a single buffer will be transferred, then the MDMA will wait for another request on the same channel.

Note: In this case, the hardware request for the currently active channel (data in the FIFO) will not be considered again until the end of the write phase for this channel. In this case, even if the channel would still be active at the end of the read phase, another channel (even with lower priority) could start the read phase. Because of this, lower priority channels can be interleaved with current channel transfer.

If TRGM[1:0] is different from 00 (multiple buffers need to be transferred), the mdma_strx for the current channel remains active (internally memorized) until the whole transfer defined by TRGM (block, repeated block or whole channel/linked list data) is completed. However, after transferring an individual buffer, the MDMA will enter in a new arbitration phase (between new external requests and internally memorized ones). If no other higher priority, channel request is active, a new buffer transfer will be started for the same channel.

Note: When TRGM[1:0] is different from 00, a larger array of data will be transferred for a single request. But, as the channel arbitration is done after each buffer transfer, no higher level MDMA requests would be blocked for the more than a buffer transfer period, on any lower priority channel.

13.3.8 Request arbitration

An arbiter manages the MDMA channel requests based on their priority. When MDMA is idle and after the end of each buffer transfer, all MDMA requests (hardware or software) are checked for all enabled channels.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the MDMA_CxCR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: at hardware level, the channel priority is fixed. If two requests have the same software priority level, the channel with the lower number takes priority over the stream with the higher number. For example, Channel 2 takes priority over Channel 4 when they have the same software priority level.

13.3.9 FIFO

A FIFO structure is used to temporarily store data coming from the source before writing them to the destination. There is a central FIFO structure which is used for all channels.

In order to maximize data bandwidth and bus usage, the following mechanisms are used, allowing multiple read/write operation to be executed in parallel.

- During a buffer transfer, as soon as the FIFO contains enough data for a destination burst transfer, the write operation will start.
- When the complete data for a buffer transfer has been read into the FIFO, the arbitration procedure will be started. Following that, the next buffer data to be transferred can be read to the FIFO.

When an active channel is disabled due to an error, during a buffer transfer, the remaining data in the internal FIFO will be discarded.

13.3.10 Block transfer

A block is a “contiguous” array of data, up to 64 Kbytes, which is transferred by successive buffer transfers.

Each block of data is defined by the start address and the block length. When a block transfer is completed, one of the following three actions can be executed:

- The block is part of a repeated block transfer: the block length is reloaded and new block start address is computed (based on the information in the MDMA_CxBRUR register)
- It is a single block or the last block in a repeated block transfer: the next block information is loaded from the memory (using the linked list address information, from the MDMA_CxLAR)
- It is the last block which needs to be transferred for the current MDMA channel (MDMA_CxLAR = 0): the channel is disabled and no further MDMA requests will be accepted for this channel

13.3.11 Block repeat mode

The block repeat mode allows to repeat a block transfer, with different start addresses for source and destination.

When the repeat block mode is active (repeat counter non 0), at the end of the current block transfer, the block parameters will be updated (the BNDT value reloaded and SAR/DAR values updated according to BRSUM/BRDUM configuration), and the repeat counter decremented by 1.

When the repeat block counter reaches 0, this last block will be treated as a single block transfer.

13.3.12 Linked list mode

The Linked list mode allows to load a new MDMA configuration (CxTCR, CxBNDTR, CxSAR, CxDAR, CxBRUR, CxLAR, CxTBR, CxMAR and CxMDR registers), from the address given in the CxLAR register. This address must address a memory mapped on the AXI system bus.

Following this operation, the channel is ready to accept new requests, as defined in the block/repeated block modes above, or continue the transfer if TRGM[1:0] equals 11.

The trigger source can be automatically changed, when loading the CxTBR value.

The TRGM and SWRM values must not be changed when TRGM[1:0] equals 11.

13.3.13 MDMA transfer completion

Different events can generate an end of transfer by setting the CTCIF bit in the status register (MDMA_CxISR):

- The MDMA_CxBNDTR counter has reached zero, the Block Repeat Counter is 0 and the Link list pointer address is 0
- The channel is disabled before the end of transfer (by clearing the EN bit in the MDMA_CxCR register) and all the remaining data have been transferred from the FIFO to the destination

13.3.14 MDMA transfer suspension

At any time, a MDMA transfer can be suspended in order to be to be restarted later on or to be definitively disabled before the end of the MDMA transfer.

There are two cases:

- The channel is disabled, with no later-on restart from the point where it was stopped. There is no particular action to do, besides clearing the EN bit in the MDMA_CxCR register to disable the channel. The stream can take time to be disabled (on going buffer transfer is completed first). The transfer complete interrupt flag is set in order to indicate the end of transfer. The value of the EN bit in MDMA_CxCR is now 0 to confirm the channel interruption. The MDMA_CxNDTR register contains the number of remaining data items at the moment when the channel was stopped so that the software can determine how many data items have been transferred before the channel was interrupted.
- The channel is suspended before the number of remaining bytes to be transferred in the MDMA_CxBNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the channel. The channel transfer complete interrupt flag CTCIF is set in order to indicate the end of transfer. If the MDMA_CxBNDTR, SAR and DAR registers are not modified by software, the transfer will continue when the channel is re-enabled. CTCIF must also be reset before restarting the channel.

Note: If the completed buffer is the last of the block, the configuration registers are also updated before disabling the channel, in order to be correctly prepared for a soft restart.

Note: Before reprogramming the channel, software must wait the CTCIF register is set, in order to guarantee that any ongoing operation has been completed.

13.3.15 Error management

The MDMA controller can detect the following errors:

The transfer error interrupt flag (TEIF) is set when:

- A bus error occurs during a MDMA read or a write access
- The address alignment does not correspond to the data size
- The block size is not a multiple of the data size (for source and/or destination): this error is activated on the last transfer and the error address points to the last transfer (which cannot be done)

13.4 MDMA interrupts

For each MDMA channel, an interrupt can be produced on the following events:

- Channel Transfer Completed
- Block-Transfer Completed
- Block-Transfer Repeat Completed
- buffer Transfer Completed
- Transfer Error

Separate interrupt enable control bits are available for flexibility as shown in *Table 99*.

Table 99. MDMA interrupt requests

Interrupt event	Event flag	Enable control bit
Channel Transfer Completed	CTCIF	CTCIE
Block-Transfer Repeat completed	BTRIF	BTRIE
Block-Transfer completed	BTIF	BTIE
buffer Transfer Completed	TCIF	TCIE
Transfer Error	TEIF	TEIE

Note: Before setting an Enable control bit to 1, the corresponding event flag should be cleared, otherwise an interrupt might be immediately generated, if the bit is already set.

Note: When at least one interrupt flag and the respective enable control bit are set, the channel interrupt bit is set in the GISR. The Interrupt output is also activated. This will generate an interrupt if the respective interrupt channel is enabled in the NVIC.

13.5 MDMA registers

The MDMA registers can be accessed in word/half-word or byte format.

13.5.1 MDMA global interrupt/status register (MDMA_GISR0)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GIF15	GIF14	GIF13	GIF12	GIF11	GIF10	GIF9	GIF8	GIF7	GIF6	GIF5	GIF4	GIF3	GIF2	GIF1	GIF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **GIF[15:0]**: Channel x global interrupt flag (x=0..15)

This bit is set and reset by hardware. It is a logical OR of all the Channel x interrupt flags (CTCIF, BTIF, BRTIF, TEIF) which are enabled in the interrupt mask register (CTCIEx, BTIEx, BRTIEx, TEIEx)

0: No interrupt generated by channel x

1: Interrupt generated by channel x

13.5.2 MDMA channel x interrupt/status register (MDMA_CxISR)

Address offset: 0x40 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRQA
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCIF	BTIF	BRTIF	CTCIF	TEIF
											r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CRQA**: Channel x ReQuest Active flag

This bit is set by software writing 1 to the SWRQx bit in the MDMA_CxCR register, in order to request a MDMA transfer, and the channel x is enabled.

It is also set by hardware when the channel request become active and the channel is enabled. The hardware request memorized until it is served.

It is cleared by hardware, when the Channel x Request is completed (after the source write phase of the last buffer transfer due for the current request).

0: The MDMA transfer mdma_strx is inactive for channel x.

1: The MDMA transfer mdma_strx is active for channel x

This bit is also reset by hardware when the channel is disabled (in case of transfer error or when reaching the end of the channel data transfer - repeat block = 0 and linked list pointer null - or by software programming the channel enable bit to 0 before that).

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TCIF**: Channel x buffer transfer complete interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.

0: No buffer transfer complete event on channel x

1: A buffer transfer complete event occurred on channel x

TC is set when a single buffer was transferred. It will be activated on each channel transfer request.

This can be used as a debug feature (without interrupt), indicating that (at least) an MDMA buffer transfer had been generated since the last flag reset.

Bit 3 **BTIF**: Channel x block transfer complete interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.

0: No block transfer complete event on channel x

1: A block transfer complete event occurred on channel x

Bit 2 **BRTIF**: Channel x block repeat transfer complete interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.

0: No block repeat transfer complete event on channel x

1: A block repeat transfer complete event occurred on channel x

Bit 1 **CTCIF**: Channel x Channel Transfer Complete interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.

0: No channel transfer complete event on channel x

1: A channel transfer complete event occurred on channel x

CTC is set when the last block was transferred and the channel has been automatically disabled.

CTC is also set when the channel is suspended, as a result of writing EN bit to 0.

Bit 0 **TEIF**: Channel x transfer error interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the MDMA_IFCRy register.

0: No transfer error on stream x

1: A transfer error occurred on stream x

13.5.3 MDMA channel x interrupt flag clear register (MDMA_CxIFCR)

Address offset: 0x44 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLTCIF	CBTIF	CBRTIF	CCTCIF	CTEIF
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CLTCIF**: CLear buffer Transfer Complete Interrupt Flag for channel x

Writing 1 into this bit clears TCIF in the MDMA_ISRy register

Bit 3 **CBTIF**: Channel x Clear block transfer complete interrupt flag

Writing 1 into this bit clears BTIF in the MDMA_ISRy register

Bit 2 **CBRTIF**: Channel x clear block repeat transfer complete interrupt flag

Writing 1 into this bit clears BRTIF in the MDMA_ISRy register

Bit 1 **CCTCIF**: Clear Channel transfer complete interrupt flag for channel x

Writing 1 into this bit clears CTCIF in the MDMA_ISRy register

Bit 0 **CTEIF**: Channel x clear transfer error interrupt flag

Writing 1 into this bit clears TEIF in the MDMA_ISRy register

13.5.4 MDMA channel x error status register (MDMA_CxESR)

Address offset: 0x48 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BSE	ASE	TEMD	TELD	TED	TEA[6:0]						
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BSE**: Block Size Error

These bit is set by hardware, when the block size is not an integer multiple of the data size either for source or destination. TED will indicate whether the problem is on the source or destination.

It is cleared by software writing 1 to the **CTEIF** bit in the MDMA_IFCRy register.

0: No block size error.

1: Programmed block size is not an integer multiple of the data size.

Bit 10 **ASE**: Address/Size Error

These bit is set by hardware, when the programmed address is not aligned with the data size. TED will indicate whether the problem is on the source or destination.

It is cleared by software writing 1 to the **CTEIF** bit in the MDMA_IFCRy register.

0: No address/size error.

1: Programmed address is not coherent with the data size.

Bit 9 **TEMD**: Transfer Error Mask Data

These bit is set by hardware, in case of a transfer error while writing the Mask Data.

It is cleared by software writing 1 to the **CTEIF** bit in the MDMA_IFCRy register.

0: No mask write access error.

1: The last transfer error on the channel was a related to a write of the Mask Data.

Bit 8 **TELD**: Transfer Error Link Data

These bit is set by hardware, in case of a transfer error while reading the block link data structure.

It is cleared by software writing 1 to the **CTEIF** bit in the MDMA_IFCRy register.

0: No link data read access error.

1: The last transfer error on the channel was a related to a read of the Link Data structure.

Bit 7 **TED**: Transfer Error Direction

These bit is set and cleared by hardware, in case of an MDMA data transfer error.

0: The last transfer error on the channel was a related to a read access.

1: The last transfer error on the channel was a related to a write access.

Bits 6:0 **TEA[6:0]**: Transfer Error Address

These bits are set and cleared by hardware, in case of an MDMA data transfer error. It is used in conjunction with TED.

This field indicates the 7 LSBits of the address which generated a transfer/access error.

It can be used by software to retrieve the failing address, by adding this value (truncated to the buffer transfer length size) to the current SAR/DAR value.

Note: The SAR/DAR current value doesn't reflect this last address due to the FIFO management system. The SAR/DAR are only updated at the end of a (buffer) transfer (of TLEN+1 bytes).

Note: It is not set in case of a link data error.

13.5.5 MDMA channel x control register (MDMA_CxCR)

This register is used to control the concerned channel.

Address offset: 0x4C + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWRQ
															w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WEX	HEX	BEX	Res.	Res.	Res.	Res.	PL[1:0]		TCIE	BTIE	BRTIE	CTCIE	TEIE	EN
	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SWRQ**: Software Request

Writing 1 into this bit sets the CRQA in MDMA_ISRy register, activating the request on Channel x

Note: Either the whole CxCR register or the 8-bit/16-bit register at Address offset: 0x4E + 0x40 × channel number can be used for SWRQ activation.

In case of a software request, acknowledge is not generated (neither hardware signal, nor CxMAR write access).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WEX**: Word Endianess exchange

This bit is set and cleared by software.
0: Little endianess preserved for words
1: word order exchanged in double word
When this bit is set, the word order in the destination double word is reversed: higher address word contains the data read from the lower address of the source.
If destination is not a double word, do not care of the value of this bit.
This bit is protected and can be written only if EN is 0.

Bit 13 **HEX**: Half word Endianess exchange

This bit is set and cleared by software.
0: Little endianess preserved for half words
1: half-word order exchanged in each word
When this bit is set, the half-word order in each destination word is reversed: higher address half-word contains the data read from the lower address of the source.
If destination length is shorter than word, do not care of the value of this bit.
This bit is protected and can be written only if EN is 0.

Bit 12 **BEX**: Byte Endianess exchange

This bit is set and cleared by software.
0: Little endianess preserved for bytes
1: byte order exchanged in each half-word
When this bit is set, the byte order in each destination Half Word is reversed: higher address word contains the data read from the lower address of the source.
If destination is byte, do not care of the value of this bit.
This bit is protected and can be written only if EN is 0.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **PL[1:0]**: Priority level

These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high
These bits are protected and can be written only if EN is 0.

Bit 5 **TCIE**: buffer Transfer Complete interrupt enable

This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 4 **BTIE**: Block Transfer interrupt enable

This bit is set and cleared by software.
0: BT complete interrupt disabled
1: BT complete interrupt enabled

Bit 3 **BRTIE**: Block Repeat transfer interrupt enable

This bit is set and cleared by software.
0: BT interrupt disabled
1: BT interrupt enabled

Bit 2 **CTCIE**: Channel Transfer Complete interrupt enable

This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 1 **TEIE**: Transfer error interrupt enable

This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 0 **EN**: Channel enable / flag channel ready when read low

This bit is set and cleared by software.
0: Channel disabled
1: Channel enabled
This bit can be cleared by hardware:

- on a MDMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB/AXI master buses (bus error/hard fault)
- if another error condition is encountered (data alignment, block/data size incompatibility)

When this bit is reset by software, the ongoing buffer transfer (if any) will be completed. All status/configuration registers will keep their current values. If the channel is re enabled without writing these registers, the channel will continue from the point where it was interrupted.

When this bit is read as 0, the software is allowed to program the configuration registers. It is forbidden to write these registers when the EN bit is read as 1 (writes are ignored).

Note: When this bit is reset by software, it is recommended to wait for the CTCIF = 1, in order to ensure that any ongoing buffer transfer has been completed, before reprogramming the channel.

13.5.6 MDMA channel x transfer configuration register (MDMA_CxTCR)

This register is used to configure the concerned channel. In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x00).

Address offset: 0x50 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWM	SWRM	TRGM[1:0]		PAM[1:0]		PKE	TLEN[6:0]							DBURST[2:1]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBURST [0]	SBURST[2:0]			DINCOS[1:0]		SINCOS[1:0]		DSIZE[1:0]		SSIZE[1:0]		DINC[1:0]		SINC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BWM**: Bufferable Write Mode

This bit is set and cleared by software.

0: The destination write operation is non-bufferable.

1: The destination write operation is bufferable.

This bit is protected and can be written only if EN is 0.

Note: All MDMA destination accesses are non-cacheable.

Bit 30 **SWRM**: Software Request Mode

This bit is set and cleared by software. If a hardware or software request is currently active, the bit change will be delayed until the current transfer is completed.

0: hardware request are taken into account: the transfer is initiated as defined by TRGM value and acknowledged by the MDMA ACKx signal.

If the CxMAR contains a valid address, the CxMDR value will also be written at CxMAR address.

1: hardware request are ignored. Transfer is triggered by software writing 1 to the SWRQ bit.

This bit is protected and can be written only if EN is 0.

Bits 29:28 **TRGM[1:0]**: Trigger Mode

These bits are set and cleared by software.

00: Each MDMA request (software or hardware) triggers a buffer transfer

01: Each MDMA request (software or hardware) triggers a block transfer

10: Each MDMA request (software or hardware) triggers a repeated block transfer (if the block repeat is 0, a single block is transferred)

11: Each MDMA request (software or hardware) triggers the transfer of the whole data for the respective channel (e.g. linked list) until the channel reach the end and it is disabled.

Note: If TRGM is 11 for the current block, all the values loaded at the end of the current block through the linked list mechanism must keep the same value (TRGM=11) and the same SWRM value, otherwise the result is undefined.

These bits are protected and can be written only if EN is 0.

Bits 27:26 **PAM[1:0]**: Padding/Alignement Mode
These bits are set and cleared by software.

Case 1: Source data size smaller than destination data size - 3 options are valid.
00: Right Aligned, padded w/ 0s (default)
01: Right Aligned, Sign extended
10: Left Aligned (padded with 0s)
11: Reserved

Case 2: Source data size larger than destination data size.
00: Right Aligned - only the LSBs part of the Source is written to the destination address
10: Left Aligned - only the MSBs part of the Source is written to the destination address
The remainder part is discarded.

When PKE = 1 or DSIZE=SSIZE, these bits are ignored.
These bits are protected and can be written only if EN is 0

Bit 25 **PKE**: Pack Enable
This bit is set and cleared by software.
0: The source data is written to the destination as is.
If the Source Size is smaller than the destination, it will be padded according to the PAM value.
If the Source data size is larger than the destination one, it will be truncated. The alignment will be done according to the PAM[1:0] value.
1: The source data is packed/unpacked into the destination data size. All data are right aligned, in Little Endian mode.
This bit is protected and can be written only if EN is 0

Bits 24:18 **TLEN[6:0]**: buffer Transfer Length (number of bytes - 1)
These bits are set and cleared by software.
The value of TLEN+1 represents the number of bytes to be transferred in a single transfer.
The Transfer Length MUST be a multiple of the data size (for both Source and Destination)

Note: When the source/destination sizes are different and padding/truncation is used, the TLEN+1 refers to the source data array size.

These bits are protected and can be written only if EN is 0
DBURST value must be programmed in order to ensure that the burst size will be lower than the Transfer Size.

Bits 17:15 **DBURST[2:0]**: Destination burst transfer configuration
These bits are set and cleared by software.
000: single transfer
N: burst of 2^N beats
These bits are protected and can be written only if EN is 0
DBURST value must be programmed as to ensure that the burst size will be lower than the Transfer Length. If this is not ensured, the result is unpredictable.

Note: When the destination bus is TCM/AHB (DBUS=1) and DINCOS=11 or DINC=00 or DINCOS/=DSIZE, DBURST must be programmed to 000 (single transfer), else the result is unpredictable.

Note: When the destination bus is system/AXI bus (DBUS=0) and DINC=00, DBURST must be maximum 100 (burst of 16), else the result is unpredictable.

Bits 14:12 **SBURST[2:0]**: Source burst transfer configuration

These bits are set and cleared by software.

000: single transfer

N: burst of 2^N beats

These bits are protected and can be written only if EN is 0

SBURST value must be programmed as to ensure that the burst size will be lower than the transfer length. If this is not ensured, the result is unpredictable.

Note: When the source bus is TCM (SBUS=1) and SINCOS=11 or SINC = 00 or SINCOS/=SSIZE, SBURST must be programmed to 000 (single transfer), else the result is unpredictable.

Note: When the source bus is system/AXI bus (SBUS=0) and SINC=00, SBURST must be maximum 100 (burst of 16), else the result is unpredictable.

Bits 11:10 **DINCOS[1:0]**: Destination increment offset size

These bits are set and cleared by software.

00: byte (8-bit)

01: half-word (16-bit)

10: word (32-bit)

11: Double-Word (64-bit) -

This bits have no meaning if bit DINC[1:0] = '00'.

These bits are protected and can be written only if EN = '0'.

If DINCOS < DSIZE and DINC /= 00, the result will be unpredictable.

If destination is AHB and DBURST =/ 000, destination address must be aligned with DINCOS size, else the result is unpredictable.

Bits 9:8 **SINCOS[1:0]**: Source increment offset size

These bits are set and cleared by software.

00: byte (8-bit)

01: half-word (16-bit)

10: word (32-bit)

11: Double-Word (64-bit) -

This bits have no meaning if bit SINC[1:0] = '00'.

These bits are protected and can be written only if EN = '0'.

If SINCOS < SSIZE and SINC /= 00, the result will be unpredictable.

If source is TCM/AHB and SBURST =/ 000, source address must be aligned with SINCOS size, else the result is unpredictable.

Bits 7:6 **DSIZE[1:0]**: Destination data size

These bits are set and cleared by software.

00: byte (8-bit)

01: half-word (16-bit)

10: word (32-bit)

11: Double-Word (64-bit) -

These bits are protected and can be written only if EN is 0.

Note: If a value of 11 is programmed for the TCM access/AHB port, a transfer error will occur (TEIF bit set)

If DINCOS < DSIZE and DINC /= 00, the result will be unpredictable.

Note: DSIZE = 11 (double-word) is forbidden when destination is TCM/AHB bus (DBUS=1).

Bits 5:4 **SSIZE[1:0]**: Source data size

These bits are set and cleared by software.

00: Byte (8-bit)

01: Half-word (16-bit)

10: Word (32-bit)

11: Double-Word (64-bit)

These bits are protected and can be written only if EN is 0

Note: If a value of 11 is programmed for the TCM access/AHB port, a transfer error will occur (TEIF bit set)

If SINCOS < SSIZE and SINC /= 00, the result will be unpredictable.

Note: SSIZE = 11 (double-word) is forbidden when source is TCM/AHB bus (SBUS=1).

Bits 3:2 **DINC[1:0]**: Destination increment mode

These bits are set and cleared by software.

00: Destination address pointer is fixed

10: Destination address pointer is incremented after each data transfer (increment is done according to DINCOS)

11: Destination address pointer is decremented after each data transfer (increment is done according to DINCOS)

These bits are protected and can be written only if EN is 0

Note: When destination is AHB (DBUS=1), DINC = 00 is forbidden.

Bits 1:0 **SINC[1:0]**: Source increment mode

These bits are set and cleared by software.

00: Source address pointer is fixed

10: Source address pointer is incremented after each data transfer (increment is done according to SINCOS)

11: Source address pointer is decremented after each data transfer (decrement is done according to SINCOS)

These bits are protected and can be written only if EN is 0

Note: When source is AHB (SBUS=1), SINC = 00 is forbidden.

13.5.7 MDMA channel x block number of data register (MDMA_CxBNDTR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x04).

Address offset: 0x54 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRC[11:0]												BRDUM	BRSUM	Res.	BNDT[16]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 **BRC[11:0]**: Block Repeat Count

This field contains the number of repetitions of the current block (0 to 4095). When the channel is enabled, this register is read-only, indicating the remaining number of blocks, excluding the current one. This register decrements after each complete block transfer.

Once the last block transfer has completed, this register can either stay at zero or be reloaded automatically from memory (in Linked List mode - i.e. Link Address valid).

These bits are protected and can be written only if EN is 0.

Bit 19 **BRDUM**: Block Repeat Destination address Update Mode

0: At the end of a Block transfer, the DAR register will be updated by adding the DUV to the current DAR value (current Destination Address)

1: At the end of a block transfer, the DAR register will be updated by subtracting the DUV from the current DAR value (current Destination Address)

These bits are protected and can be written only if EN is 0.

Bit 18 **BRSUM**: Block Repeat Source address Update Mode

0: At the end of a block transfer, the SAR register will be updated by adding the SUV to the current SAR value (current Source Address)

1: At the end of a block transfer, the SAR register will be updated by subtracting the SUV from the current SAR value (current Source Address)

These bits are protected and can be written only if EN is 0.

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **BNDT[16:0]**: Block Number of data bytes to transfer

Number of bytes to be transferred (0 up to 65536) in the current block. When the channel is enabled, this register is read-only, indicating the remaining data items to be transmitted. During the channel activity, this register decrements, indicating the number of data items remaining in the current block.

Once the block transfer has completed, this register can either stay at zero or be reloaded automatically with the previously programmed value if the channel is configured in block Repeat mode.

If the value of this register is zero, no transaction can be served even if the stream is enabled.

These bits are protected and can be written only if EN is 0.

Note: 1: If the BNDT value is not an integer multiple of the TLEN+1 value, the last transfer will be shorter and contain only the remaining data in the Block.

Note: 2: The size of the block must be a multiple of the source and destination data size. If this is not true, an error will be set and the no data will be written.

13.5.8 MDMA channel x source address register (MDMA_CxSAR)

In Linked List mode, at the end of a Block (single or last Block in repeated Block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x08).

Address offset: 0x58 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SAR[31:0]**: Source address

These bits represent the base address of the peripheral data register from/to which the data will be read. They must be aligned with the SSIZE (e.g. SAR[1:0] = 00 when SSIZE=10), but may be unaligned with the SINCOS.

When source is TCM/AHB, if address is not aligned with SINCOS, access must be programmed as single (SBURST=000).

These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

During the channel activity, this register is updated, reflecting the current address from which the data will be read next.

When the block repeat mode is active, when a block transfer is completed, the source address is updated by adding/subtracting the SAU value to the current value (already updated after the last transfer in the block).

When the Linked List mode is active, at the end of a block (repeated or not) transfer, the SAR value will be loaded from memory (from address LSA + m)

13.5.9 MDMA channel x destination address register (MDMA_CxDAR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x0C).

Address offset: 0x5C + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DAR[31:0]**: Destination address

Base address of the destination address to which the data will be written.

These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

Must be aligned with the DSIZE (e.g. DAR[0] = 0 when DSIZE=01), but may be unaligned with the DINCOS.

When destination is AHB, if address is not aligned with DINCOS, access must be programmed as single (DBURST=000).

During the channel activity, this register is updated, reflecting the current address to which the data will be written next.

When the block repeat mode is active, when a block transfer is completed, the Destination address is updated by adding/subtracting the DAU value to the current value (after the last transfer in the block).

When the Linked List mode is active, at the end of a block (repeated or not) transfer, the DAR value will be loaded from memory (from address LSA + m)

13.5.10 MDMA channel x block repeat address update register (MDMA_CxBRUR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x10).

Address offset: 0x60 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DUV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **DUV[15:0]**: Destination address Update Value

This value is used to update (by addition or subtraction) the current destination address at the end of a block transfer. Must be an integer multiple of DSIZE, in order to keep DAR aligned to DSIZE (e.g. DAR[1:0] = 00 when DSIZE=10).

If this value is 0, the next repetition of the block transfer will continue to the next address.

When the block repeat mode is not active (BRC=0), this field is ignored.

These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

Note: This field must be programmed to 0 when DINC[1:0] = 00.

Bits 15:0 **SUV[15:0]**: Source address Update Value

This value is used to update (by addition or subtraction) the current source address at the end of a block Transfer. Must be an integer multiple of SSIZE, in order to keep SAR aligned to SSIZE (e.g. SAR[1:0] = 00 when SSIZE=10).

If this value is 0, the next repetition of the block transfer will continue from the next address.

When the block repeat mode is not active (BRC=0), this field is ignored.

These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

Note: This field must be programmed to 0 when SINC[1:0] = 00.

13.5.11 MDMA channel x link address register (MDMA_CxLAR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x14).

Note: The new value is only taken into account after all registers are updated, for the next end of block.

Address offset: 0x64 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **LAR[31:0]**: Link Address Register

At the end of a (repeated) block Transfer, the current channel configuration registers (CxTCR, CxBNDTR, CxSAR, CxDAR, CxBRUR, CxMAR, CxMDR and the CxLAR register itself) are loaded with the data structure found at this address.

If the value of this register is 0, no register update will take place, the channel will be disabled and the CTCIF will be set, indicating the end of the transfer for this channel. These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

The channel configuration (LAR address) must be in the AXI address space.

LAR value must be aligned at a Double Word address, i.e. LAR[2:0] = 0x0

13.5.12 MDMA channel x trigger and bus selection register (MDMA_CxTBR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x18).

Address offset: 0x68 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBUS	SBUS
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSEL[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DBUS**: Destination BUS select

0: The system/AXI bus is used as destination (write operation) on channel x.

1: The AHB bus/TCM is used as destination (write operation) on channel x.

This bit is protected and can be written only if EN is 0.

Bit 16 **SBUS**: Source BUS select

0: The system/AXI bus is used as source (read operation) on channel x.

1: The AHB bus/TCM is used as source (read operation) on channel x.

This bit is protected and can be written only if EN is 0.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **TSEL[5:0]**: Trigger Selection

This bit field selects the hardware trigger (RQ) input for channel x. The ACK is sent on the ACK output having the same index value.

When SWRM bit is set (software request selected), this bit field is ignored.

These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

Note: If multiple channels are triggered by the same event (have the same TSEL value), all of them will be triggered in parallel. However, only the channel with the lowest index will acknowledge the request .

13.5.13 MDMA channel x mask address register (MDMA_CxMAR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x20).

Address offset: 0x70 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MAR[31:0]**: Mask address

A write of the MDR value will also be done to this address. This allows to clear the RQ signal generated by the DMA2 by writing to its Interrupt Clear register.
If the value of this register is 0, this function is disabled. These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

13.5.14 MDMA channel x mask data register (MDMA_CxMDR)

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register will be loaded from memory (from address given by current LAR[31:0] + 0x24).

Address offset: 0x74 + 0x40 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MDR[31:0]**: Mask Data

A write of the MDR value will also be done to the address defined by the MAR register. This allows to clear the RQ signal generated by the DMA2 by writing to its Interrupt Clear register.
These bits are write-protected and can be written only when bit EN = '0' in the MDMA_CxCR register.

13.5.15 MDMA register map

Table 100 summarizes the MDMA registers.

Table 100. MDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	MDMA_GISR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GIF15	GIF14	GIF13	GIF12	GIF11	GIF10	GIF9	GIF8	GIF7	GIF6	GIF5	GIF4	GIF3	GIF2	GIF1	GIF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04 - 0x3C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x40 + 0x40 × channel number	MDMA_CxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRQA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCIF	BTIF	BRTIF	CTCIF	TEIF
	Reset value																0												0	0	0	0	0
0x44 + 0x40 × channel number	MDMA_CxIFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLTCIF	CBTIF	CBRTIF	CCTCIF	CTEIF
	Reset value																												0	0	0	0	0
0x48 + 0x40 × channel number	MDMA_CxESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSE	ASE	TEMD	TELD	TED	TEA[6:0]						
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x4C + 0x40 × channel number	MDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWRQ	Res.	WEX	HEX	BEX	Res.	Res.	Res.	Res.	PL[1:0]		TCIE	BTIE	BRTIE	CTCIE	TEIE	EN
	Reset value																0		0	0	0					0	0	0	0	0	0	0	0
0x50 + 0x40 × channel number	MDMA_CxTCR	BWM	SWRM	TRGM[1:0]		PAM[1:0]		PKE	TLEN[6:0].							DBURST[2:0]		SBURST[2:0]			DINCOS[1:0]		SINCOS[1:0]		DSIZE[1:0]		SSIZE[1:0]		DINC[1:0]		SINC[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54 + 0x40 × channel number	MDMA_CxBNDTR	BRC[11:0]												BRDUM	BRSUM	Res.	BNDT[16:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58 + 0x40 × channel number	MDMA_CxSAR	SAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C + 0x40 × channel number	MDMA_CxDAR	DAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60 + 0x40 × channel number	MDMA_CxBRUR	DUV[15:0].																SUV[15:0].															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64 + 0x40 × channel number	MDMA_CxLAR	LAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68 + 0x40 × channel number	MDMA_CxTBR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBUS	SBUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSEL[5:0]					
	Reset value															0	0											0	0	0	0	0	0
0x6C + 0x40 × channel number	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x70 + 0x40 × channel number	MDMA_CxMAR	MAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 100. MDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x74 + 0x40 × channel number	MDMA_CxMDR	MDR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74 - 0x7C + 0x40 × channel number	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

14 Direct memory access controller (DMA)

14.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be quickly moved by DMA without any CPU action. This keeps CPU resources free for other operations.

The DMA controller combines a powerful dual AHB master bus architecture with independent FIFO to optimize the bandwidth of the system, based on a complex bus matrix architecture.

The two DMA controllers (DMA1 and DMA2) have 16 streams in total (8 for each controller), each dedicated to managing memory access requests from one or more peripherals.

Each DMA stream is driven by one DMAMUX1 output channel (request). Any DMAMUX1 output request can be individually programmed in order to select the DMA request source signal, from any of the 115 available request input signals.

Refer to the *Section 16.3: DMAMUX implementation* for more information about the DMA requests and streams mapping.

Each DMA controller has an arbiter for handling the priority between DMA requests.

14.2 DMA main features

The main DMA features are:

- Dual AHB master bus architecture, one dedicated to memory accesses and one dedicated to peripheral accesses
- AHB slave programming interface supporting only 32-bit accesses
- 8 streams for each DMA controller, up to 115 channels (requests) per stream
- Four-word depth 32 first-in, first-out memory buffers (FIFOs) per stream, that can be used in FIFO mode or direct mode:
  - FIFO mode: with threshold level software selectable between 1/4, 1/2 or 3/4 of the FIFO size
  - Direct mode: each DMA request immediately initiates a transfer from/to the memory. When it is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads only one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.
- Each stream can be configured to be:
  - a regular channel that supports peripheral-to-memory, memory-to-peripheral and memory-to-memory transfers
  - a double buffer channel that also supports double buffering on the memory side
- Priorities between DMA stream requests are software-programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (for example, request 0 has priority over request 1)
- Each stream also supports software trigger for memory-to-memory transfers

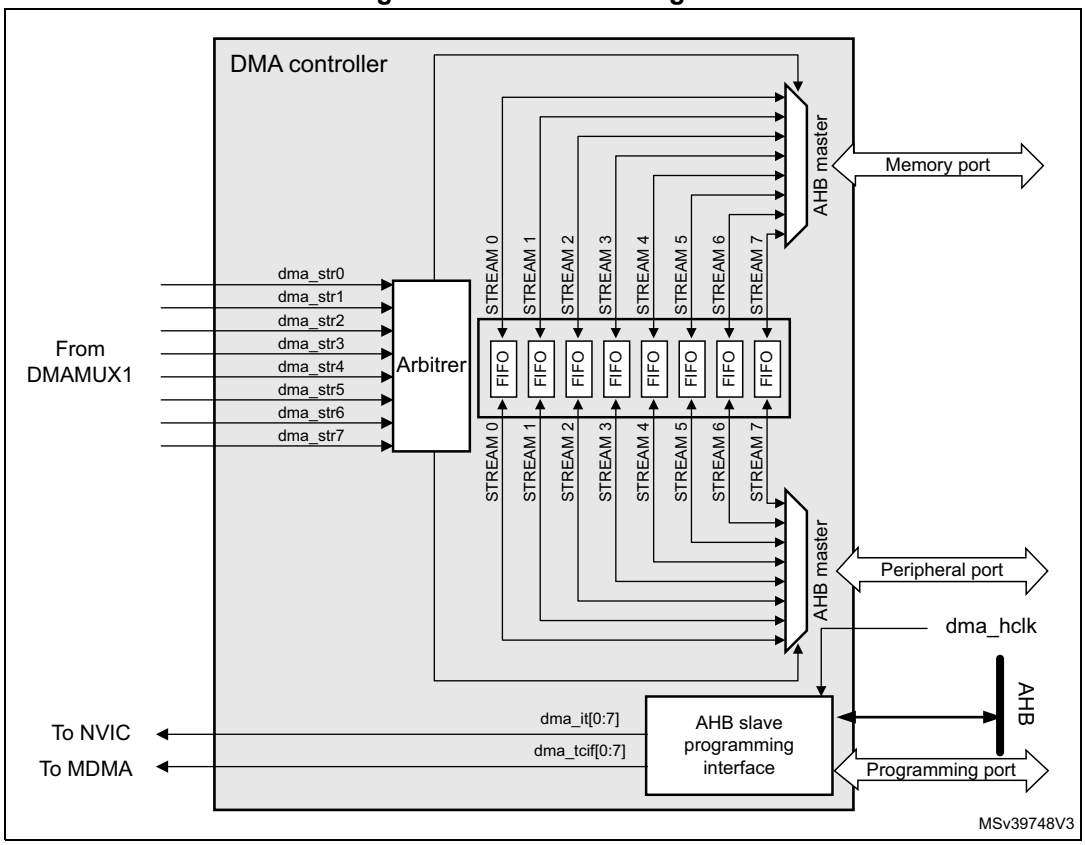
- Each stream request can be selected among up to 115 possible channel requests. This selection is software-configurable by the DMAMUX1 and allows 107 peripherals to initiate DMA requests
- The number of data items to be transferred can be managed either by the DMA controller or by the peripheral:
  - DMA flow controller: the number of data items to be transferred is software-programmable from 1 to 65535
  - Peripheral flow controller: the number of data items to be transferred is unknown and controlled by the source or the destination peripheral that signals the end of the transfer by hardware
- Independent source and destination transfer width (byte, half-word, word): when the data widths of the source and destination are not equal, the DMA automatically packs/unpacks the necessary transfers to optimize the bandwidth. This feature is only available in FIFO mode
- Incrementing or non-incrementing addressing for source and destination
- Supports incremental burst transfers of 4, 8 or 16 beats. The size of the burst is software-configurable, usually equal to half the FIFO size of the peripheral
- Each stream supports circular buffer management
- 5 event flags (DMA half transfer, DMA transfer complete, DMA transfer error, DMA FIFO error, direct mode error) logically ORed together in a single interrupt request for each stream

14.3 DMA functional description

14.3.1 DMA block diagram

Figure 74 shows the block diagram of a DMA.

Figure 74. DMA block diagram



14.3.2 DMA internal signals

Table 101 shows the internal DMA signals.

Table 101. DMA internal input/output signals

Signal name	Signal type	Description
dma_hclk	Digital input	DMA AHB clock
dma_it[0:7]	Digital outputs	DMA stream [0:7] global interrupts
dma_tcif[0:7]	Digital outputs	MDMA triggers
dma_str[0:7]	Digital input	DMA stream [0:7] requests

14.3.3 DMA overview

The DMA controller performs direct memory transfer: as an AHB master, it can take the control of the AHB bus matrix to initiate AHB transactions.

It carries out the following transactions:

- peripheral-to-memory
- memory-to-peripheral
- memory-to-memory

The DMA controller provides two AHB master ports: the AHB memory port, intended to be connected to memories and the AHB peripheral port, intended to be connected to peripherals. However, to allow memory-to-memory transfers, the AHB peripheral port must also have access to the memories.

The AHB slave port is used to program the DMA controller (it supports only 32-bit accesses).

14.3.4 DMA transactions

A DMA transaction consists of a sequence of a given number of data transfers. The number of data items to be transferred and their width (8-bit, 16-bit or 32-bit) are software-programmable.

Each DMA transfer consists of three operations:

- a loading from the peripheral data register or a location in memory, addressed through the DMA_SxPAR or DMA_SxM0AR register
- a storage of the data loaded to the peripheral data register or a location in memory addressed through the DMA_SxPAR or DMA_SxM0AR register
- a post-decrement of the DMA_SxNDTR register, containing the number of transactions that still have to be performed

After an event, the peripheral sends a request signal to the DMA controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA controller accesses the peripheral, an Acknowledge signal is sent to the peripheral by the DMA controller. The peripheral releases its request as soon as it gets the Acknowledge signal from the DMA controller. Once the request has been deasserted by the peripheral, the DMA controller releases the Acknowledge signal. If there are more requests, the peripheral can initiate the next transaction.

14.3.5 DMA request mapping

The DMA request mapping from peripherals to DMA streams is described in *Section 16.3.2: DMAMUX1 mapping*.

14.3.6 Arbiter

An arbiter manages the 8 DMA stream requests based on their priority for each of the two AHB master ports (memory and peripheral ports) and launches the peripheral/memory access sequences.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the DMA_SxCR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: If two requests have the same software priority level, the stream with the lower number takes priority over the stream with the higher number. For example, stream 2 takes priority over stream 4.

14.3.7 DMA streams

Each of the 8 DMA controller streams provides a unidirectional transfer link between a source and a destination.

Each stream can be configured to perform:

- Regular type transactions: memory-to-peripherals, peripherals-to-memory or memory-to-memory transfers
- Double-buffer type transactions: double buffer transfers using two memory pointers for the memory (while the DMA is reading/writing from/to a buffer, the application can write/read to/from the other buffer).

The amount of data to be transferred (up to 65535) is programmable and related to the source width of the peripheral that requests the DMA transfer connected to the peripheral AHB port. The register that contains the amount of data items to be transferred is decremented after each transaction.

14.3.8 Source, destination and transfer modes

Both source and destination transfers can address peripherals and memories in the entire 4 Gbytes area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The direction is configured using the DIR[1:0] bits in the DMA_SxCR register and offers three possibilities: memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers. *Table 102* describes the corresponding source and destination addresses.

Table 102. Source and destination address

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR
01	Memory-to-peripheral	DMA_SxM0AR	DMA_SxPAR
10	Memory-to-memory	DMA_SxPAR	DMA_SxM0AR
11	Reserved	-	-

When the data width (programmed in the PSIZE or MSIZE bits in the DMA_SxCR register) is a half-word or a word, respectively, the peripheral or memory address written into the DMA_SxPAR or DMA_SxM0AR/M1AR registers has to be aligned on a word or half-word address boundary, respectively.

Peripheral-to-memory mode

Figure 75 describes this mode.

When this mode is enabled (by setting the bit EN in the DMA_SxCR register), each time a peripheral request occurs, the stream initiates a transfer from the source to fill the FIFO.

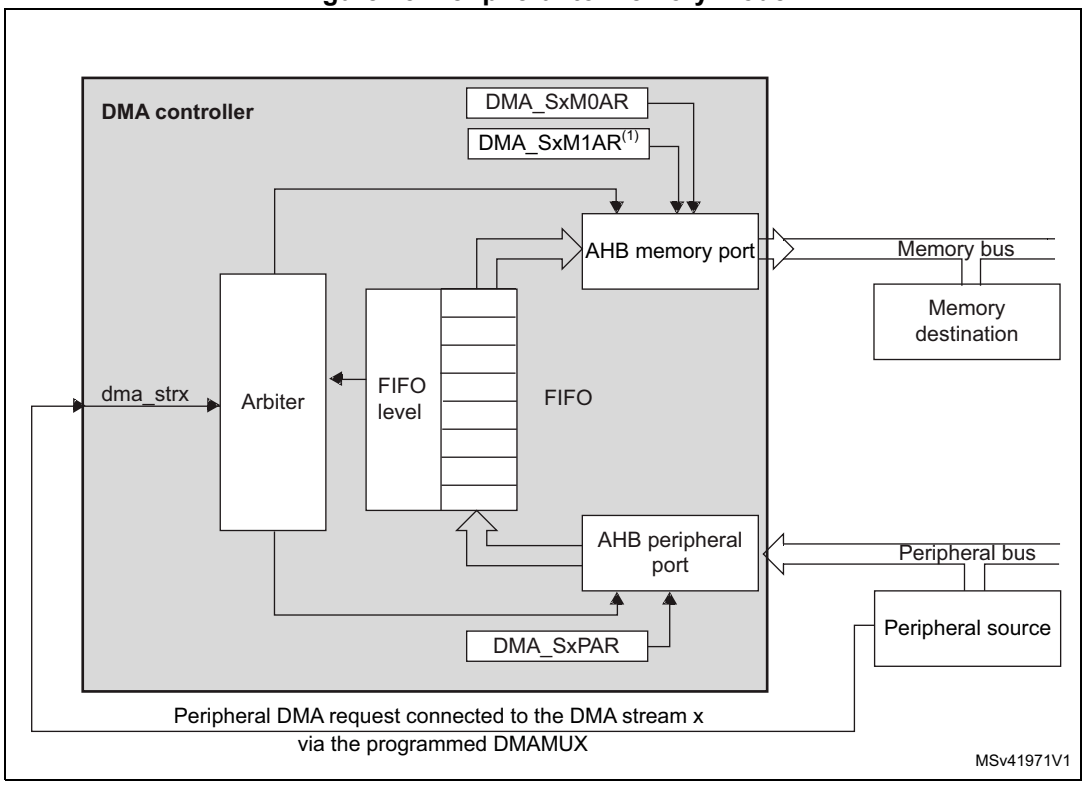
When the threshold level of the FIFO is reached, the contents of the FIFO are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is '0'), the threshold level of the FIFO is not used: after each single data transfer from the peripheral to the FIFO, the corresponding data are immediately drained and stored into the destination.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 75. Peripheral-to-memory mode



1. For double-buffer mode.

Memory-to-peripheral mode

Figure 76 describes this mode.

When this mode is enabled (by setting the EN bit in the DMA_SxCR register), the stream immediately initiates transfers from the source to entirely fill the FIFO.

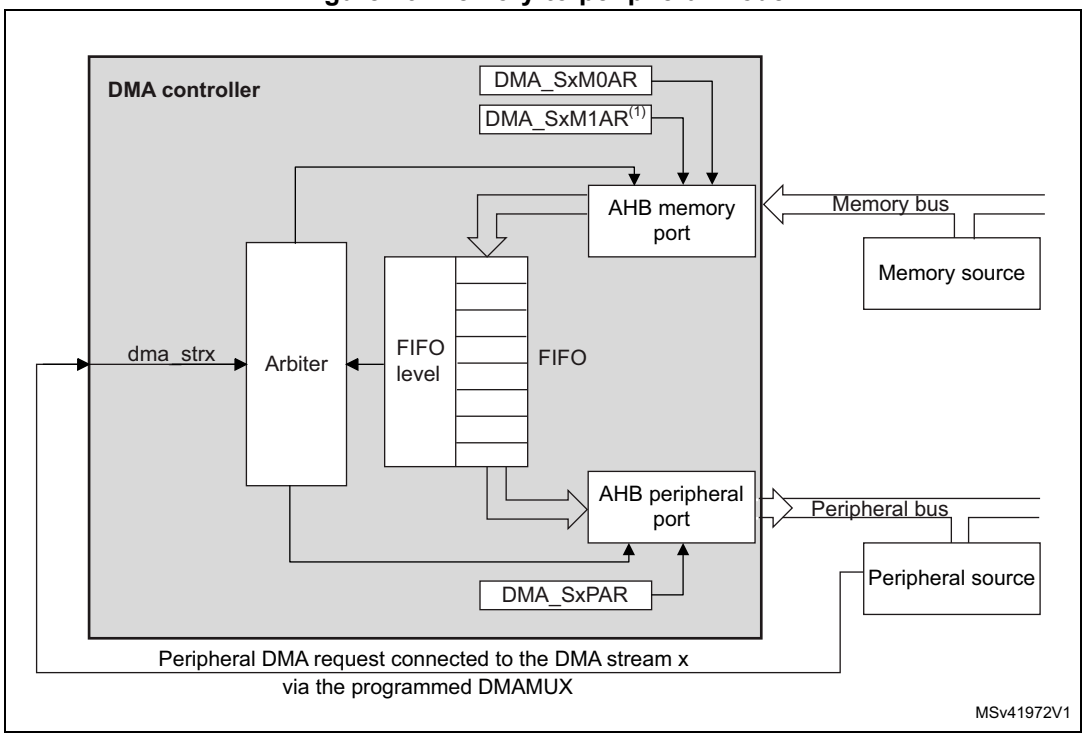
Each time a peripheral request occurs, the contents of the FIFO are drained and stored into the destination. When the level of the FIFO is lower than or equal to the predefined threshold level, the FIFO is fully reloaded with data from the memory.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is '0'), the threshold level of the FIFO is not used. Once the stream is enabled, the DMA preloads the first data to transfer into an internal FIFO. As soon as the peripheral requests a data transfer, the DMA transfers the preloaded value into the configured destination. It then reloads again the empty internal FIFO with the next data to be transfer. The preloaded data size corresponds to the value of the PSIZE bitfield in the DMA_SxCR register.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 76. Memory-to-peripheral mode



1. For double-buffer mode.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This is the memory-to-memory mode, described in *Figure 77*.

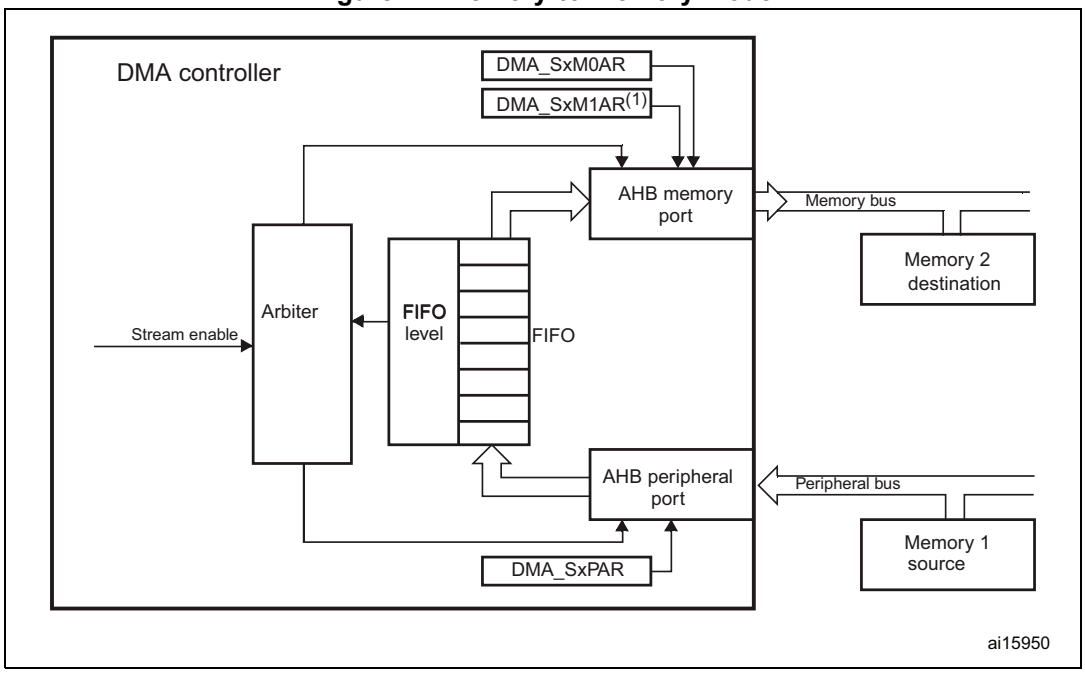
When the stream is enabled by setting the Enable bit (EN) in the DMA_SxCR register, the stream immediately starts to fill the FIFO up to the threshold level. When the threshold level is reached, the FIFO contents are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero or when the EN bit in the DMA_SxCR register is cleared by software.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Note: When memory-to-memory mode is used, the circular and direct modes are not allowed.

Figure 77. Memory-to-memory mode



1. For double-buffer mode.

14.3.9 Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented or kept constant after each transfer depending on the PINC and MINC bits in the DMA_SxCR register.

Disabling the increment mode is useful when the peripheral source or destination data is accessed through a single register.

If the increment mode is enabled, the address of the next transfer is the address of the previous one incremented by 1 (for bytes), 2 (for half-words) or 4 (for words) depending on the data width programmed in the PSIZE or MSIZE bits in the DMA_SxCR register.

In order to optimize the packing operation, it is possible to fix the increment offset size for the peripheral address whatever the size of the data transferred on the AHB peripheral port. The PINCOS bit in the DMA_SxCR register is used to align the increment offset size with the data size on the peripheral AHB port, or on a 32-bit address (the address is then incremented by 4). The PINCOS bit has an impact on the AHB peripheral port only.

If the PINCOS bit is set, the address of the following transfer is the address of the previous one incremented by 4 (automatically aligned on a 32-bit address), whatever the PSIZE value. The AHB memory port, however, is not impacted by this operation.

14.3.10 Circular mode

The circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_SxCR register.

When the circular mode is activated, the number of data items to be transferred is automatically reloaded with the initial value programmed during the stream configuration phase, and the DMA requests continue to be served.

Note: In the circular mode, it is mandatory to respect the following rule in case of a burst mode configured for memory:

DMA_SxNDTR = Multiple of ((Mburst beat) × (Msize)/(Psize)), where:

- *(Mburst beat) = 4, 8 or 16 (depending on the MBURST bits in the DMA_SxCR register)*
- *((Msize)/(Psize)) = 1, 2, 4, 1/2 or 1/4 (Msize and Psize represent the MSIZE and PSIZE bits in the DMA_SxCR register. They are byte dependent)*
- *DMA_SxNDTR = Number of data items to transfer on the AHB peripheral port*

For example: Mburst beat = 8 (INCR8), MSIZE = '00' (byte) and PSIZE = '01' (half-word), in this case: DMA_SxNDTR must be a multiple of (8 × 1/2 = 4).

If this formula is not respected, the DMA behavior and data integrity are not guaranteed.

NDTR must also be a multiple of the Peripheral burst size multiplied by the peripheral data size, otherwise this could result in a bad DMA behavior.

14.3.11 Double-buffer mode

This mode is available for all the DMA1 and DMA2 streams.

The double-buffer mode is enabled by setting the DBM bit in the DMA_SxCR register.

A double-buffer stream works as a regular (single buffer) stream with the difference that it has two memory pointers. When the double-buffer mode is enabled, the circular mode is automatically enabled (CIRC bit in DMA_SxCR is not relevant) and at each end of transaction, the memory pointers are swapped.

In this mode, the DMA controller swaps from one memory target to another at each end of transaction. This allows the software to process one memory area while the second memory area is being filled/used by the DMA transfer. The double-buffer stream can work in both directions (the memory can be either the source or the destination) as described in *Table 103: Source and destination address registers in double-buffer mode (DBM = 1)*.

*Note: In double-buffer mode, it is possible to update the base address for the AHB memory port on-the-fly (*DMA_SxM0AR or DMA_SxM1AR*) when the stream is enabled, by respecting the following conditions:*

- *When the CT bit is '0' in the DMA_SxCR register, the DMA_SxM1AR register can be written. Attempting to write to this register while CT = '1' sets an error flag (TEIF) and the stream is automatically disabled.*
- *When the CT bit is '1' in the DMA_SxCR register, the DMA_SxM0AR register can be written. Attempting to write to this register while CT = '0', sets an error flag (TEIF) and the stream is automatically disabled.*

To avoid any error condition, it is advised to change the base address as soon as the TCIF flag is asserted because, at this point, the targeted memory must have changed from memory 0 to 1 (or from 1 to 0) depending on the value of CT in the DMA_SxCR register in accordance with one of the two above conditions.

For all the other modes (except the double-buffer mode), the memory address registers are write-protected as soon as the stream is enabled.

Table 103. Source and destination address registers in double-buffer mode (DBM = 1)

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR / DMA_SxM1AR
01	Memory-to-peripheral	DMA_SxM0AR / DMA_SxM1AR	DMA_SxPAR
10	Not allowed(1)		
11	Reserved	-	-

1. When the double-buffer mode is enabled, the circular mode is automatically enabled. Since the memory-to-memory mode is not compatible with the circular mode, when the double-buffer mode is enabled, it is not allowed to configure the memory-to-memory mode.

14.3.12 Programmable data width, packing/unpacking, endianness

The number of data items to be transferred has to be programmed into DMA_SxNDTR (number of data items to transfer bit, NDT) before enabling the stream (except when the flow controller is the peripheral, PFCTRL bit in DMA_SxCR is set).

When using the internal FIFO, the data widths of the source and destination data are programmable through the PSIZE and MSIZE bits in the DMA_SxCR register (can be 8-, 16- or 32-bit).

When PSIZE and MSIZE are not equal:

- The data width of the number of data items to transfer, configured in the DMA_SxNDTR register is equal to the width of the peripheral bus (configured by the PSIZE bits in the DMA_SxCR register). For instance, in case of peripheral-to-memory, memory-to-peripheral or memory-to-memory transfers and if the PSIZE[1:0] bits are configured for half-word, the number of bytes to be transferred is equal to 2 × NDT.
- The DMA controller only copes with little-endian addressing for both source and destination. This is described in *Table 104: Packing/unpacking and endian behavior (bit PINC = MINC = 1)*.

This packing/unpacking procedure may present a risk of data corruption when the operation is interrupted before the data are completely packed/unpacked. So, to ensure data coherence, the stream may be configured to generate burst transfers: in this case, each

group of transfers belonging to a burst are indivisible (refer to *Section 14.3.13: Single and burst transfers*).

In direct mode (DMDIS = 0 in the DMA_SxFCR register), the packing/unpacking of data is not possible. In this case, it is not allowed to have different source and destination transfer data widths: both are equal and defined by the PSIZE bits in the DMA_SxCR register. MSIZE bits are not relevant.

Table 104. Packing/unpacking and endian behavior (bit PINC = MINC = 1)

AHB memory port width	AHB peripheral port width	Number of data items to transfer (NDT)	Memory transfer number	Memory port address / byte lane	Peripheral transfer number	Peripheral port address / byte lane	
						PINCOS = 1	PINCOS = 0
8	8	4	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
8	16	2	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
8	32	1	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
16	8	4	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
16	16	2	1 2	0x0 / B1|B0[15:0] 0x2 / B1|B0[15:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
16	32	1	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
32	8	4	1	0x0 / B3|B2|B1|B0[31:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
32	16	2	1	0x0 /B3|B2|B1|B0[31:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
32	32	1	1	0x0 /B3|B2|B1|B0 [31:0]	1	0x0 /B3|B2|B1|B0 [31:0]	0x0 / B3|B2|B1|B0[31:0]

Note: Peripheral port may be the source or the destination (it could also be the memory source in the case of memory-to-memory transfer).

PSIZE, MSIZE and NDT[15:0] have to be configured so as to ensure that the last transfer will not be incomplete. This can occur when the data width of the peripheral port (PSIZE bits) is lower than the data width of the memory port (MSIZE bits). This constraint is summarized in *Table 105*.

Table 105. Restriction on NDT versus PSIZE and MSIZE

PSIZE[1:0] of DMA_SxCR	MSIZE[1:0] of DMA_SxCR	NDT[15:0] of DMA_SxNDTR
00 (8-bit)	01 (16-bit)	must be a multiple of 2
00 (8-bit)	10 (32-bit)	must be a multiple of 4
01 (16-bit)	10 (32-bit)	must be a multiple of 2

14.3.13 Single and burst transfers

The DMA controller can generate single transfers or incremental burst transfers of 4, 8 or 16 beats.

The size of the burst is configured by software independently for the two AHB ports by using the MBURST[1:0] and PBURST[1:0] bits in the DMA_SxCR register.

The burst size indicates the number of beats in the burst, not the number of bytes transferred.

To ensure data coherence, each group of transfers that form a burst are indivisible: AHB transfers are locked and the arbiter of the AHB bus matrix does not degrant the DMA master during the sequence of the burst transfer.

Depending on the single or burst configuration, each DMA request initiates a different number of transfers on the AHB peripheral port:

- When the AHB peripheral port is configured for single transfers, each DMA request generates a data transfer of a byte, half-word or word depending on the PSIZE[1:0] bits in the DMA_SxCR register
- When the AHB peripheral port is configured for burst transfers, each DMA request generates 4,8 or 16 beats of byte, half word or word transfers depending on the PBURST[1:0] and PSIZE[1:0] bits in the DMA_SxCR register.

The same as above has to be considered for the AHB memory port considering the MBURST and MSIZE bits.

In direct mode, the stream can only generate single transfers and the MBURST[1:0] and PBURST[1:0] bits are forced by hardware.

The address pointers (DMA_SxPAR or DMA_SxM0AR registers) must be chosen so as to ensure that all transfers within a burst block are aligned on the address boundary equal to the size of the transfer.

The burst configuration has to be selected in order to respect the AHB protocol, where bursts **must not** cross the 1 Kbyte address boundary because the minimum address space that can be allocated to a single slave is 1 Kbyte. This means that the 1 Kbyte address boundary **must not** be crossed by a burst block transfer, otherwise an AHB error is generated, that is not reported by the DMA registers.

14.3.14 FIFO

FIFO structure

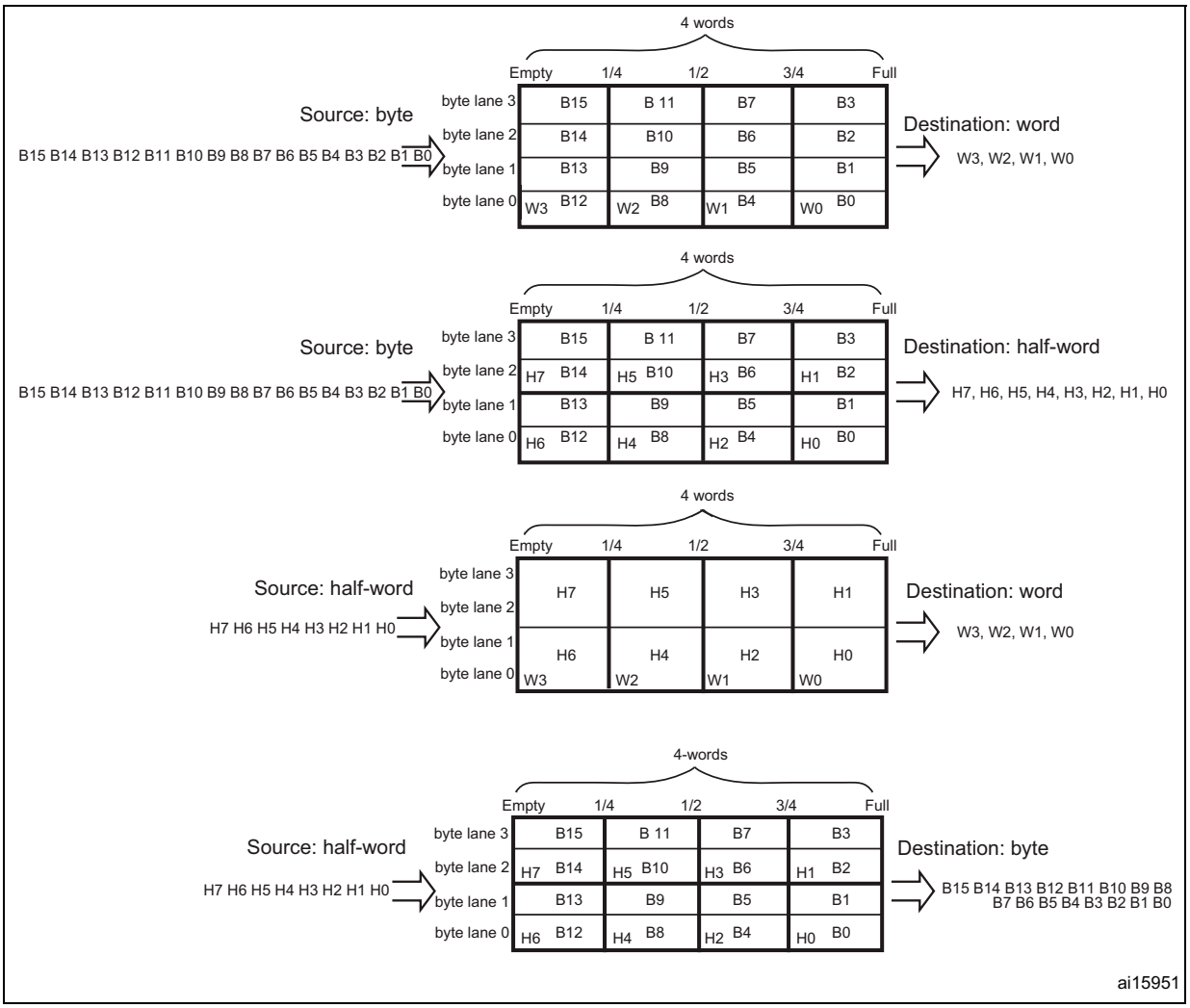
The FIFO is used to temporarily store data coming from the source before transmitting them to the destination.

Each stream has an independent 4-word FIFO and the threshold level is software-configurable between 1/4, 1/2, 3/4 or full.

To enable the use of the FIFO threshold level, the direct mode must be disabled by setting the DMDIS bit in the DMA_SxFCR register.

The structure of the FIFO differs depending on the source and destination data widths, and is described in *Figure 78: FIFO structure*.

Figure 78. FIFO structure



FIFO threshold and burst configuration

Caution is required when choosing the FIFO threshold (bits FTH[1:0] of the DMA_SxFCR register) and the size of the memory burst (MBURST[1:0] of the DMA_SxCR register): The content pointed by the FIFO threshold must exactly match an integer number of memory burst transfers. If this is not in the case, a FIFO error (flag FEIFx of the DMA_HISR or DMA_LISR register) is generated when the stream is enabled, then the stream is automatically disabled. The allowed and forbidden configurations are described in *Table 106*. The forbidden configurations are highlighted in gray in the table.

Table 106. FIFO threshold configurations

MSIZE	FIFO level	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
Byte	1/4	1 burst of 4 beats	Forbidden	Forbidden
	1/2	2 bursts of 4 beats	1 burst of 8 beats	
	3/4	3 bursts of 4 beats	Forbidden	
	Full	4 bursts of 4 beats	2 bursts of 8 beats	1 burst of 16 beats
Half-word	1/4	Forbidden	Forbidden	Forbidden
	1/2	1 burst of 4 beats		
	3/4	Forbidden		
	Full	2 bursts of 4 beats	1 burst of 8 beats	
Word	1/4	Forbidden	Forbidden	
	1/2			
	3/4			
	Full	1 burst of 4 beats		

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be: 1 (byte), 2 (half-word) or 4 (word)).

Incomplete burst transfer at the end of a DMA transfer may happen if one of the following conditions occurs:

- For the AHB peripheral port configuration: the total number of data items (set in the DMA_SxNDTR register) is not a multiple of the burst size multiplied by the data size.
- For the AHB memory port configuration: the number of remaining data items in the FIFO to be transferred to the memory is not a multiple of the burst size multiplied by the data size.

In such cases, the remaining data to be transferred is managed in single mode by the DMA, even if a burst transaction is requested during the DMA stream configuration.

Note: When burst transfers are requested on the peripheral AHB port and the FIFO is used (DMDIS = 1 in the DMA_SxCR register), it is mandatory to respect the following rule to avoid permanent underrun or overrun conditions, depending on the DMA stream direction:

If (PBURST × PSIZE) = FIFO_SIZE (4 words), FIFO_Threshold = 3/4 is forbidden with PSIZE = 1, 2 or 4 and PBURST = 4, 8 or 16.

This rule ensures that enough FIFO space at a time is free to serve the request from the peripheral.

FIFO flush

The FIFO can be flushed when the stream is disabled by resetting the EN bit in the DMA_SxCR register and when the stream is configured to manage peripheral-to-memory or memory-to-memory transfers. If some data are still present in the FIFO when the stream is disabled, the DMA controller continues transferring the remaining data to the destination (even though stream is effectively disabled). When this flush is completed, the transfer complete status bit (TCIFx) in the DMA_LISR or DMA_HISR register is set.

The remaining data counter DMA_SxNDTR keeps the value in this case to indicate how many data items are currently available in the destination memory.

Note that during the FIFO flush operation, if the number of remaining data items in the FIFO to be transferred to memory (in bytes) is less than the memory data width (for example 2 bytes in FIFO while MSIZE is configured to word), data is sent with the data width set in the MSIZE bit in the DMA_SxCR register. This means that memory is written with an undesired value. The software may read the DMA_SxNDTR register to determine the memory area that contains the good data (start address and last address).

If the number of remaining data items in the FIFO is lower than a burst size (if the MBURST bits in DMA_SxCR register are set to configure the stream to manage burst on the AHB memory port), single transactions are generated to complete the FIFO flush.

Direct mode

By default, the FIFO operates in direct mode (DMDIS bit in the DMA_SxFCR is reset) and the FIFO threshold level is not used. This mode is useful when the system requires an immediate and single transfer to or from the memory after each DMA request.

When the DMA is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.

To avoid saturating the FIFO, it is recommended to configure the corresponding stream with a high priority.

This mode is restricted to transfers where:

- the source and destination transfer widths are equal and both defined by the PSIZE[1:0] bits in DMA_SxCR (MSIZE[1:0] bits are not relevant)
- burst transfers are not possible (PBURST[1:0] and MBURST[1:0] bits in DMA_SxCR are don’t care)

Direct mode must not be used when implementing memory-to-memory transfers.

14.3.15 DMA transfer completion

Different events can generate an end of transfer by setting the TCIFx bit in the DMA_LISR or DMA_HISR status register:

- In DMA flow controller mode:
  - The DMA_SxNDTR counter has reached zero in the memory-to-peripheral mode.
  - The stream is disabled before the end of transfer (by clearing the EN bit in the DMA_SxCR register) and (when transfers are peripheral-to-memory or memory-

to-memory) all the remaining data have been flushed from the FIFO into the memory.

- In Peripheral flow controller mode:
  - The last external burst or single request has been generated from the peripheral and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory
  - The stream is disabled by software, and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory

Note: The transfer completion is dependent on the remaining data in FIFO to be transferred into memory only in the case of peripheral-to-memory mode. This condition is not applicable in memory-to-peripheral mode.

If the stream is configured in noncircular mode, after the end of the transfer (that is when the number of data to be transferred reaches zero), the DMA is stopped (EN bit in DMA_SxCR register is cleared by Hardware) and no DMA request is served unless the software reprograms the stream and re-enables it (by setting the EN bit in the DMA_SxCR register).

14.3.16 DMA transfer suspension

At any time, a DMA transfer can be suspended to be restarted later on or to be definitively disabled before the end of the DMA transfer.

There are two cases:

- The stream disables the transfer with no later-on restart from the point where it was stopped. There is no particular action to do, except to clear the EN bit in the DMA_SxCR register to disable the stream. The stream may take time to be disabled (ongoing transfer is completed first). The transfer complete interrupt flag (TCIF in the DMA_LISR or DMA_HISR register) is set in order to indicate the end of transfer. The value of the EN bit in DMA_SxCR is now ‘0’ to confirm the stream interruption. The DMA_SxNDTR register contains the number of remaining data items at the moment when the stream was stopped so that the software can determine how many data items have been transferred before the stream was interrupted.
- The stream suspends the transfer before the number of remaining data items to be transferred in the DMA_SxNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the stream. In order to restart from the point where the transfer was stopped, the software has to read the DMA_SxNDTR register after disabling the stream by writing the EN bit in DMA_SxCR register (and then checking that it is at ‘0’) to know the number of data items already collected. Then:
  - The peripheral and/or memory addresses have to be updated in order to adjust the address pointers
  - The SxNDTR register has to be updated with the remaining number of data items to be transferred (the value read when the stream was disabled)
  - The stream may then be re-enabled to restart the transfer from the point it was stopped

Note: A transfer complete interrupt flag (TCIF in DMA_LISR or DMA_HISR) is set to indicate the end of transfer due to the stream interruption.

14.3.17 Flow controller

The entity that controls the number of data to be transferred is known as the flow controller. This flow controller is configured independently for each stream using the PFCTRL bit in the DMA_SxCR register.

The flow controller can be:

- The DMA controller: in this case, the number of data items to be transferred is programmed by software into the DMA_SxNDTR register before the DMA stream is enabled.
- The peripheral source or destination: this is the case when the number of data items to be transferred is unknown. The peripheral indicates by hardware to the DMA controller when the last data are being transferred. This feature is only supported for peripherals that are able to signal the end of the transfer.

When the peripheral flow controller is used for a given stream, the value written into the DMA_SxNDTR has no effect on the DMA transfer. Actually, whatever the value written, it will be forced by hardware to 0xFFFF as soon as the stream is enabled, to respect the following schemes:

- Anticipated stream interruption: EN bit in DMA_SxCR register is reset to 0 by the software to stop the stream before the last data hardware signal (single or burst) is sent by the peripheral. In such a case, the stream is switched off and the FIFO flush is triggered in the case of a peripheral-to-memory DMA transfer. The TCIFx flag of the corresponding stream is set in the status register to indicate the DMA completion. To know the number of data items transferred during the DMA transfer, read the DMA_SxNDTR register and apply the following formula:
  – Number_of_data_transferred = 0xFFFF – DMA_SxNDTR
- Normal stream interruption due to the reception of a last data hardware signal: the stream is automatically interrupted when the peripheral requests the last transfer (single or burst) and when this transfer is complete. the TCIFx flag of the corresponding stream is set in the status register to indicate the DMA transfer completion. To know the number of data items transferred, read the DMA_SxNDTR register and apply the same formula as above.
- The DMA_SxNDTR register reaches 0: the TCIFx flag of the corresponding stream is set in the status register to indicate the forced DMA transfer completion. The stream is automatically switched off even though the last data hardware signal (single or burst) has not been yet asserted. The already transferred data is not lost. This means that a maximum of 65535 data items can be managed by the DMA in a single transaction, even in peripheral flow control mode.

Note: When configured in memory-to-memory mode, the DMA is always the flow controller and the PFCTRL bit is forced to 0 by hardware.

The circular mode is forbidden in the peripheral flow controller mode.

14.3.18 Summary of the possible DMA configurations

Table 107 summarizes the different possible DMA configurations. The forbidden configurations are highlighted in gray in the table.

Table 107. Possible DMA configurations

DMA transfer mode	Source	Destination	Flow controller	Circular mode	Transfer type	Direct mode	Double-buffer mode
Peripheral-to-memory	AHB peripheral port	AHB memory port	DMA	Possible	single	Possible	Possible
					burst	Forbidden	
			Peripheral	Forbidden	single	Possible	Forbidden
					burst	Forbidden	
Memory-to-peripheral	AHB memory port	AHB peripheral port	DMA	Possible	single	Possible	Possible
					burst	Forbidden	
			Peripheral	Forbidden	single	Possible	Forbidden
					burst	Forbidden	
Memory-to-memory	AHB peripheral port	AHB memory port	DMA only	Forbidden	single	Forbidden	Forbidden
					burst		

14.3.19 Stream configuration procedure

The following sequence must be followed to configure a DMA stream x (where x is the stream number):

1. If the stream is enabled, disable it by resetting the EN bit in the DMA_SxCR register, then read this bit in order to confirm that there is no ongoing stream operation. Writing this bit to 0 is not immediately effective since it is actually written to 0 once all the current transfers are finished. When the EN bit is read as 0, this means that the stream is ready to be configured. It is therefore necessary to wait for the EN bit to be cleared before starting any stream configuration. All the stream dedicated bits set in the status register (DMA_LISR and DMA_HISR) from the previous data block DMA transfer must be cleared before the stream can be re-enabled.
2. Set the peripheral port register address in the DMA_SxPAR register. The data is moved from/ to this address to/ from the peripheral port after the peripheral event.
3. Set the memory address in the DMA_SxMA0R register (and in the DMA_SxMA1R register in the case of a double-buffer mode). The data is written to or read from this memory after the peripheral event.
4. Configure the total number of data items to be transferred in the DMA_SxNDTR register. After each peripheral event or each beat of the burst, this value is decremented.
5. Use DMAMUX1 to route a DMA request line to the DMA channel.
6. If the peripheral is intended to be the flow controller and if it supports this feature, set the PFCTRL bit in the DMA_SxCR register.
7. Configure the stream priority using the PL[1:0] bits in the DMA_SxCR register.
8. Configure the FIFO usage (enable or disable, threshold in transmission and reception)

9. Configure the data transfer direction, peripheral and memory incremented/fixed mode, single or burst transactions, peripheral and memory data widths, circular mode, double-buffer mode and interrupts after half and/or full transfer, and/or errors in the DMA_SxCR register.
10. Activate the stream by setting the EN bit in the DMA_SxCR register.

As soon as the stream is enabled, it can serve any DMA request from the peripheral connected to the stream.

Once half the data have been transferred on the AHB destination port, the half-transfer flag (HTIF) is set and an interrupt is generated if the half-transfer interrupt enable bit (HTIE) is set. At the end of the transfer, the transfer complete flag (TCIF) is set and an interrupt is generated if the transfer complete interrupt enable bit (TCIE) is set.

Warning: **To switch off a peripheral connected to a DMA stream request, it is mandatory to, first, switch off the DMA stream to which the peripheral is connected, then to wait for EN bit = 0. Only then can the peripheral be safely disabled.**

14.3.20 Error management

The DMA controller can detect the following errors:

- **Transfer error**: the transfer error interrupt flag (TEIFx) is set when:
  - a bus error occurs during a DMA read or a write access
  - a write access is requested by software on a memory address register in double-buffer mode whereas the stream is enabled and the current target memory is the one impacted by the write into the memory address register (refer to *Section 14.3.11: Double-buffer mode*)
- **FIFO error**: the FIFO error interrupt flag (FEIFx) is set if:
  - a FIFO underrun condition is detected
  - a FIFO overrun condition is detected (no detection in memory-to-memory mode because requests and transfers are internally managed by the DMA)
  - the stream is enabled while the FIFO threshold level is not compatible with the size of the memory burst (refer to *Table 106: FIFO threshold configurations*)
- **Direct mode error**: the direct mode error interrupt flag (DMEIFx) can only be set in the peripheral-to-memory mode while operating in direct mode and when the MINC bit in the DMA_SxCR register is cleared. This flag is set when a DMA request occurs while the previous data have not yet been fully transferred into the memory (because the memory bus was not granted). In this case, the flag indicates that 2 data items were be transferred successively to the same destination address, which could be an issue if the destination is not able to manage this situation

In direct mode, the FIFO error flag can also be set under the following conditions:

- In the peripheral-to-memory mode, the FIFO can be saturated (overrun) if the memory bus is not granted for several peripheral requests.
- In the memory-to-peripheral mode, an underrun condition may occur if the memory bus has not been granted before a peripheral request occurs.

If the TEIFx or the FEIFx flag is set due to incompatibility between burst size and FIFO threshold level, the faulty stream is automatically disabled through a hardware clear of its EN bit in the corresponding stream configuration register (DMA_SxCR).

If the DMEIFx or the FEIFx flag is set due to an overrun or underrun condition, the faulty stream is not automatically disabled and it is up to the software to disable or not the stream by resetting the EN bit in the DMA_SxCR register. This is because there is no data loss when this kind of errors occur.

When the stream's error interrupt flag (TEIF, FEIF, DMEIF) in the DMA_LISR or DMA_HISR register is set, an interrupt is generated if the corresponding interrupt enable bit (TEIE, FEIE, DMIE) in the DMA_SxCR or DMA_SxFCR register is set.

Note: When a FIFO overrun or underrun condition occurs, the data is not lost because the peripheral request is not acknowledged by the stream until the overrun or underrun condition is cleared. If this acknowledge takes too much time, the peripheral itself may detect an overrun or underrun condition of its internal buffer and data might be lost.

14.4 DMA interrupts

For each DMA stream, an interrupt can be produced on the following events:

- Half-transfer reached
- Transfer complete
- Transfer error
- FIFO error (overrun, underrun or FIFO level error)
- Direct mode error

Separate interrupt enable control bits are available for flexibility as shown in *Table 108*.

Table 108. DMA interrupt requests

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE
FIFO overrun/underrun	FEIF	FEIE
Direct mode error	DMEIF	DMEIE

Note: Before setting an enable control bit EN = 1, the corresponding event flag must be cleared, otherwise an interrupt is immediately generated.

14.5 DMA registers

The DMA registers have to be accessed by words (32 bits).

14.5.1 DMA low interrupt status register (DMA_LISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCIF3	HTIF3	TEIF3	DMEIF3	Res.	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Res.	FEIF2
				r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TCIF1	HTIF1	TEIF1	DMEIF1	Res.	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Res.	FEIF0
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIF[3:0]**: stream x transfer complete interrupt flag (x = 3..0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: no transfer complete event on stream x
1: a transfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIF[3:0]**: stream x half transfer interrupt flag (x = 3..0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: no half transfer event on stream x
1: a half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIF[3:0]**: stream x transfer error interrupt flag (x = 3..0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: no transfer error on stream x
1: a transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIF[3:0]**: stream x direct mode error interrupt flag (x = 3..0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: No direct mode error on stream x
1: a direct mode error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIF[3:0]**: stream x FIFO error interrupt flag (x = 3..0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_LIFCR register.
0: no FIFO error event on stream x
1: a FIFO error event occurred on stream x

14.5.2 DMA high interrupt status register (DMA_HISR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCIF7	HTIF7	TEIF7	DMEIF7	Res.	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Res.	FEIF6
				r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TCIF5	HTIF5	TEIF5	DMEIF5	Res.	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Res.	FEIF4
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIF[7:4]**: stream x transfer complete interrupt flag (x = 7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: no transfer complete event on stream x

1: atransfer complete event occurred on stream x

Bits 26, 20, 10, 4 **HTIF[7:4]**: stream x half transfer interrupt flag (x = 7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: no half transfer event on stream x

1: a half transfer event occurred on stream x

Bits 25, 19, 9, 3 **TEIF[7:4]**: stream x transfer error interrupt flag (x = 7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: no transfer error on stream x

1: a transfer error occurred on stream x

Bits 24, 18, 8, 2 **DMEIF[7:4]**: stream x direct mode error interrupt flag (x = 7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: no direct mode error on stream x

1: a direct mode error occurred on stream x

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIF[7:4]**: stream x FIFO error interrupt flag (x = 7..4)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_HIFCR register.

0: no FIFO error event on stream x

1: a FIFO error event occurred on stream x

14.5.3 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Res.	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res.	CFEIF2
				w	w	w	w		w	w	w	w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res.	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res.	CFEIF0
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIF[3:0]**: stream x clear transfer complete interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_LISR register.

Bits 26, 20, 10, 4 **CHTIF[3:0]**: stream x clear half transfer interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_LISR register

Bits 25, 19, 9, 3 **CTEIF[3:0]**: Stream x clear transfer error interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_LISR register.

Bits 24, 18, 8, 2 **CDMEIF[3:0]**: stream x clear direct mode error interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_LISR register.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIF[3:0]**: stream x clear FIFO error interrupt flag (x = 3..0)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_LISR register.

14.5.4 DMA high interrupt flag clear register (DMA_HIFCR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Res.	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Res.	CFEIF6
				w	w	w	w		w	w	w	w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Res.	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Res.	CFEIF4
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIF[7:4]**: stream x clear transfer complete interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_HISR register.

Bits 26, 20, 10, 4 **CHTIF[7:4]**: stream x clear half transfer interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_HISR register.

Bits 25, 19, 9, 3 **CTEIF[7:4]**: stream x clear transfer error interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_HISR register.

Bits 24, 18, 8, 2 **CDMEIF[7:4]**: stream x clear direct mode error interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_HISR register.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIF[7:4]**: stream x clear FIFO error interrupt flag (x = 7..4)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_HISR register.

14.5.5 DMA stream x configuration register (DMA_SxCR)

This register is used to configure the concerned stream.

Address offset: 0x10 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		Res.	CT	DBM	PL[1:0]	
							rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:23 **MBURST[1:0]**: memory burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN is ‘0’.
In direct mode, these bits are forced to 0x0 by hardware as soon as bit EN= '1'.

Bits 22:21 **PBURST[1:0]**: peripheral burst transfer configuration
These bits are set and cleared by software.
00: single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN is ‘0’.
In direct mode, these bits are forced to 0x0 by hardware.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CT**: current target (only in double-buffer mode)
This bit is set and cleared by hardware. It can also be written by software.
0: current target memory is Memory 0 (addressed by the DMA_SxM0AR pointer)
1: current target memory is Memory 1 (addressed by the DMA_SxM1AR pointer)
This bit can be written only if EN is ‘0’ to indicate the target memory area of the first transfer.
Once the stream is enabled, this bit operates as a status flag indicating which memory area is the current target.

Bit 18 **DBM**: double-buffer mode
This bit is set and cleared by software.
0: no buffer switching at the end of transfer
1: memory target switched at the end of the DMA transfer
This bit is protected and can be written only if EN is ‘0’.

Bits 17:16 **PL[1:0]**: priority level
These bits are set and cleared by software.
00: low
01: medium
10: high
11: very high
These bits are protected and can be written only if EN is ‘0’.

Bit 15 **PINCOS**: peripheral increment offset size
This bit is set and cleared by software
0: The offset size for the peripheral address calculation is linked to the PSIZE
1: The offset size for the peripheral address calculation is fixed to 4 (32-bit alignment).
This bit has no meaning if bit PINC = '0'.
This bit is protected and can be written only if EN = '0'.
This bit is forced low by hardware when the stream is enabled (bit EN = '1') if the direct mode is selected or if PBURST are different from “00”.

Bits 14:13 **MSIZE[1:0]**: memory data size
These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN is ‘0’.
In direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as bit EN = '1'.

Bits 12:11 **PSIZE[1:0]**: peripheral data size
These bits are set and cleared by software.
00: byte (8-bit)
01: half-word (16-bit)
10: word (32-bit)
11: reserved
These bits are protected and can be written only if EN is ‘0’.

Bit 10 **MINC**: memory increment mode
This bit is set and cleared by software.
0: memory address pointer is fixed
1: memory address pointer is incremented after each data transfer (increment is done according to MSIZE)
This bit is protected and can be written only if EN is ‘0’.

Bit 9 **PINC**: peripheral increment mode
This bit is set and cleared by software.
0: peripheral address pointer is fixed
1: peripheral address pointer is incremented after each data transfer (increment is done according to PSIZE)
This bit is protected and can be written only if EN is ‘0’.

Bit 8 **CIRC**: circular mode

This bit is set and cleared by software and can be cleared by hardware.
0: circular mode disabled
1: circular mode enabled
When the peripheral is the flow controller (bit PFCTRL = 1) and the stream is enabled (bit EN = 1), then this bit is automatically forced by hardware to 0.
It is automatically forced by hardware to 1 if the DBM bit is set, as soon as the stream is enabled (bit EN ='1').

Bits 7:6 **DIR[1:0]**: data transfer direction

These bits are set and cleared by software.
00: peripheral-to-memory
01: memory-to-peripheral
10: memory-to-memory
11: reserved
These bits are protected and can be written only if EN is ‘0’.

Bit 5 **PFCTRL**: peripheral flow controller

This bit is set and cleared by software.
0: DMA is the flow controller
1: The peripheral is the flow controller
This bit is protected and can be written only if EN is ‘0’.
When the memory-to-memory mode is selected (bits DIR[1:0]=10), then this bit is automatically forced to 0 by hardware.

Bit 4 **TCIE**: transfer complete interrupt enable

This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 3 **HTIE**: half transfer interrupt enable

This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 2 **TEIE**: transfer error interrupt enable

This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 1 **DMEIE**: direct mode error interrupt enable

This bit is set and cleared by software.
0: DME interrupt disabled
1: DME interrupt enabled

Bit 0 **EN**: stream enable / flag stream ready when read low

This bit is set and cleared by software.

0: stream disabled

1: stream enabled

This bit may be cleared by hardware:

- on a DMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AHB master buses
- when the FIFO threshold on memory AHB port is not compatible with the size of the burst

When this bit is read as 0, the software is allowed to program the configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.

Note: Before setting EN bit to '1' to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

14.5.6 DMA stream x number of data register (DMA_SxNDTR)

Address offset: 0x14 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]**: number of data items to transfer (0 up to 65535)

This register can be written only when the stream is disabled. When the stream is enabled, this register is read-only, indicating the remaining data items to be transmitted. This register decrements after each DMA transfer.

Once the transfer is completed, this register can either stay at zero (when the stream is in normal mode) or be reloaded automatically with the previously programmed value in the following cases:

- when the stream is configured in circular mode.
- when the stream is enabled again by setting EN bit to '1'.

If the value of this register is zero, no transaction can be served even if the stream is enabled.

14.5.7 DMA stream x peripheral address register (DMA_SxPAR)

Address offset: 0x18 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PAR[31:0]**: peripheral address

Base address of the peripheral data register from/to which the data is read/written.
These bits are write-protected and can be written only when bit EN = '0' in the DMA_SxCR register.

14.5.8 DMA stream x memory 0 address register (DMA_SxM0AR)

Address offset: 0x1C + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M0A[31:0]**: memory 0 address

Base address of memory area 0 from/to which the data is read/written.
These bits are write-protected. They can be written only if:

- the stream is disabled (bit EN= '0' in the DMA_SxCR register) or
- the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '1' in the DMA_SxCR register (in double-buffer mode).

14.5.9 DMA stream x memory 1 address register (DMA_SxM1AR)

Address offset: 0x20 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M1A[31:0]**: memory 1 address (used in case of double-buffer mode)

Base address of memory area 1 from/to which the data is read/written.

This register is used only for the double-buffer mode.

These bits are write-protected. They can be written only if:

– the stream is disabled (bit EN= '0' in the DMA_SxCR register) or

– the stream is enabled (EN='1' in DMA_SxCR register) and bit CT = '0' in the DMA_SxCR register.

14.5.10 DMA stream x FIFO control register (DMA_SxFCR)

Address offset: 0x24 + 0x24 * x, (x = 0 to 7)

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEIE	Res.	FS[2:0]			DMDIS	FTH[1:0]	
								rw		r	r	r	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **FEIE**: FIFO error interrupt enable

This bit is set and cleared by software.

0: FE interrupt disabled

1: FE interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:3 **FS[2:0]**: FIFO status

These bits are read-only.

000: 0 < fifo_level < 1/4

001: 1/4 ≤ fifo_level < 1/2

010: 1/2 ≤ fifo_level < 3/4

011: 3/4 ≤ fifo_level < full

100: FIFO is empty

101: FIFO is full

others: no meaning

These bits are not relevant in the direct mode (DMDIS bit is zero).

Bit 2 **DMDIS**: direct mode disable

This bit is set and cleared by software. It can be set by hardware.

0: direct mode enabled

1: direct mode disabled

This bit is protected and can be written only if EN is '0'.

This bit is set by hardware if the memory-to-memory mode is selected (DIR bit in DMA_SxCR are "10") and the EN bit in the DMA_SxCR register is '1' because the direct mode is not allowed in the memory-to-memory configuration.

Bits 1:0 **FTH[1:0]**: FIFO threshold selection
These bits are set and cleared by software.
00: 1/4 full FIFO
01: 1/2 full FIFO
10: 3/4 full FIFO
11: full FIFO
These bits are not used in the direct mode when the DMIS value is zero.
These bits are protected and can be written only if EN is ‘0’.

14.5.11 DMA register map

Table 109 summarizes the DMA registers.

Table 109. DMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	DMA_LISR	Res.	Res.	Res.	Res.	TCIF3	HTIF3	TEIF3	DMEIF3	Res.	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Res.	FEIF2	Res.	Res.	Res.	Res.	TCIF1	HTIF1	TEIF1	DMEIF1	Res.	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Res.	FEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0004	DMA_HISR	Res	Res	Res	Res	TCIF7	HTIF7	TEIF7	DMEIF7	Res	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Res	FEIF6	Res	Res	Res	Res	TCIF5	HTIF5	TEIF5	DMEIF5	Res	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Res	FEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0008	DMA_LIFCR	Res	Res	Res	Res	CTCIF3	CHTIF3	TEIF3	CDMEIF3	Res	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res	CFEIF2	Res	Res	Res	Res	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res	CFEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x000C	DMA_HIFCR	Res	Res	Res	Res	CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Res	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Res	CFEIF6	Res	Res	Res	Res	CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Res	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Res	CFEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0010	DMA_S0CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		Res	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]	MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	DMA_S0NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	DMA_S0PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	DMA_S0M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	DMA_S0M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	DMA_S0FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0028	DMA_S1CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		Res	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]	MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	DMA_S1NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 109. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0030	DMA_S1PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	DMA_S1M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	DMA_S1M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	DMA_S1FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0040	DMA_S2CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		Res	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	DMA_S2NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	DMA_S2PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	DMA_S2M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	DMA_S2M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	DMA_S2FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0058	DMA_S3CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		Res	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	DMA_S3NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	DMA_S3PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	DMA_S3M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 109. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0068	DMA_S3M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x006C	DMA_S3FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0070	DMA_S4CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		Res.	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0074	DMA_S4NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0078	DMA_S4PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x007C	DMA_S4M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0080	DMA_S4M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	DMA_S4FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0088	DMA_S5CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		Res	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008C	DMA_S5NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0090	DMA_S5PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094	DMA_S5M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0098	DMA_S5M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x009C	DMA_S5FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1

Table 109. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00A0	DMA_S6CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		Res	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A4	DMA_S6NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A8	DMA_S6PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00AC	DMA_S6M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B0	DMA_S6M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B4	DMA_S6FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x00B8	DMA_S7CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		Res	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00BC	DMA_S7NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C0	DMA_S7PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C4	DMA_S7M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8	DMA_S7M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00CC	DMA_S7FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

15 Basic direct memory access controller (BDMA)

15.1 Introduction

The basic direct memory access (BDMA) controller is a bus master and system peripheral.

The BDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The BDMA controller features a single AHB master architecture.

There is one instance of BDMA with 8 channels.

Each channel is dedicated to managing memory access requests from one or more peripherals. The BDMA includes an arbiter for handling the priority between DMA requests.

15.2 BDMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access to D3 domain SRAM and AHB/APB peripherals (BDMA)
- All BDMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to 2^{16} - 1
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.

15.3 BDMA implementation

15.3.1 BDMA

BDMA is implemented with the hardware configuration parameters shown in *Table 110*.

Table 110. BDMA implementation

Feature	BDMA
Number of channels (double-buffer)	8

15.3.2 BDMA request mapping

The BDMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

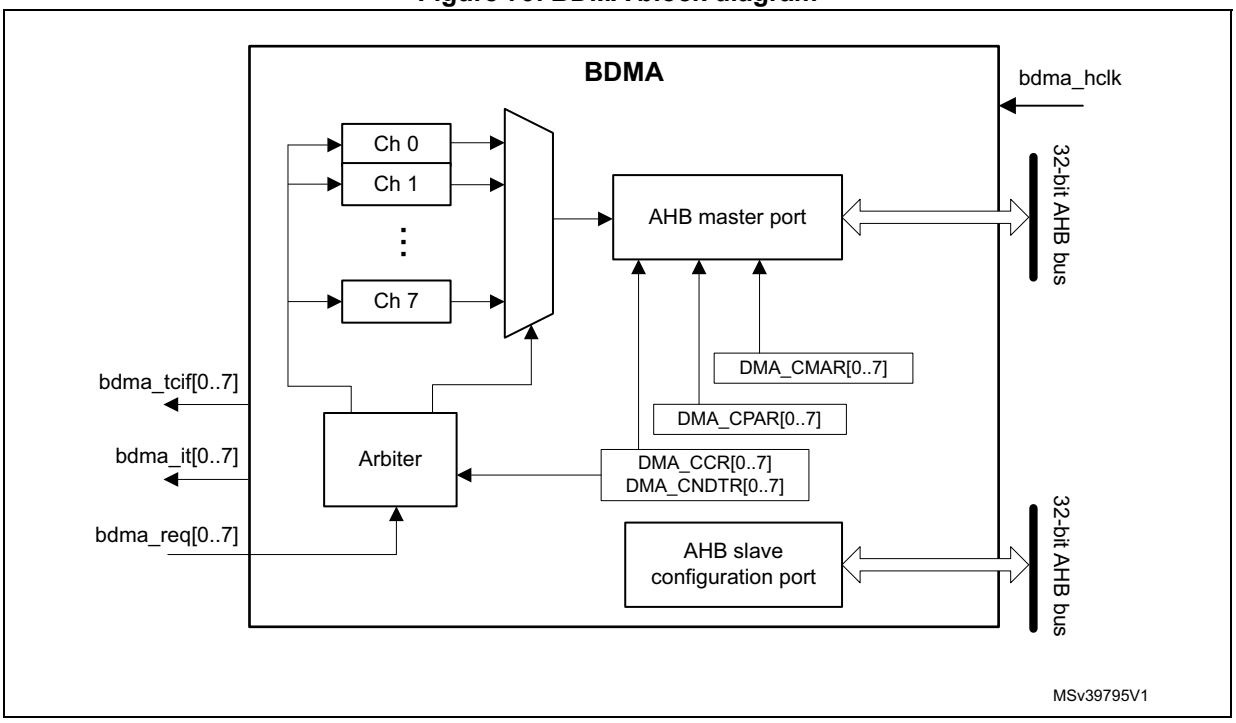
For the mapping of the different requests, refer to the *Section 16.3: DMAMUX implementation*.

15.4 BDMA functional description

15.4.1 BDMA block diagram

The BDMA block diagram is shown in *Figure 79*.

Figure 79. BDMA block diagram



The BDMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests may stop the CPU access to the system bus for a number of bus cycles, when CPU and BDMA target the same destination (memory or peripheral).

The BDMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

According to its configuration through the AHB slave interface, the BDMA controller arbitrates between the DMA channels and their associated received requests. The BDMA controller also schedules the DMA data transfers over the single AHB port master.

The BDMA controller generates an interrupt per channel to the interrupt controller.

15.4.2 BDMA pins and internal signals

Table 111. BDMA internal input/output signals

Signal name	Signal type	Description
bdma_req[x]	Input	BDMA channel x request
bdma_ack[x]	Output	BDMA channel x acknowledge
bdma_it[x]	Output	BDMA channel x interrupt

15.4.3 BDMA transfers

The software configures the BDMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB bus transfers.

A BDMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single BDMA transfer occur:

1. The peripheral sends a single DMA request signal to the BDMA controller.
2. The BDMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the BDMA controller grants the peripheral, an acknowledge is sent to the peripheral by the BDMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the BDMA controller.
5. Once the request is de-asserted by the peripheral, the BDMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single BDMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the BDMA controller. The BDMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a BDMA block transfer consists of a repeated sequence of:

- a single BDMA transfer, encapsulating two AHB transfers of a single data, over the BDMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
    The start address used for the first single transfer is the base address of the peripheral or memory, and is programmed in the BDMA_CPARx or BDMA_CM0/1ARx register.
  - a single data write (byte, half-word or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.

The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the BDMA_CPARx or BDMA_CM0/1ARx register.

- post-decrementing of the programmed BDMA_CNDTRx register
  This register contains the remaining number of data items to transfer (number of AHB 'read followed by write' transfers).

This sequence is repeated until BDMA_CNDTRx is null.

Note: The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.

15.4.4 BDMA arbitration

The BDMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single BDMA transfer is issued (such as a AHB 'read followed by write' transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the BDMA_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low
- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single BDMA transfer of this channel x. Whenever there is another concurrent active requested channel, the BDMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

15.4.5 BDMA channels

Each channel may handle a BDMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable. The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.

Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the BDMA_CCRx register.

Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the BDMA_CCRx register.

If the **incremented mode** is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the BDMA_CPARx or BDMA_CM0/1ARx register. During transfers, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel x is configured in **non-circular mode**, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the BDMA_CNDTRx register.

Note: If the channel x is disabled, the BDMA registers are not reset. The DMA channel registers (BDMA_CCRx, BDMA_CPARx and BDMA_CM0ARx) retain the initial values programmed during the channel configuration phase.

In **circular mode**, after the last data transfer, the BDMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the BDMA_CPARx and BDMA_CM0/1ARx registers.

Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set the peripheral register address in the BDMA_CPARx register.
   The data is moved from/to this address to/from the memory after the peripheral event, or after the channel is enabled in memory-to-memory mode.
2. Set the memory address in the BDMA_CM0ARx register.
   The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the BDMA_CNDTRx register.
   After each data transfer, this value is decremented.
4. Configure the parameters listed below in the BDMA_CCRx register:
   - the channel priority
   - the data transfer direction
   - the circular mode
   - the double-buffer mode
   - the peripheral and memory incremented mode
   - the peripheral and memory data size
   - the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the BDMA_CCRx register.

A channel, as soon as enabled, may serve any BDMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.

Note: The two last steps of the channel configuration procedure may be merged into a single access to the BDMA_CCRx register, to configure and enable the channel.

When a channel is enabled and still active (not completed), the software must perform two separate write accesses to the BDMA_CCRx register, to disable the channel, then to reprogram the channel for another next block transfer.
Some fields of the BDMA_CCRx register are read-only when the EN bit is set to 1.

Stop and resume a channel

Once the software activates a channel, it waits for the completion of the programmed transfer. The BDMA controller is not able to resume an aborted active channel with a possible suspended bus transfer.

To correctly stop and disable a channel, the software clears the EN bit of the BDMA_CCRx register. The software secures that no pending request from the peripheral is served by the BDMA controller before the transfer completion. The software waits for the transfer complete or transfer error interrupt.

When a channel transfer error occurs, the EN bit of the BDMA_CCRx register is cleared by hardware. This EN bit can not be set again by software to re-activate the channel x, until the TEIFx bit of the BDMA_CxISR register is set.

Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the BDMA_CCRx register.

Note: The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the BDMA_CCRx register. When the circular mode is activated, the amount of data to transfer is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel. The software must explicitly program the BDMA_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.

Memory-to-memory mode

The BDMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the BDMA_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the BDMA_CNDTRx register reaches zero.

Note: The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the BDMA_CCRx register.

Double-buffer mode (in memory-to-peripheral and peripheral-to memory transfers)

The BDMA channels can operate in double-buffer mode.

The difference compared to a regular operation is that the BDMA controller toggles between two memory address pointers at the end of each BDMA transfer, thus accessing two memory areas in an alternate way. This allows the software to access one of the two memory areas while the BDMA controller accesses the other one. The double-buffer mode transfer operates in both directions, so the target memory can be either the source or the destination.

The double-buffer mode is configured by setting both the DBM and CIRC bits of the BDMA_CCRx register.

Note: The double-buffer mode must not be used in memory-to-memory mode. Before enabling a channel in double-buffer mode (DBM = 1), the software has to configure appropriately the MEM2MEM bit (MEM2MEM = 0).

The steps described below allow the configuration of a BDMA channel x in double-buffer mode:

- Set the DBM and CIRC bits and clear the MEM2MEM bit of the BDMA_CCRx register. The circular mode is then activated for the swap mechanism to occur.
- Configure the second memory address register BDMA_CM1ARx.
- Continue with the regular channel configuration procedure, and lastly, activate the channel by setting the EN bit of the BDMA_CCRx register. The first BDMA transfer target memory of the corresponding BDMA channel x, is given by the CT bit of the BDMA_CCRx register.

Note: Independently from the value of DBM bit of the BDMA_CCRx register, if CT = 1, the memory address pointer for the BDMA transfer is defined by BDMA_CM1ARx, and not by BDMA_CM0ARx.

Peripheral-to-peripheral mode

Any BDMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the BDMA channel

  This peripheral is the BDMA initiator and paces the data transfer from/to this peripheral to/from a register belonging to another memory-mapped peripheral (this one being not configured in DMA mode).
- when no peripheral request is selected and connected to the BDMA channel

  The software configures a register-to-register transfer by setting the MEM2MEM bit of the BDMA_CCRx register.

Programming transfer direction, assigning source/destination

The value of the DIR bit of the BDMA_CCRx register sets the direction of the transfer, and consequently, it identifies the source and the destination, regardless the source/destination type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  - The **source** attributes are defined by the BDMA_MARx register, the MSIZE[1:0] field and MINC bit of the BDMA_CCRx register.
    Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the source peripheral in peripheral-to-peripheral mode.
  - The **destination** attributes are defined by the BDMA_PARx register, the PSIZE[1:0] field and PINC bit of the BDMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the destination memory in memory-to-memory mode.
- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  - The **source** attributes are defined by the BDMA_PARx register, the PSIZE[1:0] field and PINC bit of the BDMA_CCRx register.
    Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the source memory in memory-to-memory mode
  - The **destination** attributes are defined by the BDMA_MARx register, the MSIZE[1:0] field and MINC bit of the BDMA_CCRx register.

Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.

15.4.6 BDMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the BDMA controller performs some data alignments as described in *Table 112*.

Table 112. Programmable data width and endian behavior (when PINC = MINC = 1)

Source port width (MSIZE if DIR = 1, else PSIZE)	Destination port width (PSIZE if DIR = 1, else MSIZE)	Number of data items to transfer (NDT)	Source content: address / data (BDMA_CM0/1ARx if DIR = 1, else BDMA_CPARx)	DMA transfers	Destination content: address / data (BDMA_CPARx if DIR = 1, else BDMA_CM0/1ARx)
8	8	8	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write B0[7:0] @0x0 2: read B1[7:0] @0x1 then write B1[7:0] @0x1 3: read B2[7:0] @0x2 then write B2[7:0] @0x2 4: read B3[7:0] @0x3 then write B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 00B0[15:0] @0x0 2: read B1[7:0] @0x1 then write 00B1[15:0] @0x2 3: read B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[7:0] @0x3 then write 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: read B0[7:0] @0x0 then write 000000B0[31:0] @0x0 2: read B1[7:0] @0x1 then write 000000B1[31:0] @0x4 3: read B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0 2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2 3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4 4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0 2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4 3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8 4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1 3: read BBBAB9B8[31:0] @0x8 then write B8[7:0] @0x2 4: read BFBEBDBC[31:0] @0xC then write BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2 3: read BBBAB9B8[31:0] @0x8 then write B9B8[15:0] @0x4 4: read BFBEBDBC[31:0] @0xC then write BDBC[15:0] @0x6	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4 3: read BBBAB9B8[31:0] @0x8 then write BBBAB9B8[31:0] @0x8 4: read BFBEBDBC[31:0] @0xC then write BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Addressing AHB peripherals not supporting byte/half-word write transfers

When the BDMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the BDMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the BDMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in AHB master bus).
- To write the byte 0xAB, the BDMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2 or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses, is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

15.4.7 BDMA error management

A BDMA transfer error is generated when reading from or writing to a reserved address space. When a BDMA transfer error occurs during a BDMA read or write access, the faulty channel x is automatically disabled through a hardware clear of its EN bit in the corresponding BDMA_CCRx register.

The TEIFx bit of the BDMA_ISR register is set. An interrupt is then generated if the TEIE bit of the BDMA_CCRx register is set.

The EN bit of the BDMA_CCRx register can not be set again by software (channel x re-activated) until the TEIFx bit of the BDMA_ISR register is cleared (by setting the CTEIFx bit of the BDMA_IFCR register).

When the software is notified with a transfer error over a channel which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both BDMA and the peripheral in DMA mode for a new transfer.

15.5 BDMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

Table 113. BDMA interrupt requests

Interrupt request	Interrupt event	Event flag	Interrupt enable bit
Channel x interrupt	Half transfer on channel x	HTIFx	HTIEx
	Transfer complete on channel x	TCIFx	TCIEx
	Transfer error on channel x	TEIFx	TEIEx
	Half transfer or transfer complete or transfer error on channel x	GIFx	-

15.6 BDMA registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The BDMA registers have to be accessed by words (32-bit).

15.6.1 BDMA interrupt status register (BDMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the BDMA_IFCR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1	TEIF0	HTIF0	TCIF0	GIF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **TEIF7**: transfer error (TE) flag for channel 7
0: no TE event
1: a TE event occurred

Bit 30 **HTIF7**: half transfer (HT) flag for channel 7
0: no HT event
1: a HT event occurred

Bit 29 **TCIF7**: transfer complete (TC) flag for channel 7
0: no TC event
1: a TC event occurred

Bit 28 **GIF7**: global interrupt flag for channel 7
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 27 **TEIF6**: transfer error (TE) flag for channel 6
0: no TE event
1: a TE event occurred

Bit 26 **HTIF6**: half transfer (HT) flag for channel 6
0: no HT event
1: a HT event occurred

Bit 25 **TCIF6**: transfer complete (TC) flag for channel 6
0: no TC event
1: a TC event occurred

Bit 24 **GIF6**: global interrupt flag for channel 6
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 23 **TEIF5**: transfer error (TE) flag for channel 5
0: no TE event
1: a TE event occurred

Bit 22 **HTIF5**: half transfer (HT) flag for channel 5
0: no HT event
1: a HT event occurred

Bit 21 **TCIF5**: transfer complete (TC) flag for channel 5
0: no TC event
1: a TC event occurred

Bit 20 **GIF5**: global interrupt flag for channel 5
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 19 **TEIF4**: transfer error (TE) flag for channel 4
0: no TE event
1: a TE event occurred

Bit 18 **HTIF4**: half transfer (HT) flag for channel 4
0: no HT event
1: a HT event occurred

Bit 17 **TCIF4**: transfer complete (TC) flag for channel 4
0: no TC event
1: a TC event occurred

Bit 16 **GIF4**: global interrupt flag for channel 4
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 15 **TEIF3**: transfer error (TE) flag for channel 3
0: no TE event
1: a TE event occurred

Bit 14 **HTIF3**: half transfer (HT) flag for channel 3
0: no HT event
1: a HT event occurred

Bit 13 **TCIF3**: transfer complete (TC) flag for channel 3
0: no TC event
1: a TC event occurred

Bit 12 **GIF3**: global interrupt flag for channel 3
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 11 **TEIF2**: transfer error (TE) flag for channel 2
0: no TE event
1: a TE event occurred

Bit 10 **HTIF2**: half transfer (HT) flag for channel 2
0: no HT event
1: a HT event occurred

Bit 9 **TCIF2**: transfer complete (TC) flag for channel 2
0: no TC event
1: a TC event occurred

Bit 8 **GIF2**: global interrupt flag for channel 2
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 7 **TEIF1**: transfer error (TE) flag for channel 1
0: no TE event
1: a TE event occurred

Bit 6 **HTIF1**: half transfer (HT) flag for channel 1
0: no HT event
1: a HT event occurred

Bit 5 **TCIF1**: transfer complete (TC) flag for channel 1
0: no TC event
1: a TC event occurred

Bit 4 **GIF1**: global interrupt flag for channel 1
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

Bit 3 **TEIF0**: transfer error (TE) flag for channel 0
0: no TE event
1: a TE event occurred

Bit 2 **HTIF0**: half transfer (HT) flag for channel 0
0: no HT event
1: a HT event occurred

Bit 1 **TCIF0**: transfer complete (TC) flag for channel 0
0: no TC event
1: a TC event occurred

Bit 0 **GIF0**: global interrupt flag for channel 0
0: no TE, HT or TC event
1: a TE, HT or TC event occurred

15.6.2 BDMA interrupt flag clear register (BDMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

Setting the global clear bit CGIFx of the channel x in this BDMA_IFCR register, causes the BDMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the BDMA_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this BDMA_IFCR register, causes the BDMA hardware to clear the corresponding individual flag and the global flag GIFx in the BDMA_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1	CTEIF0	CHTIF0	CTCIF0	CGIF0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **CTEIF7**: transfer error flag clear for channel 7

Bit 30 **CHTIF7**: half transfer flag clear for channel 7

Bit 29 **CTCIF7**: transfer complete flag clear for channel 7

Bit 28 **CGIF7**: global interrupt flag clear for channel 7

Bit 27 **CTEIF6**: transfer error flag clear for channel 6

Bit 26 **CHTIF6**: half transfer flag clear for channel 6

Bit 25 **CTCIF6**: transfer complete flag clear for channel 6

Bit 24 **CGIF6**: global interrupt flag clear for channel 6

Bit 23 **CTEIF5**: transfer error flag clear for channel 5

Bit 22 **CHTIF5**: half transfer flag clear for channel 5

Bit 21 **CTCIF5**: transfer complete flag clear for channel 5

Bit 20 **CGIF5**: global interrupt flag clear for channel 5

Bit 19 **CTEIF4**: transfer error flag clear for channel 4

Bit 18 **CHTIF4**: half transfer flag clear for channel 4

Bit 17 **CTCIF4**: transfer complete flag clear for channel 4

Bit 16 **CGIF4**: global interrupt flag clear for channel 4

Bit 15 **CTEIF3**: transfer error flag clear for channel 3

Bit 14 **CHTIF3**: half transfer flag clear for channel 3

Bit 13 **CTCIF3**: transfer complete flag clear for channel 3

Bit 12 **CGIF3**: global interrupt flag clear for channel 3

Bit 11 **CTEIF2**: transfer error flag clear for channel 2

Bit 10 **CHTIF2**: half transfer flag clear for channe2

Bit 9 **CTCIF2**: transfer complete flag clear for channel 2

Bit 8 **CGIF2**: global interrupt flag clear for channel 2

Bit 7 **CTEIF1**: transfer error flag clear for channel 1

Bit 6 **CHTIF1**: half transfer flag clear for channel 1

Bit 5 **CTCIF1**: transfer complete flag clear for channel 1

Bit 4 **CGIF1**: global interrupt flag clear for channel 0

Bit 3 **CTEIF0**: transfer error flag clear for channel 0

Bit 2 **CHTIF0**: half transfer flag clear for channel 0

Bit 1 **CTCIF0**: transfer complete flag clear for channel 0

Bit 0 **CGIF0**: global interrupt flag clear for channel 0

15.6.3 BDMA channel x configuration register (BDMA_CCRx)

Address offset: 0x08 + 0x14 * x, (x = 0 to 7)

Reset value: 0x0000 0000

The register fields/bits CT, DBM, MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CT**: current target memory of DMA transfer in double-buffer mode

This bit is toggled by hardware at the end of each channel transfer in double-buffer mode.

0: memory 0 (addressed by the BDMA_CM0AR pointer)

1: memory 1 (addressed by the BDMA_CM1AR pointer)

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 15 **DBM**: double-buffer mode

This bit must be set only in memory-to-peripheral and peripheral-to-memory transfers (MEM2MEM=0). The CIRC bit must also be set in double buffer mode.

0: disabled (no memory address switch at the end of the BDMA transfer)

1: enabled (memory address switched at the end of the BDMA transfer)

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 14 **MEM2MEM**: memory-to-memory mode

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 13:12 **PL[1:0]**: priority level

00: low

01: medium

10: high

11: very high

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 11:10 **MSIZE[1:0]**: memory size

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bits 9:8 **PSIZE[1:0]**: peripheral size

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 7 **MINC**: memory increment mode

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this field identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 6 **PINC**: peripheral increment mode

Defines the increment mode for each DMA transfer to the identified peripheral.

n memory-to-memory mode, this field identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this field identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 5 **CIRC**: circular mode

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 4 **DIR**: data transfer direction

This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.

0: read from peripheral

- Source attributes are defined by PSIZE and PINC, plus the BDMA_CPARx register. This is still valid in a memory-to-memory mode.
- Destination attributes are defined by MSIZE and MINC, plus the BDMA_CM0/1ARx register. This is still valid in a peripheral-to-peripheral mode.

1: read from memory

- Destination attributes are defined by PSIZE and PINC, plus the BDMA_CPARx register. This is still valid in a memory-to-memory mode.
- Source attributes are defined by MSIZE and MINC, plus the BDMA_CM0/1ARx register. This is still valid in a peripheral-to-peripheral mode.

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

Bit 3 **TEIE**: transfer error interrupt enable

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 2 **HTIE**: half transfer interrupt enable

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 1 **TCIE**: transfer complete interrupt enable

0: disabled

1: enabled

Note: this bit is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

Bit 0 **EN**: channel enable

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the BDMA_ISR register is cleared (by setting the CTEIFx bit of the BDMA_IFCR register).

0: disabled

1: enabled

Note: this bit is set and cleared by software.

15.6.4 BDMA channel x number of data to transfer register (BDMA_CNDTRx)

Address offset: 0x0C + 0x14 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]**: number of data to transfer (0 to 2^{16} - 1)

This field is updated by hardware when the channel is enabled:

- It is decremented after each single BDMA 'read followed by write' transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the BDMA_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this field is zero, no transfer can be served whatever the channel status (enabled or not).

Note: this field is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is read-only when the channel is enabled (EN = 1).

15.6.5 BDMA channel x peripheral address register (BDMA_CPARx)

Address offset: 0x10 + 0x14 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PA[31:0]**: peripheral address

It contains the base address of the peripheral data register from/to which the data will be read/written.

When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.

When PSIZE = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral destination address DIR = 1 and the peripheral source address if DIR = 0.

Note: this register is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

15.6.6 BDMA channel x memory 0 address register (BDMA_CM0ARx)

Address offset: 0x14 + 0x14 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MA[31:0]**: peripheral address

It contains the base address of the memory from/to which the data will be read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral source address DIR = 1 and the peripheral destination address if DIR = 0.

Note: this register is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

15.6.7 BDMA channel x memory 1 address register (BDMA_CM1ARx)

Address offset: 0x18 + 0x14 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MA[31:0]**: peripheral address

It contains the base address of the memory from/to which the data will be read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this register identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this register identifies the peripheral source address DIR = 1 and the peripheral destination address if DIR = 0.

Note: this register is set and cleared by software.
It must not be written when the channel is enabled (EN = 1).
It is not read-only when the channel is enabled (EN = 1).

15.6.8 BDMA register map and reset values

Table 114 gives the BDMA register map and reset values.

Table 114. BDMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	BDMA_ISR	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1	TEIF0	HTIF0	TCIF0	GIF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	BDMA_IFCR	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1	CTEIF0	CHTIF0	CTCIF0	CGIF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	BDMA_CCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	BDMA_CNDTR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	BDMA_CPAR0	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	BDMA_CM0AR0	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	BDMA_CM1AR0	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 114. BDMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	BDMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	BDMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	BDMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	BDMA_CM0AR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	BDMA_CM1AR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	BDMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	BDMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	BDMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	BDMA_CM0AR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	BDMA_CM1AR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	BDMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	BDMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	BDMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	BDMA_CM0AR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	BDMA_CM1AR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	BDMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	BDMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	BDMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	BDMA_CM0AR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	BDMA_CM1AR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 114. BDMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x06C	BDMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	BDMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	BDMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	BDMA_CM0AR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	BDMA_CM1AR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	BDMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	BDMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	BDMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	BDMA_CM0AR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	BDMA_CM1AR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	BDMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CT	DBM	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098	BDMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	BDMA_CPAR7	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	BDMA_CM0AR7	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	BDMA_CM1AR7	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2* for register boundary addresses.

16 DMA request multiplexer (DMAMUX)

16.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until it is served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is de-asserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controllers of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in *Section 16.3.1*.

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend on the product implementation, and are detailed in *Section 16.3.2*.

16.2 DMAMUX main features

- Up to 16-channel programmable DMA request line multiplexer output
- Up to 8-channel DMA request generator
- Up to 32 trigger inputs to DMA request generator
- Up to 16 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - Up to 107 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining

16.3 DMAMUX implementation

16.3.1 DMAMUX1 and DMAMUX2 instantiation

The product integrates two instances of DMA request multiplexer:

- DMAMUX1 for DMA1 and DMA2 (D2 domain)
- DMAMUX2 for BDMA (D3 domain)

DMAMUX1 and DMAMUX2 are instantiated with the hardware configuration parameters listed in the following table.

Table 115. DMAMUX1 and DMAMUX2 instantiation

Feature	DMAMUX1	DMAMUX2
Number of DMAMUX output request channels	16	8
Number of DMAMUX request generator channels	8	8
Number of DMAMUX request trigger inputs	8	32
Number of DMAMUX synchronization inputs	8	16
Number of DMAMUX peripheral request inputs	107	12

16.3.2 DMAMUX1 mapping

The mapping of resources to DMAMUX1 is hardwired.

DMAMUX1 is used with the DMA1 and DMA2 in D2 domain:

- DMAMUX1 channels 0 to 7 are connected to DMA1 channels 0 to 7
- DMAMUX1 channels 8 to 15 are connected to DMA2 channels 0 to 7

Table 116. DMAMUX1: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux1_req_gen0	44	usart2_tx_dma	87	sai1a_dma
2	dmamux1_req_gen1	45	usart3_rx_dma	88	sai1b_dma
3	dmamux1_req_gen2	46	usart3_tx_dma	89	sai2a_dma
4	dmamux1_req_gen3	47	TIM8_CH1	90	sai2b_dma
5	dmamux1_req_gen4	48	TIM8_CH2	91	swpmi_rx_dma
6	dmamux1_req_gen5	49	TIM8_CH3	92	swpmi_tx_dma
7	dmamux1_req_gen6	50	TIM8_CH4	93	spdifrx_dat_dma
8	dmamux1_req_gen7	51	TIM8_UP	94	spdifrx_ctrl_dma
9	adc1_dma	52	TIM8_TRIG	95	HR_REQ(1)
10	adc2_dma	53	TIM8_COM	96	HR_REQ(2)
11	TIM1_CH1	54	Reserved	97	HR_REQ(3)
12	TIM1_CH2	55	TIM5_CH1	98	HR_REQ(4)
13	TIM1_CH3	56	TIM5_CH2	99	HR_REQ(5)

Table 116. DMAMUX1: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
14	TIM1_CH4	57	TIM5_CH3	100	HR_REQ(6)
15	TIM1_UP	58	TIM5_CH4	101	dfsdm1_dma0
16	TIM1_TRIG	59	TIM5_UP	102	dfsdm1_dma1
17	TIM1_COM	60	TIM5_TRIG	103	dfsdm1_dma2
18	TIM2_CH1	61	spi3_rx_dma	104	dfsdm1_dma3
19	TIM2_CH2	62	spi3_tx_dma	105	TIM15_CH1
20	TIM2_CH3	63	uart4_rx_dma	106	TIM15_UP
21	TIM2_CH4	64	uart4_tx_dma	107	TIM15_TRIG
22	TIM2_UP	65	uart5_rx_dma	108	TIM15_COM
23	TIM3_CH1	66	uart5_tx_dma	109	TIM16_CH1
24	TIM3_CH2	67	dac_ch1_dma	110	TIM16_UP
25	TIM3_CH3	68	dac_ch2_dma	111	TIM17_CH1
26	TIM3_CH4	69	TIM6_UP	112	TIM17_UP
27	TIM3_UP	70	TIM7_UP	113	sai3_a_dma
28	TIM3_TRIG	71	usart6_rx_dma	114	sai3_b_dma
29	TIM4_CH1	72	usart6_tx_dma	115	adc3_dma
30	TIM4_CH2	73	i2c3_rx_dma	116	Reserved
31	TIM4_CH3	74	i2c3_tx_dma	117	Reserved
32	TIM4_UP	75	dcmi_dma	118	Reserved
33	i2c1_rx_dma	76	cryp_in_dma	119	Reserved
34	i2c1_tx_dma	77	cryp_out_dma	120	Reserved
35	i2c2_rx_dma	78	hash_in_dma	121	Reserved
36	i2c2_tx_dma	79	uart7_rx_dma	122	Reserved
37	spi1_rx_dma	80	uart7_tx_dma	123	Reserved
38	spi1_tx_dma	81	uart8_rx_dma	124	Reserved
39	spi2_rx_dma	82	uart8_tx_dma	125	Reserved
40	spi2_tx_dma	83	spi4_rx_dma	126	Reserved
41	usart1_rx_dma	84	spi4_tx_dma	127	Reserved
42	usart1_tx_dma	85	spi5_rx_dma	-	-
43	usart2_rx_dma	86	spi5_tx_dma	-	-

Table 117. DMAMUX1: assignment of trigger inputs to resources

Trigger input	Resource	Trigger input	Resource
0	dmamux1_evt0	4	lptim2_out
1	dmamux1_evt1	5	lptim3_out

Table 117. DMAMUX1: assignment of trigger inputs to resources (continued)

Trigger input	Resource	Trigger input	Resource
2	dmamux1_evt2	6	extit0
3	lptim1_out	7	TIM12_TRGO

Table 118. DMAMUX1: assignment of synchronization inputs to resources

Sync. input	Resource	Sync. input	Resource
0	dmamux1_evt0	4	lptim2_out
1	dmamux1_evt1	5	lptim3_out
2	dmamux1_evt2	6	extit0
3	lptim1_out	7	TIM12_TRGO

16.3.3 DMAMUX2 mapping

DMAMUX2 channels 0 to 7 are connected to BDMA channels 0 to 7.

Table 119. DMAMUX2: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux2_req_gen0	17	adc3_dma
2	dmamux2_req_gen1	18	Reserved
3	dmamux2_req_gen2	19	Reserved
4	dmamux2_req_gen3	20	Reserved
5	dmamux2_req_gen4	21	Reserved
6	dmamux2_req_gen5	22	Reserved
7	dmamux2_req_gen6	23	Reserved
8	dmamux2_req_gen7	24	Reserved
9	lpuart1_rx_dma	25	Reserved
10	lpuart1_tx_dma	26	Reserved
11	spi6_rx_dma	27	Reserved
12	spi6_tx_dma	28	Reserved
13	i2c4_rx_dma	29	Reserved
14	i2c4_tx_dma	30	Reserved
15	sai4_a_dma	31	Reserved
16	sai4_b_dma	32	Reserved

Table 120. DMAMUX2: assignment of trigger inputs to resources

Trigger input	Resource	Trigger input	Resource
0	dmamux2_evt0	16	spi6_wkup
1	dmamux2_evt1	17	Comp1_out
2	dmamux2_evt2	18	Comp2_out
3	dmamux2_evt3	19	RTC_wkup
4	dmamux2_evt4	20	Syscfg_exti0_mux
5	dmamux2_evt5	21	Syscfg_exti2_mux
6	dmamux2_evt6	22	I2c4_event_it
7	lpuart_rx_wkup	23	spi6_it
8	lpuart_tx_wkup	24	Lpuart1_it_T
9	lptim2_wkup	25	Lpuart1_it_R
10	lptim2_out	26	adc3_it
11	lptim3_wkup	27	adc3_awd1
12	lptim3_out	28	bdma_ch0_it
13	Lptim4_ait	29	bdma_ch1_it
14	Lptim5_ait	30	Reserved
15	I2c4_wkup	31	Reserved

Table 121. DMAMUX2: assignment of synchronization inputs to resources

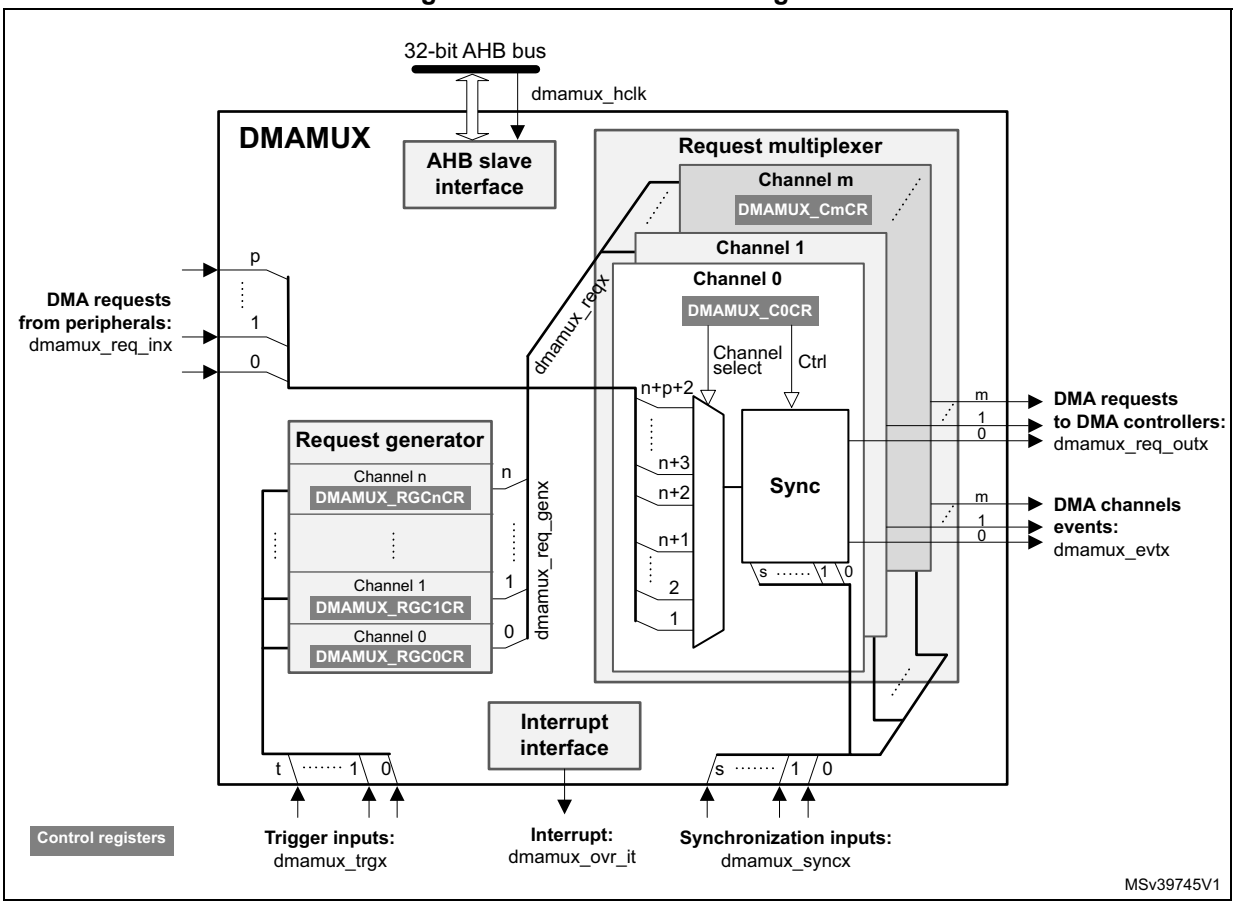
Sync input	Resource	Sync input	Resource
0	dmamux2_evt0	16	Reserved
1	dmamux2_evt1	17	Reserved
2	dmamux2_evt2	18	Reserved
3	dmamux2_evt3	19	Reserved
4	dmamux2_evt4	20	Reserved
5	dmamux2_evt5	21	Reserved
6	lpuart1_rx_wkup	22	Reserved
7	lpuart1_tx_wkup	23	Reserved
8	Lptim2_out	24	Reserved
9	Lptim3_out	25	Reserved
10	I2c4_wkup	26	Reserved
11	spi6_wkup	27	Reserved
12	Comp1_out	28	Reserved
13	RTC_wkup	29	Reserved
14	Syscfg_exti0_mux	30	Reserved
15	Syscfg_exti2_mux	31	Reserved

16.4 DMAMUX functional description

16.4.1 DMAMUX block diagram

Figure 80 shows the DMAMUX block diagram.

Figure 80. DMAMUX block diagram



DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) and from channels of the DMAMUX request generator sub-block (dmamux_req_genx)
- DMAMUX request outputs to channels of DMA controllers (dmamux_req_outx)
- Internal or external signals to DMA request trigger inputs (dmamux_trgx)
- Internal or external signals to synchronization inputs (dmamux_syncx)

16.4.2 DMAMUX signals

Table 122 lists the DMAMUX signals.

Table 122. DMAMUX signals

Signal name	Description
dmamux_hclk	DMAMUX AHB clock
dmamux_req_inx	DMAMUX DMA request line inputs from peripherals
dmamux_trgx	DMAMUX DMA request triggers inputs (to request generator sub-block)
dmamux_req_genx	DMAMUX request generator sub-block channels outputs
dmamux_reqx	DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)
dmamux_syncx	DMAMUX synchronization inputs (to request multiplexer sub-block)
dmamux_req_outx	DMAMUX requests outputs (to DMA controllers)
dmamux_evtx	DMAMUX events outputs
dmamux_ovr_it	DMAMUX overrun interrupts

16.4.3 DMAMUX channels

A DMAMUX channel is a DMAMUX request multiplexer channel that may include, depending on the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to one single channel of DMA controller(s).

Channel configuration procedure

The following sequence should be followed to configure both a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

16.4.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ_ID field in the DMAMUX_CxCR register.

Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected. It is not allowed to configure a same non-null DMAREQ_ID to two different channels of the DMAMUX request line multiplexer.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC_ID field in the DMAMUX_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once is detected a programmable rising/falling edge on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, there is a programmable DMA request counter, internally to the DMAMUX request multiplexer, which may be used for the channel request output generation and also possibly for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX_CxCR register.

As shown in *Figure 82*, upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

Note: If a synchronization event occurs while there is no pending selected input DMA request line, it will be discarded. The following asserted input request lines will not be connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is de-asserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in NBREQ field of the DMAMUX_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in NBREQ field, plus one.

Note: The NBREQ field value shall only be written by software when both synchronization enable bit SE and event generation enable EGE bit of the corresponding multiplexer channel x are disabled.

Figure 81. Synchronization mode of the DMAMUX request line multiplexer channel

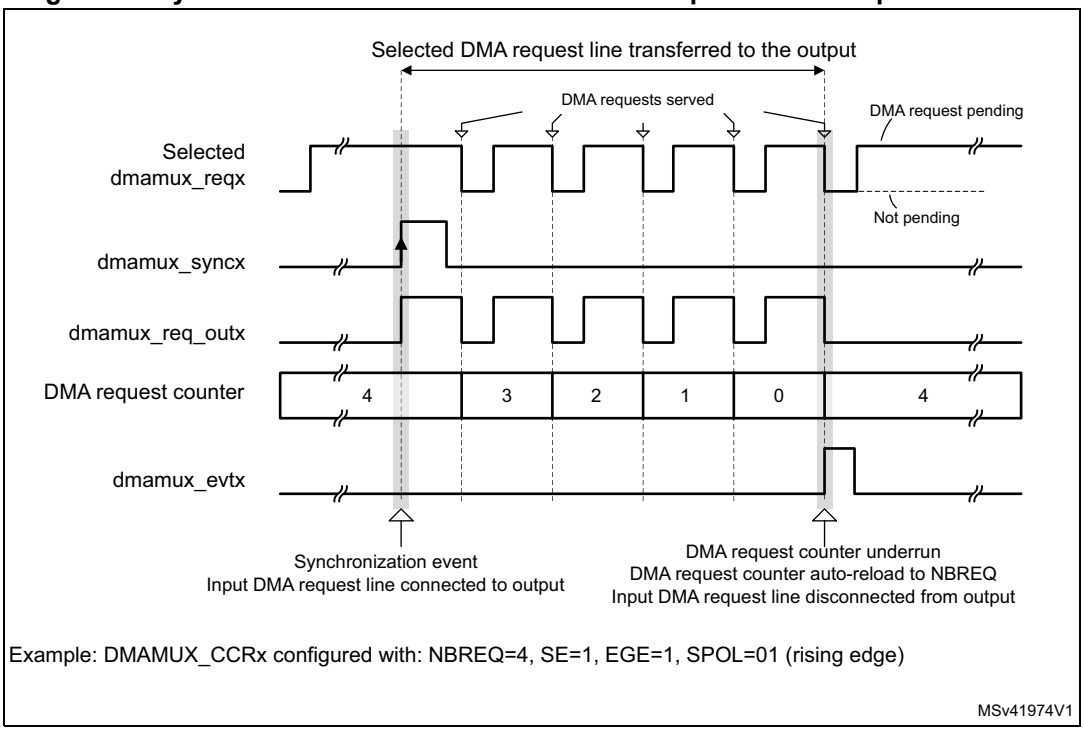
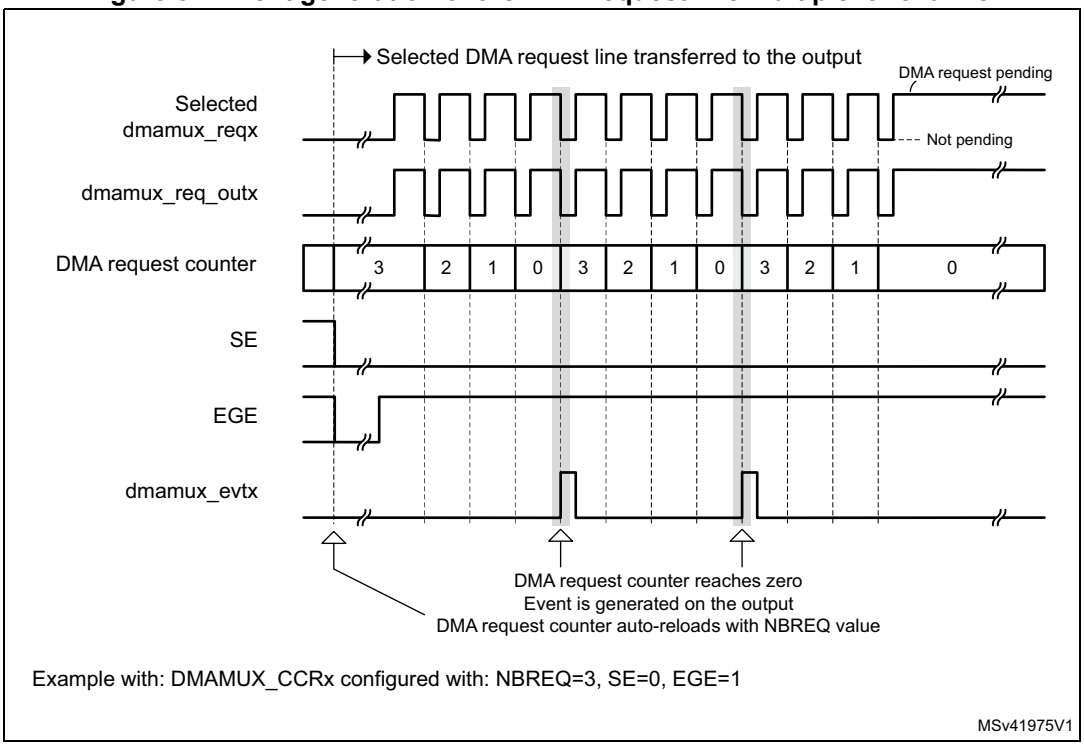


Figure 82. Event generation of the DMA request line multiplexer channel



If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in *Figure 81* and *Figure 82*.

Note: If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.

Note: A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_CxCR register, the synchronization events are masked during three AHB clock cycles.

Synchronization overrun and interrupt

If a new synchronization event occurs while the value of the DMA request counter is lower than the programmed NBREQ field value of the multiplexer channel x, the synchronization overrun flag bit SOFx is set in the DMAMUX_CSR status register.

Note: The request multiplexer channel x synchronization shall be disabled (DMAMUX_CxCR.SE = 0) at the completion of the use of the related channel of the DMA controller. Else, upon a new detected synchronization event, there will be a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

16.4.5 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG_ID (trigger signal ID) field in the corresponding DMAMUX_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is de-asserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter will be automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ+1.

Note: The GNBREQ field value shall only be written by software when the enable GE bit of the corresponding generator channel x is disabled.

A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_RGxCR register, the trigger events are masked during three AHB clock cycles.

Trigger overrun and interrupt

If a new DMA request trigger event occurs while the value of the DMAMUX request generator counter is lower than the programmed GNBREQ field value of the corresponding request generator channel x, and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the status DMAMUX_RGSR register.

Note: The request generator channel x shall be disabled (DMAMUX_RGxCR.GE = 0) at the completion of the usage of the related channel of the DMA controller. Else, upon a new detected trigger event, there will be a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX_RGxCR register.

16.5 DMAMUX interrupts

An interrupt can be generated upon:

- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel individual interrupt enable, status and clear flag register bits are available.

Table 123. DMAMUX interrupts

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamuxovr_it	Synchronization event overrun on channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on channel x of the DMAMUX request generator	OFx	COFx	OIE

16.6 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX1 and DMAMUX2 base address.

DMAMUX registers may be accessed per (8-bit) byte, (16-bit) half-word, or (32-bit) word. The address shall be aligned with the data size.

16.6.1 DMAMUX1 request line multiplexer channel x configuration register (DMAMUX1_CxCR)

Address offset: 0x04 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
						rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **SYNC_ID[2:0]**: Synchronization identification

Selects the synchronization input (see *Table 118: DMAMUX1: assignment of synchronization inputs to resources*).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.

This field shall only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:

00: no event, i.e. no synchronization nor detection.

01: rising edge

10: falling edge

11: rising and falling edge

Bit 16 **SE**: Synchronization enable

0: synchronization disabled

1: synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable

0: event generation disabled

1: event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: interrupt disabled

1: interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DMAREQ_ID[6:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

16.6.2 DMAMUX2 request line multiplexer channel x configuration register (DMAMUX2_CxCR)

Address offset: 0x04 * x, where x = 0 to 7

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SYNC_ID[4:0]					NBREQ[4:0]					SPOL[1:0]		SE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	Res.	Res.	DMAREQ_ID[4:0]				
						rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC_ID[4:0]**: Synchronization identification

Selects the synchronization input. (C.f. table: DMAMUX- assignments of synchronization inputs to resources)

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.
This field shall only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:
00: no event, i.e. no synchronization nor detection.
01: rising edge
10: falling edge
11: rising and falling edges

Bit 16 **SE**: Synchronization enable

0: synchronization disabled
1: synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable

0: event generation disabled
1: event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: interrupt disabled
1: interrupt enabled

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DMAREQ_ID[4:0]**: DMA request identification

Selects the input DMA request. C.f. the DMAMUX table about assignments of multiplexer inputs to resources.

16.6.3 DMAMUX1 request line multiplexer interrupt channel status register (DMAMUX1_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOF15	SOF14	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SOF[15:0]**: Synchronization overrun event flag

The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.
The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX_CFR register.
For DMAMUX2_CFR bits 15:8 are reserved.

16.6.4 DMAMUX2 request line multiplexer interrupt channel status register (DMAMUX2_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

This register shall be accessed at bit level by a non-secure or secure read, according to the secure mode of the considered DMAMUX2 request line multiplexer channel x, depending on the secure control bit of the connected DMA controller channel y, and considering that the DMAMUX2 x channel output is connected to the y channel of the DMA (refer to the DMAMXUX2 mapping implementation section).

This register shall be accessed at bit level by an unprivileged or privileged read, according to the privileged mode of the considered DMAMUX2 request line multiplexer channel x, depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX2 x channel output is connected to the y channel of the DMA (refer to the DMAMXUX2 mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SOF[7:0]**: Synchronization overrun event flag

The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.
The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX2_CFR register.

16.6.5 DMAMUX1 request line multiplexer interrupt clear flag register (DMAMUX1_CFR)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSOF 15	CSOF 14	CSOF 13	CSOF 12	CSOF 11	CSOF 10	CSOF 9	CSOF 8	CSOF 7	CSOF 6	CSOF 5	CSOF 4	CSOF 3	CSOF 2	CSOF 1	CSOF 0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSOF[15:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.

16.6.6 DMAMUX2 request line multiplexer interrupt clear flag register (DMAMUX2_CFR)

Address offset: 0x084

Reset value: 0x0000 0000

This register shall be written at bit level by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel x, depending on the secure control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

This register shall be written at bit level by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel x, depending on the privileged control bit of the connected DMA controller channel y, and considering that the DMAMUX x channel output is connected to the y channel of the DMA (refer to the DMAMXUX mapping implementation section).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSOF6	CSOF6	CSOF5	CSOF4	CSOF3	CSOF2	CSOF1	CSOF0
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CSOF[7:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX2_CSR register.

16.6.7 DMAMUX1 request generator channel x configuration register (DMAMUX1_RGxCR)

Address offset: 0x100 + 0x04 * (x-0) (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID[2:0]		
							rw						rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ+1.

Note: This field shall only be written when GE bit is disabled.

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input
00: no event. I.e. none trigger detection nor generation.
01: rising edge
10: falling edge
11: rising and falling edge

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled
1: DMA request generator channel x enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OIE**: Trigger overrun interrupt enable

0: interrupt on a trigger overrun event occurrence is disabled
1: interrupt on a trigger overrun event occurrence is enabled

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **SIG_ID[2:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

16.6.8 DMAMUX2 request generator channel x configuration register (DMAMUX2_RGxCR)

Address offset: 0x100 + 0x04 * (x-0) (x = 0 to 7)

Reset value: 0x0000 0000

This register shall be written by a non-secure or secure write, according to the secure mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (see the DMAREQ_ID field of the DMAMUX2_CyCR register and the DMAMUX mapping implementation section).

This register shall be written by an unprivileged or privileged write, according to the privileged mode of the considered DMAMUX request line multiplexer channel y it is assigned to, and considering that the DMAMUX request generator x channel output is selected by the y channel of the DMAMUX request line channel (see the DMAREQ_ID field of the DMAMUX2_CyCR register and the DMAMXUX mapping implementation section).

Note: With the request generator channel x disabled (GE=0) and thus ready to be configured again, the software shall operate (at least) two separated write accesses to this register. First, set/update the number of DMA requests to be generated (bits GNBREQ[4:0]), enable the request generator (i.e. set bit 16 GE), and disable the detection of the generator trigger input (i.e. set GPOL[1:0] to 00). Next, enable the detection of the generator trigger input by selecting as rising/falling the edge to be detected (i.e. set GPOL[1:0] to 01, 10 or 11).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
							rw				rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ+1.

Note: This field shall only be written when GE bit is disabled.

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input
00: no event. I.e. none trigger detection nor generation.
01: rising edge
10: falling edge
11: rising and falling edge

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled
1: DMA request generator channel x enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OIE**: Trigger overrun interrupt enable

0: interrupt on a trigger overrun event occurrence is disabled
1: interrupt on a trigger overrun event occurrence is enabled

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **SIG_ID[4:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

16.6.9 DMAMUX1 request generator interrupt status register (DMAMUX1_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF7	OF6	OF5	OF4	OF3	OF2	OF1	OF0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **OF[7:0]**: Trigger overrun event flag

The flag is set when a trigger event occurs on DMA request generator channel x, while the DMA request generator counter value is lower than GNBREQ.

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

16.6.10 DMAMUX2 request generator interrupt status register (DMAMUX2_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF7	OF6	OF5	OF4	OF3	OF2	OF1	OF0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **OF[7:0]**: Trigger overrun event flag

The flag is set when a trigger event occurs on DMA request generator channel x, while the DMA request generator counter value is lower than GNBREQ.

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX2_RGCFR register.

16.6.11 DMAMUX1 request generator interrupt clear flag register (DMAMUX1_RGCFR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF7	COF6	COF5	COF4	COF3	COF2	COF1	COF0
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **COF[7:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.

16.6.12 DMAMUX2 request generator interrupt clear flag register (DMAMUX2_RGCFR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF7	COF6	COF5	COF4	COF3	COF2	COF1	COF0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **COF[7:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX2_RGSR register.

16.6.13 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

Table 124. DMAMUX register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMAMUX_C0CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x004	DMAMUX_C1CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x008	DMAMUX_C2CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x00C	DMAMUX_C3CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x010	DMAMUX_C4CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x014	DMAMUX_C5CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x018	DMAMUX_C6CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x01C	DMAMUX_C7CR[1][2]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x020	DMAMUX_C8CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x024	DMAMUX_C9CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x028	DMAMUX_C10CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x02C	DMAMUX_C11CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x030	DMAMUX_C12CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x034	DMAMUX_C13CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x038	DMAMUX_C14CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x03C	DMAMUX_C15CR[3]	Res.	Res.	Res.	Res.	Res.	SYNC_ID [2:0]			NBREQ[4:0]					SPOL [1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x040 - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 124. DMAMUX register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	DMAMUX_CSR(4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOF15	SOF14	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMAMUX_CFR(4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSOF15	CSOF14	CSOF13	CSOF12	CSOF11	CSOF10	CSOF9	CSOF8	CSOF7	CSOF6	CSOF5	CSOF4	CSOF3	CSOF2	CSOF1	CSOF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088 - 0x0FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	DMAMUX_RG0CR(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [2:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x104	DMAMUX_RG1CR(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [2:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x108	DMAMUX_RG2CR(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [2:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x10C	DMAMUX_RG3CR(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [2:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x140	DMAMUX_RGSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF15	OF14	OF13	OF12	OF11	OF10	OF9	OF8	OF7	OF6	OF5	OF4	OF3	OF2	OF1	OF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144	DMAMUX_RGCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF15	COF14	COF13	COF12	COF11	COF10	COF9	COF8	COF7	COF6	COF5	COF4	COF3	COF2	COF1	COF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	DMAMUX_RGSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF7	OF6	OF5	OF4	OF3	OF2	OF1	OF0
	Reset value																									0	0	0	0	0	0	0	0
0x144	DMAMUX_RGCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF7	COF6	COF5	COF4	COF3	COF2	COF1	COF0
	Reset value																									0	0	0	0	0	0	0	0
0x148 - 0x3FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

1. For DMAMUX2 bits 6:5 are reserved.
2. For DMAMUX2 bits 28:24 are for SYNC_ID[4:0]
3. Only applies to DMAMUX1. For DMAMUX2 the word is reserved.
4. For DMAMUX2 bits 15:8 are reserved.
5. Valid for DMAMUX1. For DMAMUX2 bits 4:0 are for SIG_ID[4:0].

17 Chrom-Art Accelerator™ controller (DMA2D)

17.1 DMA2D introduction

The Chrom-Art Accelerator™ (DMA2D) is a specialized DMA dedicated to image manipulation. It can perform the following operations:

- Filling a part or the whole of a destination image with a specific color
- Copying a part or the whole of a source image into a part or the whole of a destination image
- Copying a part or the whole of a source image into a part or the whole of a destination image with a pixel format conversion
- Blending a part and/or two complete source images with different pixel format and copy the result into a part or the whole of a destination image with a different color format.

All the classical color coding schemes are supported from 4-bit up to 32-bit per pixel with indexed or direct color mode, including block based YCbCr to handle JPEG decoder output. The DMA2D has its own dedicated memories for CLUTs (color look-up tables).

17.2 DMA2D main features

The main DMA2D features are:

- Single AXI master bus architecture.
- AHB slave programming interface supporting 8/16/32-bit accesses (except for CLUT accesses which are 32-bit).
- User programmable working area size
- User programmable offset for sources and destination areas
- User programmable sources and destination addresses on the whole memory space
- Up to 2 sources with blending operation
- Alpha value can be modified (source value, fixed value or modulated value)
- User programmable source and destination color format
- Up to 11 color formats supported from 4-bit up to 32-bit per pixel with indirect or direct color coding
- Block based (8x8) YCbCr support with 4:4:4, 4:2:2 and 4:2:0 chroma sub-sampling factors
- 2 internal memories for CLUT storage in indirect color mode
- Automatic CLUT loading or CLUT programming via the CPU
- User programmable CLUT size
- Internal timer to control AXI bandwidth
- 6 operating modes: register-to-memory, memory-to-memory, memory-to-memory with pixel format conversion, memory-to-memory with pixel format conversion and blending, memory-to memory with pixel format conversion, blending and fixed color foreground,

and memory-to memory with pixel format conversion, blending and fixed color background.

- Area filling with a fixed color
- Copy from an area to another
- Copy with pixel format conversion between source and destination images
- Copy from two sources with independent color format and blending
- Output buffer byte swapping to support refresh of displays through parallel interface
- Abort and suspend of DMA2D operations
- Watermark interrupt on a user programmable destination line
- Interrupt generation on bus error or access conflict
- Interrupt generation on process completion

17.3 DMA2D functional description

17.3.1 General description

The DMA2D controller performs direct memory transfer. As an AXI master, it can take the control of the AXI bus matrix to initiate AXI transactions.

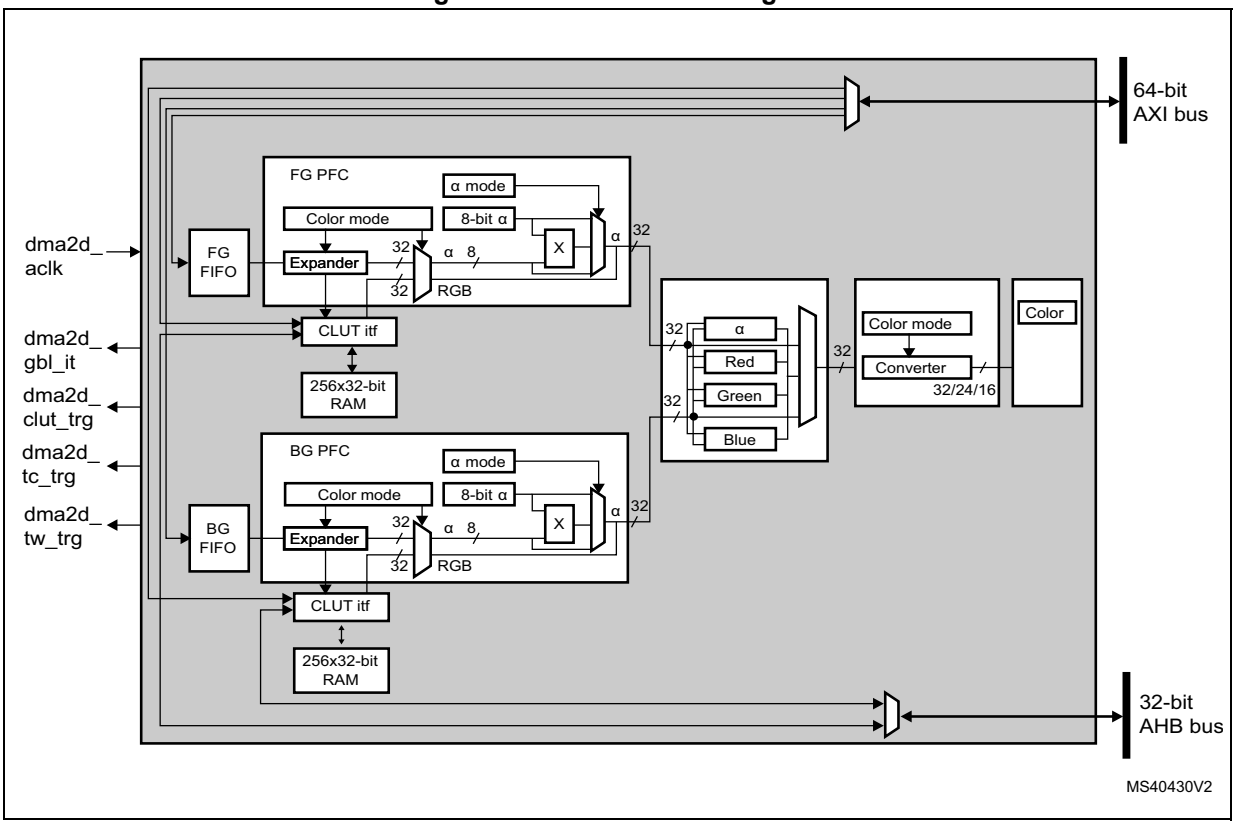
The DMA2D can operate in the following modes:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with Pixel Format Conversion
- Memory-to-memory with Pixel Format Conversion and Blending
- Memory-to memory with pixel format conversion, blending and fixed color foreground

The AHB slave port is used to program the DMA2D controller.

The block diagram of the DMA2D is shown in *Figure 83: DMA2D block diagram*.

Figure 83. DMA2D block diagram



17.4 DMA2D pins and internal signals

Table 125 lists the DMA2D internal signals.

Table 125. DMA2D internal input/output signals

Signal name	Signal type	Description
dma2d_aclk	Input	32-bit AXI bus clock
dma2d_gbl_it	Output	DMA2D global interrupt
dma2d_clut_trg	Output	CLUT transfer complete (to MDMA)
dma2d_tc_trg	Output	Transfer complete (to MDMA)
dma2d_tw_trg	Output	Transfer watermark (to MDMA)

17.4.1 DMA2D control

The DMA2D controller is configured through the DMA2D Control Register (DMA2D_CR) which allows selecting:

The user application can perform the following operations:

- Select the operating mode
- Enable/disable the DMA2D interrupt
- Start/suspend/abort ongoing data transfers

17.4.2 DMA2D foreground and background FIFOs

The DMA2D foreground (FG) FG FIFO and background (BG) FIFO fetch the input data to be copied and/or processed.

The FIFOs fetch the pixels according to the color format defined in their respective pixel format converter (PFC).

They are programmed through a set of control registers:

- DMA2D foreground memory address register (DMA2D_FGMAR)
- DMA2D foreground offset register (DMA2D_FGOR)
- DMA2D background memory address register (DMA2D_BGMAR)
- DMA2D background offset register (DMA2D_BGBOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR)

When the DMA2D operates in register-to-memory mode, none of the FIFOs is activated.

When the DMA2D operates in memory-to-memory mode (no pixel format conversion nor blending operation), only the FG FIFO is activated and acts as a buffer.

When the DMA2D operates in memory-to-memory operation with pixel format conversion (no blending operation), the BG FIFO is not activated.

17.4.3 DMA2D foreground and background pixel format converter (PFC)

DMA2D foreground pixel format converter (PFC) and background pixel format converter perform the pixel format conversion to generate a 32-bit per pixel value. The PFC can also modify the alpha channel.

The first stage of the converter converts the color format. The original color format of the foreground pixel and background pixels are configured through the CM[3:0] bits of the DMA2D_FGPFCCR and DMA2D_BGPFCCR, respectively.

The supported input formats are given in *Table 126: Supported color mode in input*.

Table 126. Supported color mode in input

CM[3:0]	Color mode
0000	ARGB8888
0001	RGB888
0010	RGB565
0011	ARGB1555
0100	ARGB4444

Table 126. Supported color mode in input

CM[3:0]	Color mode
0101	L8
0110	AL44
0111	AL88
1000	L4
1001	A8
1010	A4
1011	YCbCr (only for foreground)

The color format are coded as follows:

- Alpha value field: transparency
  0xFF value corresponds to an opaque pixel and 0x00 to a transparent one.
- R field for Red
- G field for Green
- B field for Blue
- L field: luminance
  This field is the index to a CLUT to retrieve the three/four RGB/ARGB components.

If the original format was direct color mode, then the extension to 8-bit per channel is performed by copying the MSBs into the LSBs. This ensures a perfect linearity of the conversion.

If the original format does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

If the original format is indirect color mode, a CLUT is required and each pixel format converter is associated with a 256 entry 32-bit CLUT.

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and is defined in the DMA2D_FGCOLR for foreground pixels and in the DMA2D_BGCOLR register for background pixels.

The order of the fields in the system memory is defined in *Table 127: Data order in memory*.

Table 127. Data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]
L8	L_3[7:0]	L_2[7:0]	L_1[7:0]	L_0[7:0]
AL44	A_3[3:0]L_3[3:0]	A_2[3:0]L_2[3:0]	A_1[3:0]L_1[3:0]	A_0[3:0]L_0[3:0]
AL88	A_1[7:0]	L_1[7:0]	A_0[7:0]	L_0[7:0]
L4	L_7[3:0]L_6[3:0]	L_5[3:0]L_4[3:0]	L_3[3:0]L_2[3:0]	L_1[3:0]L_0[3:0]
A8	A_3[7:0]	A_2[7:0]	A_1[7:0]	A_0[7:0]
A4	A_7[3:0]A_6[3:0]	A_5[3:0]A_4[3:0]	A_3[3:0]A_2[3:0]	A_1[3:0]A_0[3:0]

The 24-bit RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

Once the 32-bit value is generated, the alpha channel can be modified according to the AM[1:0] field of the DMA2D_FGPFCCR/DMA2D_BGPFCCR registers as shown in *Table 128: Alpha mode configuration*.

The alpha channel can be:

- kept as it is (no modification),
- replaced by the ALPHA[7:0] value of DMA2D_FGPFCCR/DMA2D_BGPFCCR,
- or replaced by the original alpha value multiplied by the ALPHA[7:0] value of DMA2D_FGPFCCR/DMA2D_BGPFCCR divided by 255.

Table 128. Alpha mode configuration

AM[1:0]	Alpha mode
00	No modification
01	Replaced by value in DMA2D_xxPFCCR
10	Replaced by original value multiplied by the value in DMA2D_xxPFCCR / 255
11	Reserved

Note: To support the alternate format, the incoming alpha value can be inverted setting the AI bit of the DMA2D_FGPFCCR/DMA2D_BGPFCCR registers. This applies also to the Alpha value stored in the DMA2D_FGPFCCR/DMA2D_BGPFCCR and in the CLUT.

The R and B fields can also be swapped setting the RBS bit of the DMA2D_FGPFCCR/DMA2D_BGPFCCR registers. This applies also to the RGB order used in the CLUT and in the DMA2D_FGCOLR/DMA2D_BGCOLR registers.

17.4.4 DMA2D foreground and background CLUT interface

The CLUT interface manages the CLUT memory access and the automatic loading of the CLUT.

Three kinds of accesses are possible:

- CLUT read by the PFC during pixel format conversion operation
- CLUT accessed through the AHB slave port when the CPU is reading or writing data into the CLUT
- CLUT written through the AXI master port when an automatic loading of the CLUT is performed

The CLUT memory loading can be done in two different ways:

- Automatic loading

  The following sequence should be followed to load the CLUT:

  a) Program the CLUT address into the DMA2D_FGCMAR register (foreground CLUT) or DMA2D_BGCMAR register (background CLUT)

  b) Program the CLUT size in the CS[7:0] field of the DMA2D_FGPFCCR register (foreground CLUT) or DMA2D_BGPFCCR register (background CLUT).

  c) Set the START bit of the DMA2D_FGPFCCR register (foreground CLUT) or DMA2D_BGPFCCR register (background CLUT) to start the transfer. During this automatic loading process, the CLUT is not accessible by the CPU. If a conflict occurs, a CLUT access error interrupt is raised assuming CAEIE is set to '1' in DMA2D_CR.

- Manual loading

  The application has to program the CLUT manually through the DMA2D AHB slave port to which the local CLUT memory is mapped.The foreground CLUT is located at address offset 0x0400 and the background CLUT at address offset 0x0800.

The CLUT format can be 24 or 32 bits. It is configured through the CCM bit of the DMA2D_FGPFCCR register (foreground CLUT) or DMA2D_BGPFCCR register (background CLUT) as shown in *Table 129: Supported CLUT color mode*.

Table 129. Supported CLUT color mode

CCM	CLUT color mode
0	32-bit ARGB8888
1	24-bit RGB888

The way the CLUT data are organized in the system memory is specified in *Table 130: CLUT data order in memory*.

Table 130. CLUT data order in memory

CLUT Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]

17.4.5 DMA2D blender

The DMA2D blender blends the source pixels by pair to compute the resulting pixel.

The blending is performed according to the following equation:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Division is rounded to the nearest lower integer

No configuration register is required by the blender. The blender usage depends on the DMA2D operating mode defined in MODE[1:0] field of the DMA2D_CR register.

17.4.6 DMA2D output PFC

The output PFC performs the pixel format conversion from 32 bits to the output format defined in the CM[2:0] field of the DMA2D output pixel format converter configuration register (DMA2D_OPFCCR).

The supported output formats are given in *Table 131: Supported color mode in output*

Table 131. Supported color mode in output

CM[2:0]	Color mode
000	ARGB8888
001	RGB888
010	RGB565
011	ARGB1555
100	ARGB4444

Note: To support the alternate format, the calculated alpha value can be inverted setting the AI bit of the DMA2D_OPFCCR registers. This applies also to the Alpha value used in the DMA2D_OCOLR.

The R and B fields can also be swapped setting the RBS bit of the DMA2D_OPFCCR registers. This applies also to the RGB order used in the DMA2D_OCOLR.

17.4.7 DMA2D output FIFO

The output FIFO programs the pixels according to the color format defined in the output PFC.

The destination area is defined through a set of control registers:

- DMA2D output memory address register (DMA2D_OMAR)
- DMA2D output offset register (DMA2D_OOR**)**
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR**)**

If the DMA2D operates in register-to-memory mode, the configured output rectangle is filled by the color specified in the DMA2D output color register (DMA2D_OCOLR) which contains a fixed 32-bit, 24-bit or 16-bit value. The format is selected by the CM[2:0] field of the DMA2D_OPFCCR register.

The data are stored into the memory in the order defined in *Table 132: Data order in memory*

Table 132. Data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]

The RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

17.4.8 DMA2D output FIFO byte reordering

The output FIFO bytes can be reordered to support display frame buffer update through a parallel interface (F(S)MC) directly from the DMA2D.

The reordering of bytes can be done using:

- RBS bit to swap red and blue component
- SB bit to swap byte two by two in the output FIFO

When the byte swapping is activated (SB field of the DMA2D_OPFCR is set), the number of pixel per line (PL field of the DMA2D_NLR) must be even and the output memory address (MA field of the DMA2D_OMAR) must be even, and the output line offset computed in bytes (resulting from LOM field of DMA2D_CR and LO field of DMA2D_OOR values) must be even. If not a configuration error will be detected.

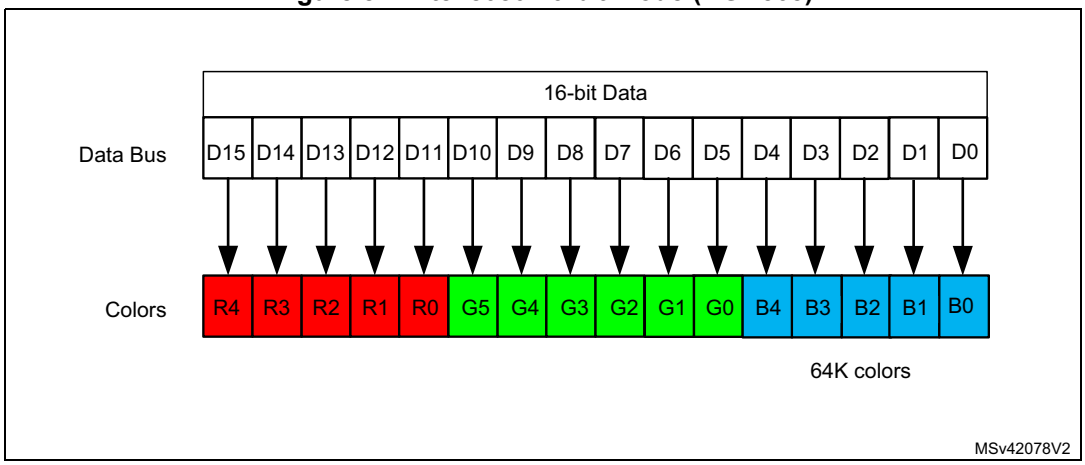
Table 133. Standard data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]

16-bit mode (RGB565)

This mode is supported without byte reordering by the DMA2D.

Figure 84. Intel 8080 16-bit mode (RGB565)



18/24-bit mode (RGB888)

This mode needs data reordering.

1. The red and the blue have to be swapped (setting the RBS bit)
2. The MSB and the LSB bytes of an half-word as to be swapped (setting the SB bit)

Figure 85. Intel 8080 18/24-bit mode (RGB888)

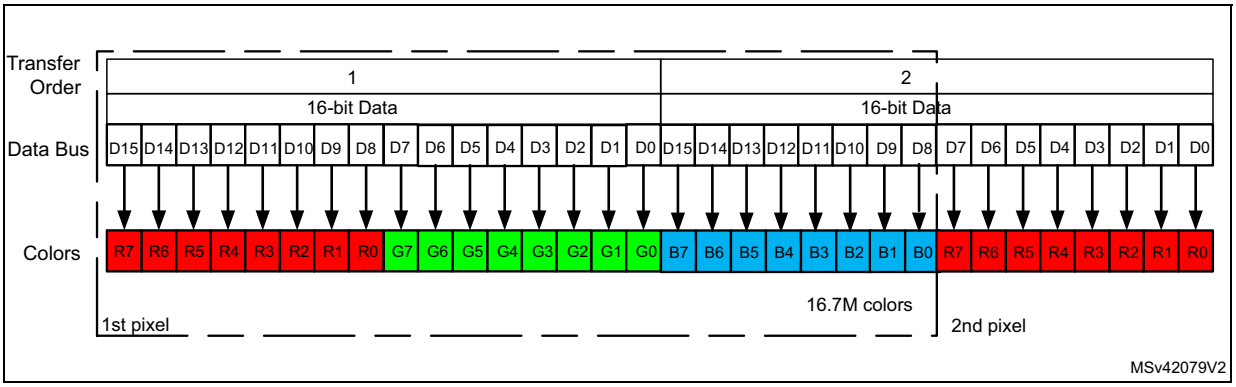


Table 134. Output FIFO byte reordering steps

Steps	@ + 3	@ + 2	@ + 1	@ + 0
Original data ordering	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
Setting the RBS bit				
Data ordering after red and blue swap (RBS set)	R_1[7:0]	B_0[7:0]	G_0[7:0]	R_0[7:0]
	G_2[7:0]	R_2[7:0]	B_1[7:0]	G_1[7:0]
	B_3[7:0]	G_3[7:0]	R_3[7:0]	B_2[7:0]

Table 134. Output FIFO byte reordering steps (continued)

Steps	@ + 3	@ + 2	@ + 1	@ + 0
Setting the SB bit				
Data ordering after byte swapping (SB set)	B_0[7:0]	R_1[7:0]	R_0[7:0]	G_0[7:0]
	R_2[7:0]	G_2[7:0]	G_1[7:0]	B_1[7:0]
	G_3[7:0]	B_3[7:0]	B_2[7:0]	R_3[7:0]

17.4.9 DMA2D AXI master port timer

An 8-bit timer is embedded into the AXI master port to provide an optional limitation of the bandwidth on the crossbar.

This timer is clocked by the AXI clock and counts a dead time between two consecutive accesses. This limits the bandwidth usage.

The timer enabling and the dead time value are configured through the AXI master port timer configuration register (DMA2D_AMPTCR).

17.4.10 DMA2D transactions

DMA2D transactions consist of a sequence of a given number of data transfers. The number of data and the width can be programmed by software.

Each DMA2D data transfer is composed of up to 4 steps:

1. Data loading from the memory location pointed by the DMA2D_FGMAR register and pixel format conversion as defined in DMA2D_FGCR.
2. Data loading from a memory location pointed by the DMA2D_BGMAR register and pixel format conversion as defined in DMA2D_BGCR.
3. Blending of all retrieved pixels according to the alpha channels resulting of the PFC operation on alpha values.
4. Pixel format conversion of the resulting pixels according to the DMA2D_OCR register and programming of the data to the memory location addressed through the DMA2D_OMAR register.

17.4.11 DMA2D configuration

Both source and destination data transfers can target peripherals and memories in the whole 4 Gbyte memory area, at addresses ranging between 0x0000 0000 and 0xFFFF FFFF.

The DMA2D can operate in any of the four following modes selected through MODE[1:0] bits of the DMA2D_CR register:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with PFC
- Memory-to-memory with PFC and blending
- Memory-to-memory with PFC, blending and fixed FG color

Register-to-memory

The register-to-memory mode is used to fill a user defined area with a predefined color.

The color format is set in the DMA2D_OPFCCR.

The DMA2D does not perform any data fetching from any source. It just writes the color defined in the DMA2D_OCOLR register to the area located at the address pointed by the DMA2D_OMAR and defined in the DMA2D_NLR and DMA2D_OOR.

Memory-to-memory

In memory-to-memory mode, the DMA2D does not perform any graphical data transformation. The foreground input FIFO acts as a buffer and the data are transferred from the source memory location defined in DMA2D_FGMAR to the destination memory location pointed by DMA2D_OMAR.

The color mode programmed in the CM[3:0] bits of the DMA2D_FGPFCCR register defines the number of bits per pixel for both input and output.

The size of the area to be transferred is defined by the DMA2D_NLR and DMA2D_FGOR registers for the source, and by DMA2D_NLR and DMA2D_OOR registers for the destination.

Memory-to-memory with PFC

In this mode, the DMA2D performs a pixel format conversion of the source data and stores them in the destination memory location.

The size of the areas to be transferred are defined by the DMA2D_NLR and DMA2D_FGOR registers for the source, and by DMA2D_NLR and DMA2D_OOR registers for the destination.

Data are fetched from the location defined in the DMA2D_FGMAR register and processed by the foreground PFC. The original pixel format is configured through the DMA2D_FGPFCCR register.

If the original pixel format is direct color mode, then the color channels are all expanded to 8 bits.

If the pixel format is indirect color mode, the associated CLUT has to be loaded into the CLUT memory.

The CLUT loading can be done automatically by following the sequence below:

1. Set the CLUT address into the DMA2D_FGCMAR.
2. Set the CLUT size in the CS[7:0] bits of the DMA2D_FGPFCCR register.
3. Set the CLUT format (24 or 32 bits) in the CCM bit of the DMA2D_FGPFCCR register.
4. Start the CLUT loading by setting the START bit of the DMA2D_FGPFCCR register.

Once the CLUT loading is complete, the CTCIF flag of the DMA2D_IFR register is raised, and an interrupt is generated if the CTCIE bit is set in DMA2D_CR. The automatic CLUT loading process can not work in parallel with classical DMA2D transfers.

The CLUT can also be filled by the CPU or by any other master through the AHB port. The access to the CLUT is not possible when a DMA2D transfer is ongoing and uses the CLUT (indirect color format).

In parallel to the color conversion process, the alpha value can be added or changed depending on the value programmed in the DMA2D_FGPFCCR register. If the original

image does not have an alpha channel, a default alpha value of 0xFF is automatically added to obtain a fully opaque pixel. The alpha value can be modified according to the AM[1:0] bits of the DMA2D_FGPFCCR register:

- It can be unchanged.
- It can be replaced by the value defined in the ALPHA[7:0] value of the DMA2D_FGPFCCR register.
- It can be replaced by the original value multiplied by the ALPHA[7:0] value of the DMA2D_FGPFCCR register divided by 255.

The resulting 32-bit data are encoded by the OUT PFC into the format specified by the CM[2:0] field of the DMA2D_OPFCCR register. The output pixel format cannot be the indirect mode since no CLUT generation process is supported.

The processed data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC and blending

In this mode, 2 sources are fetched in the foreground FIFO and background FIFO from the memory locations defined by DMA2D_FGMAR and DMA2D_BGMAR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory.

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended according to the equation below:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Division are rounded to the nearest lower integer

The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC, blending and fixed color FG

In this mode, only 1 source is fetched in the background FIFO from the memory location defined by DMA2D_BGMAR.

The value of the foreground color is given by the DMA2D_FGCOLR register and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to the AM[1:0] and ALPHA[7:0] fields of the DMA2D_FGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended together, and the resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC, blending and fixed color BG

In this mode, only 1 source is fetched in the foreground FIFO from the memory location defined by DMA2D_FGMAR.

The value of the background color is given by the DMA2D_BGCOLR register and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to the AM[1:0] and ALPHA[7:0] fields of the DMA2D_BGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter are independent and have their own CLUT memory

Once each pixel has been converted into 32 bits by their respective PFCs, they are blended together, and the resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

Configuration error detection

The DMA2D checks that the configuration is correct before any transfer. The configuration error interrupt flag is set by hardware when a wrong configuration is detected when a new transfer/automatic loading starts. An interrupt is then generated if the CEIE bit of the DMA2D_CR is set.

The wrong configurations that can be detected are listed below:

- Foreground CLUT automatic loading: MA bits of DMA2D_FGCMAR are not aligned with CCM of DMA2D_FGPFCCR.
- Background CLUT automatic loading: MA bits of DMA2D_BGCMAR are not aligned with CCM of DMA2D_BGPFCCR
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): MA bits of DMA2D_FGMAR are not aligned with CM of DMA2D_FGPFCCR
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): CM bits of DMA2D_FGPFCCR are invalid
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): PL bits of DMA2D_NLR are odd while CM of DMA2D_FGPFCCR is A4 or L4
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): LO bits of DMA2D_FGOR are odd while CM of DMA2D_FGPFCCR is A4 or L4 and LOM bit of the DMA2D_CR is pixel mode

- Memory transfer (only in blending mode and except in memory-to-memory mode with blending and fixed color FG): MA bits of DMA2D_BGMAR are not aligned with the CM of DMA2D_BGPFCCR
- Memory transfer: (only in blending mode and in blending with fixed color FG mode) CM bits of DMA2D_BGPFCCR are invalid
- Memory transfer (only in blending mode and in blending with fixed color FG mode): PL bits of DMA2D_NLR odd while CM of DMA2D_BGPFCCR is A4 or L4
- Memory transfer (only in blending mode and in blending with fixed color FG mode): LO bits of DMA2D_BGOR are odd while CM of DMA2D_BGPFCCR is A4 or L4 and LOM bit of the DMA2D_CR is pixel mode
- Memory transfer (except in memory to memory mode): MA bits of DMA2D_OMAR are not aligned with CM bits of DMA2D_OPFCCR.
- Memory transfer (except in memory to memory mode): CM bits of DMA2D_OPFCCR are invalid
- Memory transfer with byte swapping: PL bits of DMA2D_NLR are odd or MA bits of the DMA2D_OMAR are odd or LO in bytes (resulting from LOM bit of the DMA2D_CR and LO bits of DMA2D_OOR values) are odd while SB bit of DMA2D_OPFCCR is set
- Memory transfer: NL bits of DMA2D_NLR = 0
- Memory transfer: PL bits of DMA2D_NLR = 0
- Memory transfer: MODE bits of DMA2D_CR are invalid
- YCbCr format: when a CLUT loading starts setting the START bit of the DMA2D_FGPFCCR.
- YCbCr format: when the memory-to-memory mode is selected.
- YCbCr format: when YCbCR4:4:4 is selected and the sum of the number of pixel (PL) and the line offset LO is not a multiple of 8 pixels.
- YCbCr format: when YCbCr4:2:2 or YCbCr4:2:0 is selected and the sum of the number of pixel (PL) and the line offset LO is not a multiple of 16 pixels.

17.4.12 YCbCr support

The DMA2D foreground plane can support 8x8 block based YCbCr as output by the JPEG decoder with different chroma sub-sampling factors:

The memory organization follows the standard JFIF rules:

- Each of the three color component must be coded on 8-bit
- Each component must be arranged by blocks of 8x8 (64 bytes) called MCU

Depending of the chroma sub-sampling factor the MCU must be arranged in the memory as described in *Table 135: MCU order in memory*.

Table 135. MCU order in memory

Sub-sampling	@	@ + 64	@ + 128	@+192	@+256	@ + 320
4:4:4	Y_1	Cb_1	Cr_1	Y_2	Cb_2	Cr_2
4:2:2	Y_1	Y_2	Cb_{12}	Cr_{12}	Y_3	Y_4
4:2:0	Y_1	Y_2	Y_3	Y_4	Cb_{1234}	Cr_{1234}

The chroma sub-sampling factor is configured through the CSS field of the DMA2D_FGPFCCR register.

Once the DMA2D has started with the foreground configured in YCbCr color mode, the first 2 chroma MCU are loaded in the foreground CLUT. Once the chroma MCU are loaded, the DMA2D performs the loading of the Y MCU as for a classical color mode.

17.4.13 DMA2D transfer control (start, suspend, abort and completion)

Once the DMA2D is configured, the transfer can be launched by setting the START bit of the DMA2D_CR register. Once the transfer is completed, the START bit is automatically reset and the TCIF flag of the DMA2D_ISR register is raised. An interrupt can be generated if the TCIE bit of the DMA2D_CR is set.

The user application can suspend the DMA2D at any time by setting the SUSP bit of the DMA2D_CR register. The transaction can then be aborted by setting the ABORT bit of the DMA2D_CR register or can be restarted by resetting the SUSP bit of the DMA2D_CR register.

The user application can abort at any time an ongoing transaction by setting the ABORT bit of the DMA2D_CR register. In this case, the TCIF flag is not raised.

Automatic CLUT transfers can also be aborted or suspended by using their own START bits in the DMA2D_FGPFCCR and DMA2D_BGPFCCR registers.

17.4.14 Watermark

A watermark can be programmed to generate an interrupt when the last pixel of a given line has been written to the destination memory area.

The line number is defined in the LW[15:0] field of the DMA2D_LWR register.

When the last pixel of this line has been transferred, the TWIF flag of the DMA2D_ISR register is raised and an interrupt is generated if the TWIE bit of the DMA2D_CR is set.

17.4.15 Error management

Two kind of errors can be triggered:

- AXI master port errors signaled by the TEIF flag of the DMA2D_ISR register.
- Conflicts caused by CLUT access (CPU trying to access the CLUT while a CLUT loading or a DMA2D transfer is ongoing) signaled by the CAEIF flag of the DMA2D_ISR register.

Both flags are associated to their own interrupt enable flag in the DMA2D_CR register to generate an interrupt if need be (TEIE and CAEIE).

17.4.16 AXI dead time

To limit the AXI bandwidth usage, a dead time between two consecutive AXI accesses can be programmed.

This feature can be enabled by setting the EN bit in the DMA2D_AMTCR register.

The dead time value is stored in the DT[7:0] field of the DMA2D_AMTCR register. This value represents the guaranteed minimum number of cycles between two consecutive transactions on the AXI bus.

The update of the dead time value while the DMA2D is running will be taken into account for the next AXI transfer.

17.5 DMA2D interrupts

An interrupt can be generated on the following events:

- Configuration error
- CLUT transfer complete
- CLUT access error
- Transfer watermark reached
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Table 136. DMA2D interrupt requests

Interrupt event	Event flag	Enable control bit
Configuration error	CEIF	CEIE
CLUT transfer complete	CTCIF	CTCIE
CLUT access error	CAEIF	CAEIE
Transfer watermark	TWF	TWIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

17.6 DMA2D registers

17.6.1 DMA2D control register (DMA2D_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[2:0]		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	LOM	Res.	Res.	Res.	ABORT	SUSP	START
		rw	rw	rw	rw	rw	rw		rw				rs	rw	rs

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE[2:0]**: DMA2D mode

This bit is set and cleared by software. It cannot be modified while a transfer is ongoing.
000: Memory-to-memory (FG fetch only)
001: Memory-to-memory with PFC (FG fetch only with FG PFC active)
010: Memory-to-memory with blending (FG and BG fetch with PFC and blending)
011: Register-to-memory (no FG nor BG, only output stage active)
100: Memory-to-memory with Blending and fixed color FG (BG fetch only with FG and BG PFC active)
101: Memory-to-memory with Blending and fixed color BG (BG fetch only with FG and BG PFC active)
others: meaningless

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **CEIE**: Configuration Error Interrupt Enable

This bit is set and cleared by software.
0: CE interrupt disable
1: CE interrupt enable

Bit 12 **CTCIE**: CLUT transfer complete interrupt enable

This bit is set and cleared by software.
0: CTC interrupt disable
1: CTC interrupt enable

Bit 11 **CAEIE**: CLUT access error interrupt enable

This bit is set and cleared by software.
0: CAE interrupt disable
1: CAE interrupt enable

Bit 10 **TWIE**: Transfer watermark interrupt enable

This bit is set and cleared by software.
0: TW interrupt disable
1: TW interrupt enable

Bit 9 **TCIE**: Transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disable
1: TC interrupt enable

Bit 8 **TEIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disable
1: TE interrupt enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LOM**: Line Offset Mode
This bit configures how is expressed the line offset (pixels or bytes) for the foreground, background and output.
This bit is set and cleared by software. It can not be modified while a transfer is on going.
0: Line offsets are expressed in pixels
1: Line offsets are expressed in bytes

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **ABORT**: Abort
This bit can be used to abort the current transfer. This bit is set by software and is automatically reset by hardware when the START bit is reset.
0: No transfer abort requested
1: Transfer abort requested

Bit 1 **SUSP**: Suspend
This bit can be used to suspend the current transfer. This bit is set and reset by software. It is automatically reset by hardware when the START bit is reset.
0: Transfer not suspended
1: Transfer suspended

Bit 0 **START**: Start
This bit can be used to launch the DMA2D according to the parameters loaded in the various configuration registers. This bit is automatically reset by the following events:
- At the end of the transfer
- When the data transfer is aborted by the user application by setting the ABORT bit in DMA2D_CR
- When a data transfer error occurs
- When the data transfer has not started due to a configuration error or another transfer operation already ongoing (automatic CLUT loading).

17.6.2 DMA2D Interrupt Status Register (DMA2D_ISR)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CEIF**: Configuration error interrupt flag

This bit is set when the START bit of DMA2D_CR, DMA2DFGPFCCR or DMA2D_BGPFCCR is set and a wrong configuration has been programmed.

Bit 4 **CTCIF**: CLUT transfer complete interrupt flag

This bit is set when the CLUT copy from a system memory area to the internal DMA2D memory is complete.

Bit 3 **CAEIF**: CLUT access error interrupt flag

This bit is set when the CPU accesses the CLUT while the CLUT is being automatically copied from a system memory to the internal DMA2D.

Bit 2 **TWIF**: Transfer watermark interrupt flag

This bit is set when the last pixel of the watermarked line has been transferred.

Bit 1 **TCIF**: Transfer complete interrupt flag

This bit is set when a DMA2D transfer operation is complete (data transfer only).

Bit 0 **TEIF**: Transfer error interrupt flag

This bit is set when an error occurs during a DMA transfer (data transfer or automatic CLUT loading).

17.6.3 DMA2D interrupt flag clear register (DMA2D_IFCR)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CCEIF**: Clear configuration error interrupt flag
Programming this bit to 1 clears the CEIF flag in the DMA2D_ISR register

Bit 4 **CCTCIF**: Clear CLUT transfer complete interrupt flag
Programming this bit to 1 clears the CTCIF flag in the DMA2D_ISR register

Bit 3 **CAECIF**: Clear CLUT access error interrupt flag
Programming this bit to 1 clears the CAEIF flag in the DMA2D_ISR register

Bit 2 **CTWIF**: Clear transfer watermark interrupt flag
Programming this bit to 1 clears the TWIF flag in the DMA2D_ISR register

Bit 1 **CTCIF**: Clear transfer complete interrupt flag
Programming this bit to 1 clears the TCIF flag in the DMA2D_ISR register

Bit 0 **CTEIF**: Clear Transfer error interrupt flag
Programming this bit to 1 clears the TEIF flag in the DMA2D_ISR register

17.6.4 DMA2D foreground memory address register (DMA2D_FGMAR)

Address offset: 0x000C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MA[31:0]**: Memory address

Address of the data used for the foreground image. This register can only be written when data transfers are disabled. Once the data transfer has started, this register is read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

17.6.5 DMA2D foreground offset register (DMA2D_FGOR)

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LO[15:0]**: Line offset

The line offset used for the foreground image, expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] bits are considered, LO[15:14] bits are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once the data transfer has started, they become read-only.

If the image format is 4-bit per pixel, the line offset must be even.

17.6.6 DMA2D background memory address register (DMA2D_BGMAR)

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31: 0 **MA[31:0]**: Memory address

Address of the data used for the background image. This register can only be written when data transfers are disabled. Once a data transfer has started, this register is read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

17.6.7 DMA2D background offset register (DMA2D_BGOR)

Address offset: 0x0018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LO[15:0]**: Line offset

The line offset used for the background image, expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] bits are considered, LO[15:14] bits are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once the data transfer has started, they become read-only.

If the image format is 4-bit per pixel, the line offset must be even.

17.6.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR)

Address offset: 0x001C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	CSS[1:0]		AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rc_w1	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha value

These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied by the original alpha value according to the alpha mode selected through the AM[1:0] bits.
These bits can only be written when data transfers are disabled. Once a transfer has started, they become read-only.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red Blue Swap

This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.
0: Regular alpha
1: Inverted alpha

Bits 19:18 **CSS[1:0]**: Chroma Sub-Sampling

These bits define the chroma sub-sampling mode for YCbCr color mode. Once the transfer has started, these bits are read-only.
00: 4:4:4 (no chroma sub-sampling)
01: 4:2:2
10: 4:2:0
others: meaningless

Bits 17:16 **AM[1:0]**: Alpha mode

These bits select the alpha channel value to be used for the foreground image. They can only be written data the transfer are disabled. Once the transfer has started, they become read-only.
00: No modification of the foreground image alpha channel value
01: Replace original foreground image alpha channel value by ALPHA[7: 0]
10: Replace original foreground image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value
other configurations are meaningless

Bits 15:8 **CS[7:0]**: CLUT size

These bits define the size of the CLUT used for the foreground image. Once the CLUT transfer has started, this field is read-only.
The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **START**: Start

This bit can be set to start the automatic loading of the CLUT. It is automatically reset:

- at the end of the transfer
- when the transfer is aborted by the user application by setting the ABORT bit in DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already ongoing (data transfer or automatic background CLUT transfer).

Bit 4 **CCM**: CLUT color mode

This bit defines the color format of the CLUT. It can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.
0: ARGB8888
1: RGB888

Bits 3:0 **CM[3:0]**: Color mode

These bits defines the color format of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
0000: ARGB8888
0001: RGB888
0010: RGB565
0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
1011: YCbCr
others: meaningless

17.6.9 DMA2D foreground color register (DMA2D_FGCOLR)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **RED[7:0]**: Red Value

These bits defines the red value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 15:8 **GREEN[7:0]**: Green Value

These bits defines the green value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, They are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 7:0 **BLUE[7:0]**: Blue Value

These bits defines the blue value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, They are read-only.

Used also for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

17.6.10 DMA2D background PFC control register (DMA2D_BGPFCCR)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rc_w1	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha value

These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied with the original alpha value according to the alpha mode selected with bits AM[1: 0]. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red Blue Swap

This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.
0: Regular alpha
1: Inverted alpha

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **AM[1:0]**: Alpha mode

These bits define which alpha channel value to be used for the background image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
00: No modification of the foreground image alpha channel value
01: Replace original background image alpha channel value by ALPHA[7: 0]
10: Replace original background image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value
others: meaningless

Bits 15:8 **CS[7:0]**: CLUT size

These bits define the size of the CLUT used for the BG. Once the CLUT transfer has started, this field is read-only.
The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **START**: Start

This bit is set to start the automatic loading of the CLUT. This bit is automatically reset:

- at the end of the transfer
- when the transfer is aborted by the user application by setting the ABORT bit in the DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already on going (data transfer or automatic BackGround CLUT transfer).

Bit 4 **CCM**: CLUT Color mode

These bits define the color format of the CLUT. This register can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.
0: ARGB8888
1: RGB888

Bits 3:0 **CM[3:0]**: Color mode

These bits define the color format of the foreground image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
0000: ARGB8888
0001: RGB888
0010: RGB565
0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
others: meaningless

17.6.11 DMA2D background color register (DMA2D_BGCOLR)

Address offset: 0x0028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **RED[7:0]**: Red Value

These bits define the red value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color BG in memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active) mode.

Bits 15:8 **GREEN[7:0]**: Green Value

These bits define the green value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color BG in memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active) mode.

Bits 7:0 **BLUE[7:0]**: Blue Value

These bits define the blue value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Used also for fixed color BG in memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active) mode.

17.6.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR)

Address offset: 0x002C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31: 0 **MA[31:0]**: Memory Address

Address of the data used for the CLUT address dedicated to the foreground image. This register can only be written when no transfer is ongoing. Once the CLUT transfer has started, this register is read-only.

If the foreground CLUT format is 32-bit, the address must be 32-bit aligned.

17.6.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR)

Address offset: 0x0030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31: 0 **MA[31:0]**: Memory address

Address of the data used for the CLUT address dedicated to the background image. This register can only be written when no transfer is on going. Once the CLUT transfer has started, this register is read-only.

If the background CLUT format is 32-bit, the address must be 32-bit aligned.

17.6.14 DMA2D output PFC control register (DMA2D_OPFCCR)

Address offset: 0x0034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBS	AI	Res.	Res.	Res.	Res.
										rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SB	Res.	Res.	Res.	Res.	Res.	CM[2:0]		
							rw						rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red Blue Swap
This bit allows to swap the R & B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted
This bit inverts the alpha value. Once the transfer has started, this bit is read-only.
0: Regular alpha
1: Inverted alpha

Bits 19:9 Reserved, must be kept at reset value.

Bit 8 **SB**: Swap Bytes
When set, the bytes in the output FIFO are swapped two by two.
When this bit is set, the number of pixel per line (PL) must be even, and the output memory address (OMAR) must be even.
This register can only be written when the transfer is disabled. Once the transfer has started, this register is read-only.
0: Bytes in regular order in the output FIFO
1: Bytes are swapped two by two in the output FIFO

Bits 7:3 Reserved, must be kept at reset value.

Bits 2: 0 **CM[2:0]**: Color mode
These bits define the color format of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
others: meaningless

17.6.15 DMA2D output color register (DMA2D_OCOLR)

Address offset: 0x0038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								RED[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
RED[4:0]					GREEN[5:0]					BLUE[4:0]					
A	RED[4:0]				GREEN[4:0]					BLUE[4:0]					
ALPHA[3:0]				RED[3:0]				GREEN[3:0]				BLUE[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha Channel Value

These bits define the alpha channel of the output color. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 23:16 **RED[7:0]**: Red Value

These bits define the red value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 15:8 **GREEN[7:0]**: Green Value

These bits define the green value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 7:0 **BLUE[7:0]**: Blue Value

These bits define the blue value of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

17.6.16 DMA2D output memory address register (DMA2D_OMAR)

Address offset: 0x003C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31: 0 **MA[31:0]**: Memory Address

Address of the data used for the output FIFO. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned and a 16-bit per pixel format must be 16-bit aligned.

17.6.17 DMA2D output offset register (DMA2D_OOR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LO[15:0]**: Line offset

The line offset used for the output expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once data transfer has started, they become read-only.

17.6.18 DMA2D number of line register (DMA2D_NLR)

Address offset: 0x0044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PL[13:0]**: Pixel per lines

Number of pixels per lines of the area to be transferred. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
If any of the input image format is 4-bit per pixel, pixel per lines must be even.

Bits 15:0 **NL[15:0]**: Number of lines

Number of lines of the area to be transferred. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

17.6.19 DMA2D line watermark register (DMA2D_LWR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LW[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LW[15:0]**: Line watermark

These bits allow to configure the line watermark for interrupt generation.
An interrupt is raised when the last pixel of the watermarked line has been transferred.
These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

17.6.20 DMA2D AXI master timer configuration register (DMA2D_AMTCR)

Address offset: 0x004C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
rw	rw	rw	rw	rw	rw	rw	rw								rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DT[7:0]**: Dead Time

Dead time value in the AXI clock cycle inserted between two consecutive accesses on the AXI master port. These bits represent the minimum guaranteed number of cycles between two consecutive AXI accesses.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Enable

Enables the dead time functionality.

17.6.21 DMA2D register map

The following table summarizes the DMA2D registers. Refer to *Section 2.3.2 on page 124* for the DMA2D register base address.

Table 137. DMA2D register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	DMA2D_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[2:0]			Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	LOM	Res.	Res.	Res.	ABORT	SUSP	START
	Reset value														0	0	0			0	0	0	0	0	0		0				0	0	0
0x0004	DMA2D_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
	Reset value																											0	0	0	0	0	0
0x0008	DMA2D_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
	Reset value																											0	0	0	0	0	0
0x000C	DMA2D_FGMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0010	DMA2D_FGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	DMA2D_BGMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	DMA2D_BGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	DMA2D_FGPFCCR	ALPHA[7:0]								Res.	Res.	RBS	AI	CSS[1:0]		AM[1:0]		CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
	Reset value	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x0020	DMA2D_FGCOLR	APLHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	DMA2D_BGPFCCR	ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]		CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
	Reset value	0	0	0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x0028	DMA2D_BGCOLR	APLHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	DMA2D_FGCMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	DMA2D_BGCMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	DMA2D_OPFCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBS	AI	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SB	Res.	Res.	Res.	Res.	Res.	CM[2:0]		
	Reset value											0	0												0						0	0	0

Table 137. DMA2D register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0038	DMA2D_OCOLR	APLHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
		Res.																RED[4:0]					GREEN[6:0]				BLUE[4:0]						
		Res.																A	RED[4:0]				GREEN[4:0]				BLUE[4:0]						
		Res.																ALPHA[3:0]				RED[3:0]				GREEN[3:0]				BLUE[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	DMA2D_OMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	DMA2D_OOR	Res.																LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	DMA2D_NLR	Res.	Res.	PL[13:0]														NL[15:0]															
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	DMA2D_LWR	Res.																LW[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	DMA2D_AMTCR	Res.																DT[7:0]								Res.							EN
	Reset value																	0	0	0	0	0	0	0	0								0
0x0050-Ox03F		Res.																															
0x0400-0x07FF	DMA2D_FGCLUT	APLHA[7:0][255:0]								RED[7:0][255:0]								GREEN[7:0][255:0]								BLUE[7:0][255:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0800-0x0BFF	DMA2D_BGCLUT	APLHA[7:0][255:0]								RED[7:0][255:0]								GREEN[7:0][255:0]								BLUE[7:0][255:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

18 Nested Vectored Interrupt Controllers

18.1 NVIC features

The NVIC includes the following features:

- up to 150 maskable interrupt channels for STM32H7xxx (not including the 16 interrupt lines of Cortex®-M7 with FPU)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- low-latency exception and interrupt handling
- power management control
- implementation of system control registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts, including the core exceptions, are managed by the NVIC.

For more information on exceptions and NVIC programming, refer to PM0253 programming manual for Cortex®-M7.

18.1.1 SysTick calibration value register

The SysTick calibration value (SYST_CALIB) is fixed to 0x3E8. It provides a reference timebase of 1 ms based when the SysTick clock frequency is 1 MHz. To match the 1 ms timebase whatever the application frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- The SysTick clock source is the 100 MHz CPU clock (HCLK):

$$\text{reload value} = (F_{HCLK} \times \text{SYST_CALIB}) - 1$$

- or the SysTick clock source is an external clock:

$$\text{reload value} = ((F_{HCLK} / 8) \times \text{SYST_CALIB}) - 1$$

  where F_{HCLK} refers to the AHB frequency expressed in MHz.

For example, to achieve a timebase of 1 ms when the SysTick clock source is the 100 MHz HCLK:

$$\text{reload value} = (100 \times \text{SYST_CALIB}) - 1 = 0\text{x}1869\text{F}$$

18.1.2 Interrupt and exception vectors

The exception vectors connected to the NVIC are the following: reset, NMI, HardFault, MemManage, Bus Fault, UsageFault, SVCall, DebugMonitor, PendSV, SysTick.

Table 138. NVIC(1)

Signal	Priority	NVIC position	Acronym	Description	Address offset
-	-	-	-	Reserved	0x0000 0000
-	-3	-	Reset	Reset	0x0000 0004
-	-2	-	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	-	HardFault	All classes of fault	0x0000 000C
-	0	-	MemManage	Memory management	0x0000 0010
-	1	-	BusFault	Prefetch fault, memory access fault	0x0000 0014
-	2	-	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C-0x0000 002B
-	3	-	SVCall	System service call via SWI instruction	0x0000 002C
-	4	-	DebugMonitor	Debug monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	-	PendSV	Pendable request for system service	0x0000 0038
-	6	-	SysTick	System tick timer	0x0000 003C
wwdg1_it	7	0	WWDG1	Window Watchdog interrupt	0x0000 0040
exti_pwr_pvd_wkup	8	1	PVD_PVM	PVD through EXTI line detection interrupt	0x0000 0044
exti_tamp_rtc_wkup	9	2	RTC_TAMP_STAMP_CSS_LSE	RTC tamper, timestamp	0x0000 0048
lsecss_rcc_it				CSS LSE	
exti_wkup_rtc_wkup	10	3	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000 004C
flash_it	11	4	FLASH	Flash memory global interrupt	0x0000 0050
rcc_it	12	5	RCC	RCC global interrupt	0x0000 0054
exti_exti0_wkup	13	6	EXTI0	EXTI Line 0 interrupt	0x0000 0058
exti_exti1_wkup	14	7	EXTI1	EXTI Line 1 interrupt	0x0000 005C
exti_exti2_wkup	15	8	EXTI2	EXTI Line 2 interrupt	0x0000 0060

Table 138. NVIC[1] (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
exti_exti3_wkup	16	9	EXTI3	EXTI Line 3interrupt	0x0000 0064
exti_exti4_wkup	17	10	EXTI4	EXTI Line 4interrupt	0x0000 0068
dma1_it0	18	11	DMA_STR0	DMA1 Stream0 global interrupt	0x0000 006C
dma1_it1	19	12	DMA_STR1	DMA1 Stream1 global interrupt	0x0000 0070
dma1_it2	20	13	DMA_STR2	DMA1 Stream2 global interrupt	0x0000 0074
dma1_it3	21	14	DMA_STR3	DMA1 Stream3 global interrupt	0x0000 0078
dma1_it4	22	15	DMA_STR4	DMA1 Stream4 global interrupt	0x0000 007C
dma1_it5	23	16	DMA_STR5	DMA1 Stream5 global interrupt	0x0000 0080
dma1_it6	24	17	DMA_STR6	DMA1 Stream6 global interrupt	0x0000 0084
adc1_it	25	18	ADC1_2	ADC1 and ADC2 global interrupt	0x0000 0088
adc2_it					
ttfdcan_intr0_it	26	19	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 008C
fdcan_intr0_it	27	20	FDCAN2_IT0	FDCAN2 Interrupt 0	0x0000 0090
ttfdcan_intr1_it	28	21	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 0094
fdcan_intr1_it	29	22	FDCAN2_IT1	FDCAN2 Interrupt 1	0x0000 0098
exti_exti5_wkup	30	23	EXTI9_5	EXTI Line[9:5] interrupts	0x 0000 009C
exti_exti6_wkup					
exti_exti7_wkup					
exti_exti8_wkup					
exti_exti9_wkup					
tim1_brk_it	31	24	TIM1_BRK	TIM1 break interrupt	0x0000 00A0
tim1_upd_it	32	25	TIM1_UP	TIM1 update interrupt	0x0000 00A4
tim1_trg_it	33	26	TIM1_TRG_COM	TIM1 trigger and commutation interrupts	0x0000 00A8
tim1_cc_it	34	27	TIM_CC	TIM1 capture / compare interrupt	0x0000 00AC
tim2_it	35	28	TIM2	TIM2 global interrupt	0x0000 00B0
tim3_it	36	29	TIM3	TIM3 global interrupt	0x0000 00B4
tim4_it	37	30	TIM4	TIM4 global interrupt	0x0000 00B8

Table 138. NVIC(1) (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
i2c1_ev_it	38	31	I2C1_EV	I2C1 event interrupt	0x0000 00BC
exti_i2c1_ev_wkup					
i2c1_err_it	39	32	I2C1_ER	I2C1 error interrupt	0x0000 00C0
i2c2_ev_it	40	33	I2C2_EV	I2C2 event interrupt	0x0000 00C4
exti_i2c2_ev_wkup					
i2c2_err_it	41	34	I2C2_ER	I2C2 error interrupt	0x0000 00C8
spi1_it	42	35	SPI1	SPI1 global interrupt	0x0000 00CC
exti_spi1_it					
spi2_it	43	36	SPI2	SPI2 global interrupt	0x0000 00D0
exti_spi2_it					
usart1_gbl_it	44	37	USART1	USART1 global interrupt	0x0000 00D4
exti_usart1_wkup					
usart2_gbl_it	45	38	USART2	USART2 global interrupt	0x0000 00D8
exti_usart2_wkup					
usart3_gbl_it	46	39	USART3	USART3 global interrupt	0x0000 00DC
exti_usart3_wkup					
exti_exti10_it	47	40	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
exti_exti11_wkup					
exti_exti12_wkup					
exti_exti13_wkup					
exti_exti14_wkup					
exti_exti15_wkup					
exti_rtc_al	48	41	RTC_ALARM	RTC alarms (A and B) through EXTI Line interrupts	0x0000 00E4
-	49	42	-	-	0x0000 00E8
tim8_brk_it	50	43	TIM8_BRK_TIM12	TIM8 break and TIM12 global interrupts	0x0000 00EC
tim12_gbl_it					
tim8_upd_it	51	44	TIM8_UP_TIM13	TIM8 update and TIM13 global interrupts	0x0000 00F0
tim13_gbl_it					
tim8_trg_it	52	45	TIM8_TRG_COM _TIM14	TIM8 trigger /commutation and TIM14 global interrupts	0x0000 00F4
tim14_gbl_it					
tim8_cc_it	53	46	TIM8_CC	TIM8 capture / compare interrupts	0x0000 00F8

Table 138. NVIC(1) (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
dma1_it7	54	47	DMA1_STR7	DMA1 Stream7 global interrupt	0x0000 00FC
fmc_gbl_it	55	48	FMC	FMC global interrupt	0x0000 0100
sdmmc_it	56	49	SDMMC1	SDMMC global interrupt	0x0000 0104
tim5_gbl_it	57	50	TIM5	TIM5 global interrupt	0x0000 0108
spi3_it	58	51	SPI3	SPI3 global interrupt	0x0000 010C
exti_spi3_wkup					
uart4_gbl_it	59	52	UART4	UART4 global interrupt	0x0000 0110
exti_uart4_wkup					
uart5_gbl_it	60	53	UART5	UART5 global interrupt	0x0000 0114
exti_uart5_wkup					
tim6_gbl_it	61	54	TIM6_DAC	TIM6 global interrupt	0x0000 0118
dac_unr_it				DAC underrun error interrupt	
tim7_gbl_it	62	55	TIM7	TIM7 global interrupt	0x0000 011C
dma2_it0	63	56	DMA2_STR0	DMA2 Stream0 interrupt	0x0000 0120
dma2_it1	64	57	DMA2_STR1	DMA2 Stream1 interrupt	0x0000 0124
dma2_it2	65	58	DMA2_STR2	DMA2 Stream2 interrupt	0x0000 0128
dma2_it3	66	59	DMA2_STR3	DMA2 Stream3 interrupt	0x0000 012C
dma2_it4	67	60	DMA2_STR4	DMA2 Stream4 interrupt	0x0000 0130
eth_sbd_intr_it	68	61	ETH	Ethernet global interrupt	0x0000 0134
exti_eth_wkup	69	62	ETH_WKUP	Ethernet wakeup through EXTI line interrupt	0x0000 0138
fdcan_cal_it	70	63	FDCAN_CAL	FDCAN calibration interrupts	0x0000 013C
cm7_sev_it	71	64	-	Arm® Cortex®-M7 Send even interrupt	0x0000 0140
NC	72	65	-	-	0x0000 0144
NC	73	66	-	-	0x0000 0148
NC	74	67	-	-	0x0000 014C
dma2_it5	75	68	DMA2_STR5	DMA2 Stream5 interrupt	0x0000 0150
dma2_it6	76	69	DMA2_STR6	DMA2 Stream6 interrupt	0x0000 0154
dma2_it7	77	70	DMA2_STR7	DMA2 Stream7 interrupt	0x0000 0158
usart6_gbl_it	78	71	USART6	USART6 global interrupt	0x0000 015C
exti_usart6_wkup				USART6 wakeup interrupt	

Table 138. NVIC(1) (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
i2c3_ev_it exti_i2c3_ev_wkup	79	72	I2C3_EV	I2C3 event interrupt	0x0000 0160
i2c3_err_it	80	73	I2C3_ER	I2C3 error interrupt	0x0000 0164
usb1_out_it	81	74	OTG_HS_EP1_OUT	OTG_HS out global interrupt	0x0000 0168
usb1_in_it	82	75	OTG_HS_EP1_IN	OTG_HS in global interrupt	0x0000 016C
exti_usb1_wkup	83	76	OTG_HS_WKUP	OTG_HS wakeup interrupt	0x0000 0170
usb1_gbl_it	84	77	OTG_HS	OTG_HS global interrupt	0x0000 0174
dcmi_it	85	78	DCMI	DCMI global interrupt	0x0000 0178
cryp_it	86	79	CRYP	CRYP global interrupt	0x0000 017C
hash_rng_it	87	80	HASH_RNG	HASH and RNG global interrupt	0x0000 0180
cpu_fpu_it	88	81	FPU	CPU FPU	0x0000 0184
uart7_gbl_it exti_uart7_wkup	89	82	UART7	UART7 global interrupt	0x0000 0188
uart8_gbl_it exti_uart8_wkup	90	83	UART8	UART8 global interrupt	0x0000 018C
spi4_it exti_spi4_wkup	91	84	SPI4	SPI4 global interrupt	0x0000 0190
spi5_it exti_spi5_wkup	92	85	SPI5	SPI5 global interrupt	0x0000 0194
spi6_it exti_spi6_wkup	93	86	SPI6	SPI6 global interrupt	0x0000 0198
sai1_it	94	87	SAI1	SAI1 global interrupt	0x0000 019C
ltdc_it	95	88	LTDC	LCD-TFT global interrupt	0x0000 01A0
ltdc_err_it	96	89	LTDC_ER	LCD-TFT error interrupt	0x0000 01A4
dma2d_gbl_it	97	90	DMA2D	DMA2D global interrupt	0x0000 01A8
-	98	91	SAI2	SAI2 global interrupt	0x0000 01AC
quadspi_it	99	92	QUADSPI	QuadSPI global interrupt	0x0000 01B0
lptim1_it exti_lptim_wkup	100	93	LPTIM1	LPTIM1 global interrupt	0x0000 01B4
cec_it exti_cec_it	101	94	CEC	HDMI-CEC global interrupt	0x0000 01B8
i2c4_ev_it exti_i2c4_ev_it	102	95	I2C4_EV	I2C4 event interrupt	0x0000 01BC

Table 138. NVIC[1] (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
i2c4_err_it	103	96	I2C4_ER	I2C4 error interrupt	0x0000 01C0
spdifrx_it	104	97	SPDIF	SPDIFRX global interrupt	0x0000 01C4
usb2_out_it	105	98	OTG_FS_EP1_OUT	OTG_FS out global interrupt	0x0000 01C8
usb2_in_it	106	99	OTG_FS_EP1_IN	OTG_FS in global interrupt	0x0000 01CC
exti_usb2_wkup	107	100	OTG_FS_WKUP	OTG_FS wakeup	0x0000 01D0
usb2_gbl_it	108	101	OTG_FS	OTG_FS global interrupt	0x0000 01D4
dmamux1_ovr_it	109	102	DMAMUX1_OV	DMAMUX1 overrun interrupt	0x0000 01D8
hrtim1_mst_it	110	103	HRTIM1_MST	HRTIM1 master timer interrupt	0x0000 01DC
hrtim1_tima_it	111	104	HRTIM1_TIMA	HRTIM1 timer A interrupt	0x0000 01E0
hrtim1_timb_it	112	105	HRTIM_TIMB	HRTIM1 timer B interrupt	0x0000 01E4
hrtim1_timc_it	113	106	HRTIM1_TIMC	HRTIM1 timer C interrupt	0x0000 01E8
hrtim1_timd_it	114	107	HRTIM1_TIMD	HRTIM1 timer D interrupt	0x0000 01EC
hrtim1_time_it	115	108	HRTIM_TIME	HRTIM1 timer E interrupt	0x0000 01F0
hrtim1_fault_it	116	109	HRTIM1_FLT	HRTIM1 fault interrupt	0x0000 01F4
dfsdm1_it0	117	110	DFSDM1_FLT0	DFSDM1 filter 0 interrupt	0x0000 01F8
dfsdm1_it1	118	111	DFSDM1_FLT1	DFSDM1 filter 1 interrupt	0x0000 01FC
dfsdm1_it2	119	112	DFSDM1_FLT2	DFSDM1 filter 2 interrupt	0x0000 0200
dfsdm1_it3	120	113	DFSDM1_FLT3	DFSDM1 filter 3 interrupt	0x0000 0204
sai3_gbl_it_it	121	114	SAI3	SAI3 global interrupt	0x0000 0208
swpmi_gbl_it	122	115	SWPMI1	SWPMI global interrupt	0x0000 020C
exti_swpmi_wup				SWPMI wakeup	
tim15_gbl_it	123	116	TIM15	TIM15 global interrupt	0x0000 0210
tim16_gbl_it	124	117	TIM16	TIM16 global interrupt	0x0000 0214
tim17_gbl_it	125	118	TIM17	TIM17 global interrupt	0x0000 0218
exti_mdios_wkup	126	119	MDIOS_WKUP	MDIOS wakeup	0x0000 021C
mdios_it	127	120	MDIOS	MDIOS global interrupt	0x0000 0220
jpeg_it	128	121	JPEG	JPEG global interrupt	0x0000 0224
mdma_it	129	122	MDMA	MDMA	0x0000 0228
sdmmc2_it	131	124	SDMMC2	SDMMC global interrupt	0x0000 0230
hsem1_it	132	125	HSEM0	HSEM global interrupt 1	0x0000 0234
-	133	-	-	-	0x0000 0238
adc3_it	134	127	ADC3	ADC3 global interrupt	0x0000 023C
dmamux2_ovr_it	135	128	DMAMUX2_OVR	DMAMUX2 overrun interrupt	0x0000 0240

Table 138. NVIC(1) (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
bdma_ch0_it	136	129	BDMA_CH0	BDMA channel 0 interrupt	0x0000 0244
bdma_ch1_it	137	130	BDMA_CH1	BDMA channel 1 interrupt	0x0000 0248
bdma_ch2_it	138	131	BDMA_CH2	BDMA channel 2 interrupt	0x0000 024C
bdma_ch3_it	139	132	BDMA_CH3	BDMA channel 3 interrupt	0x0000 0250
bdma_ch4_it	140	133	BDMA_CH4	BDMA channel 4 interrupt	0x0000 0254
bdma_ch5_it	141	134	BDMA_CH5	BDMA channel 5 interrupt	0x0000 0258
bdma_ch6_it	142	135	BDMA_CH6	BDMA channel 6 interrupt	0x0000 025C
bdma_ch7_it	143	136	BDMA_CH7	BDMA channel 7 interrupt	0x0000 0260
comp_gbl_it	144	137	COMP	COMP1 and COMP2 global interrupt	0x0000 0264
exti_comp1_wkup					
exti_comp2_wkup					
lptim2_it	145	138	LPTIM2	LPTIM2 timer interrupt	0x0000 0268
exti_lptim2_wkup					
lptim3_it	146	139	LPTIM3	LPTIM2 timer interrupt	0x0000 026C
exti_lptim3_wkup					
lptim4_it	147	140	LPTIM4	LPTIM2 timer interrupt	0x0000 0270
exti_lptim4_wkup					
lptim5_it	148	141	LPTIM5	LPTIM2 timer interrupt	0x0000 0274
exti_lptim5_wkup					
lpuart_gbl_it	149	142	LPUART	LPUART global interrupt	0x0000 0278
exti_lpuart_rx_it					
exti_lpuart_tx_it					
exti_d1_wwdg1_wkup	150	143	WWDG1_RST	Window Watchdog interrupt	0x0000 027C
crs_it	151	144	CRS	Clock Recovery System global interrupt	0x0000 0280
ramecc1_it	152	145	RAMECC1	ECC diagnostic global interrupt for RAMECC D1	0x0000 0284
ramecc2_it			RAMECC2	ECC diagnostic global interrupt for RAMECC D2	
ramecc2_it			RAMECC3	ECC diagnostic global interrupt for RAMECC D3	
-	153	146	SAI4	SAI4 global interrupt	0x0000 0288
-	154	147	-	-	0x0000 028C
-	155	148	-	-	0x0000 0290

Table 138. NVIC(1) (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
exti_wkup1_wkup	156	149	WKUP	WKUP0 to WKUP5 pins	0x0000 0294
exti_wkup2_wkup					
exti_wkup3_wkup					
exti_wkup4_wkup					
exti_wkup5_wkup					
exti_wkup6_wkup					

1. When different signals are connected to the same NVIC interrupt line, they are OR-ed.

19 Extended interrupt and event controller (EXTI)

The Extended Interrupt and event controller (EXTI) manages wakeup through configurable and direct event inputs. It provides wakeup requests to the Power Control, and generates interrupt requests to the CPU NVIC and to the D3 domain DMAMUX2, and events to the CPU event input.

The EXTI wakeup requests allow the system to be woken up from Stop mode, and the CPU to be woken up from CStop mode.

Both the interrupt request and event request generation can also be used in Run modes.

19.1 EXTI main features

The EXTI main features are the following:

- All Event inputs allow the CPU to wakeup and to generate a CPU interrupt and/or CPU event
- Some Event inputs allow the user to wakeup the D3 domain for autonomous Run mode and generate an interrupt to the D3 domain, i.e. the DMAMUX2

The asynchronous event inputs are classified in 2 groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), they have the following features:
  - Selectable active trigger edge
  - Interrupt pending status register bit
  - Individual Interrupt and Event generation mask
  - SW trigger possibility
  - Configurable System D3 domain wakeup events have a D3 Pending mask and status register and may have a D3 interrupt signal.
- Direct events (interrupt and wakeup sources from other peripherals, requiring to be cleared in the peripheral), they feature
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status is provided by the peripheral generating the event)
  - Individual Interrupt and Event generation mask
  - No SW trigger possibility
  - Direct system D3 domain wakeup events have a D3 Pending mask and status register and may have a D3 interrupt signal

19.2 EXTI block diagram

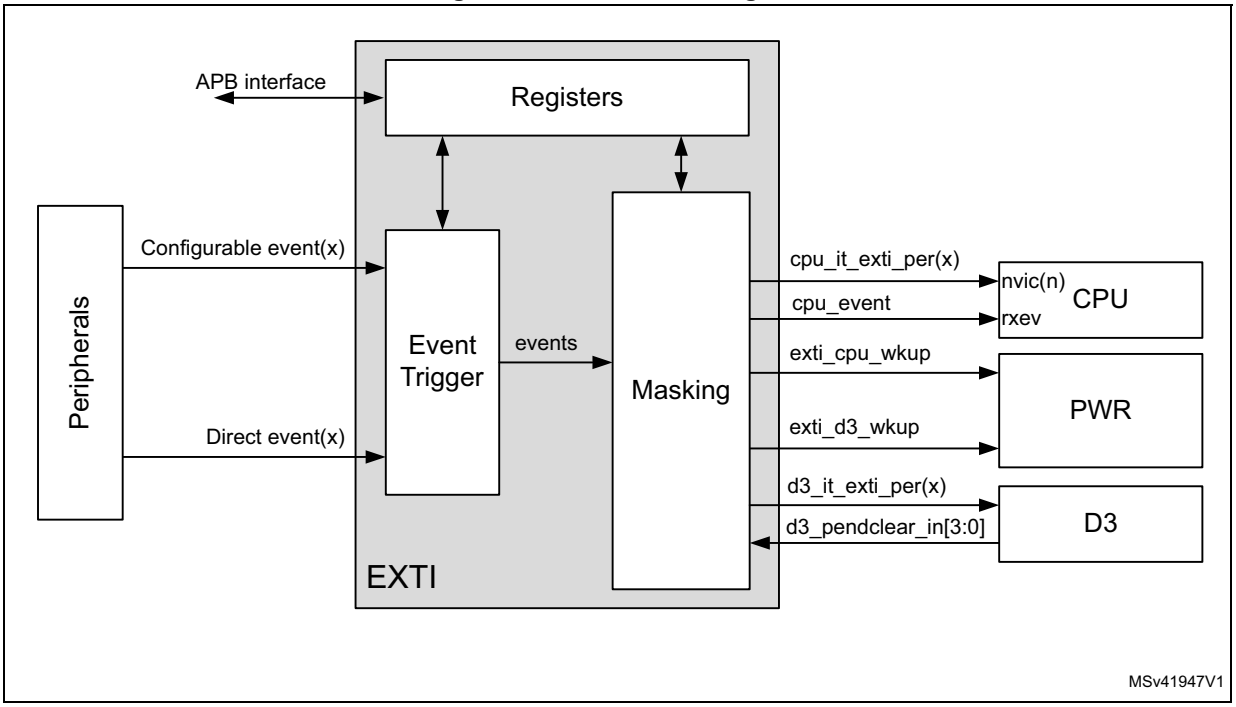
As shown in *Figure 86*, the EXTI consists of a Register block accessed via an APB interface, an Event input Trigger block, and a Masking block.

The Register block contains all EXTI registers.

The Event input trigger block provides Event input edge triggering logic.

The Masking block provides the Event input distribution to the different wakeup, interrupt and event outputs, and their masking.

Figure 86. EXTI block diagram



19.2.1 EXTI connections between peripherals, CPU, and D3 domain

The peripherals able to generate wakeup events when the system is in Stop mode or the CPU is in CStop mode are connected to an EXTI Configurable event input or Direct Event input:

- Peripheral signals that generate a pulse are connected to an EXTI Configurable Event input. For these events the EXTI provides a CPU status pending bit that has to be cleared.
- Peripheral Interrupt and Wakeup sources that have to be cleared in the peripheral are connected to an EXTI Direct Event input. There is no CPU status pending bit within the EXTI. The Interrupt or Wakeup is cleared by the CPU in the peripheral.

The Event inputs able to wakeup D3 for autonomous Run mode are provided with a D3 domain pending request function, that has to be cleared. This clearing request is taken care of by the signal selected by the Pending clear selection.

The CPU interrupts are connected to their respective CPU NVIC, and, similarly, the CPU event is connected to the CPU rxev input.

The EXTI Wakeup signals are connected to the PWR block, and are used to wakeup the D3 domain and/or the CPU.

The D3 domain interrupts allow the system to trigger events for D3 domain autonomous Run mode operation.

19.3 EXTI functional description

Depending on the EXTI Event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by *EXTI rising trigger selection register (EXTI_RTSR1)*, *EXTI rising trigger selection register (EXTI_RTSR2)*, *EXTI rising trigger selection register (EXTI_RTSR3)*, and *EXTI falling trigger selection register (EXTI_FTSR1)*, *EXTI falling trigger selection register (EXTI_FTSR2)*, *EXTI falling trigger selection register (EXTI_FTSR3)*
- Software trigger, by *EXTI software interrupt event register (EXTI_SWIER1)*, *EXTI software interrupt event register (EXTI_SWIER2)*, *EXTI software interrupt event register (EXTI_SWIER3)*
- CPU Interrupt enable, by *EXTI interrupt mask register (EXTI_CPUIMR1)*, *EXTI interrupt mask register (EXTI_CPUIMR2)*, *EXTI interrupt mask register (EXTI_CPUIMR3)*
- CPU Event enable, by *EXTI event mask register (EXTI_CPUEMR1)*, *EXTI event mask register (EXTI_CPUEMR2)*, *EXTI event mask register (EXTI_CPUEMR3)*
- D3 domain wakeup pending, by *EXTI D3 pending mask register (EXTI_D3PMR1)*, *EXTI D3 pending mask register (EXTI_D3PMR2)*, *EXTI D3 pending mask register (EXTI_D3PMR3)*

Table 139. EXTI Event input configurations and register control(1)

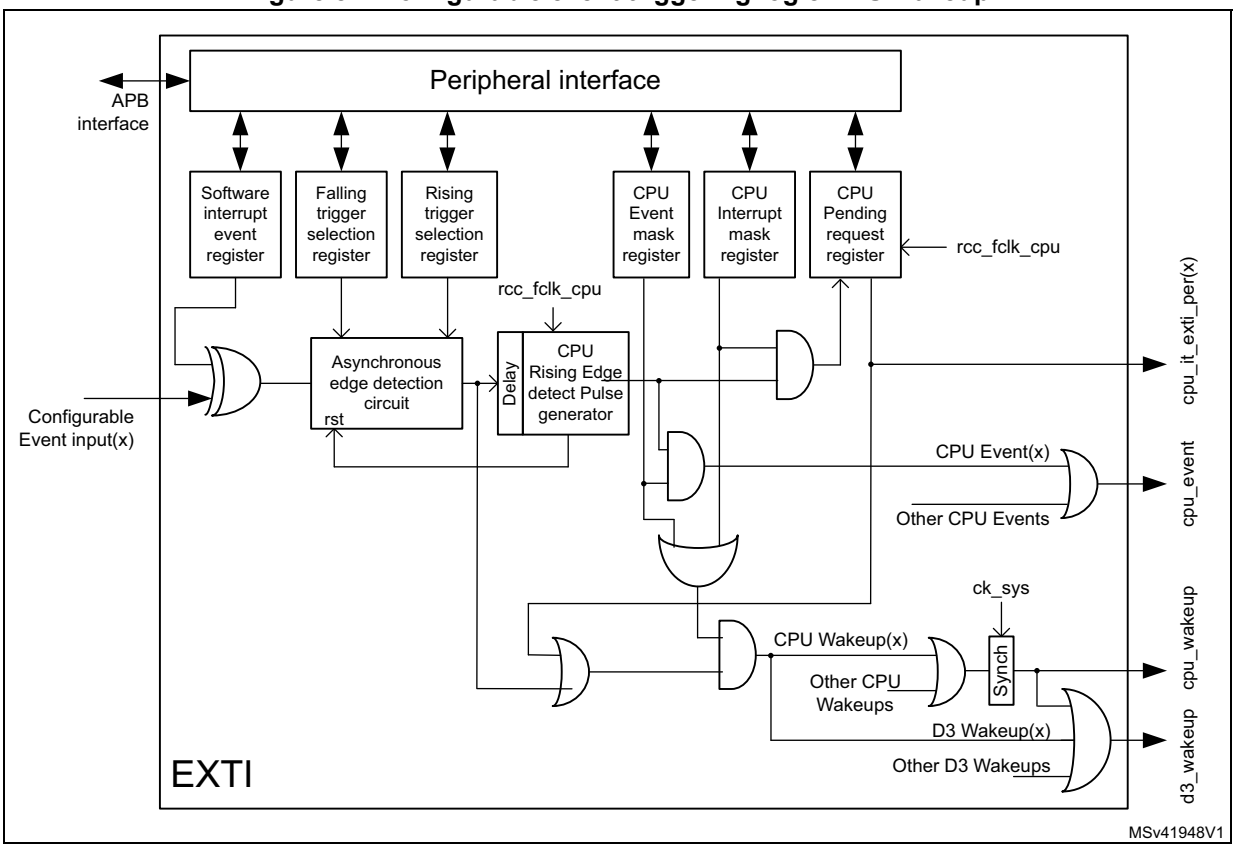
Event input type	Wakeup target(s)	Logic implementation	EXTI_RTSR	EXTI_FTSR	EXTI_SWIER	EXTI_CPUIMR	EXTI_CPUEMR	EXTI_D3PMR
Configurable	CPU	Configurable event input, CPU wakeup logic	X	X	X	X	X	-
	Any(2)	Configurable event input, Any wakeup logic						X
Direct	CPU	Direct event input, CPU wakeup logic	-	-	-	X	X	-
	Any(2)	Direct event input, Any wakeup logic						X

1. X indicates that functionality is available.
2. Waking-up D3 domain for autonomous Run mode, and/or CPU.

19.3.1 EXTI Configurable event input CPU wakeup

Figure 88 is a detailed representation of the logic associated with Configurable Event inputs which will always wake up the CPU.

Figure 87. Configurable event triggering logic CPU wakeup



The Software interrupt event register allows the system to trigger Configurable events by software, writing the *EXTI software interrupt event register (EXTI_SWIER1)*, the *EXTI software interrupt event register (EXTI_SWIER2)*, or the *EXTI software interrupt event register (EXTI_SWIER3)* register bit.

The rising edge *EXTI rising trigger selection register (EXTI_RTSR1)*, *EXTI rising trigger selection register (EXTI_RTSR2)*, *EXTI rising trigger selection register (EXTI_RTSR3)*, and falling edge *EXTI falling trigger selection register (EXTI_FTSR1)*, *EXTI falling trigger selection register (EXTI_FTSR2)*, *EXTI falling trigger selection register (EXTI_FTSR3)* selection registers allow the system to enable and select the Configurable event active trigger edge or both edges.

The devices feature dedicated interrupt mask registers, namely *EXTI interrupt mask register (EXTI_CPUIMR1)* and *EXTI interrupt mask register (EXTI_CPUIMR2)*, *EXTI interrupt mask register (EXTI_CPUIMR3)*, and *EXTI pending register (EXTI_CPUPR1)*, *EXTI pending register (EXTI_CPUPR2)*, *EXTI pending register (EXTI_CPUPR3)* for Configurable events pending request registers. The CPU pending register will only be set for an unmasked CPU interrupt. Each event provides a individual CPU interrupt to the CPU NVIC. The Configurable events interrupts need to be acknowledged by software in the EXTI_CPUPR register.

The devices feature dedicated event mask registers, i.e. *EXTI event mask register (EXTI_CPUEMR1)*, *EXTI event mask register (EXTI_CPUEMR2)*, and *EXTI event mask register (EXTI_CPUEMR3)*. The enabled event then generates an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The CPU Pending register (EXTI_CPUPR) will not be set for an unmasked CPU event.

When a CPU interrupt or CPU event is enabled, the Asynchronous edge detection circuit is reset by the clocked Delay and Rising edge detect pulse generator. This guarantees that the CPU clock is woken up before the Asynchronous edge detection circuit is reset.

Note: A detected Configurable event, enabled by the CPU, is only cleared when the CPU wakes up.

19.3.2 EXTI configurable event input Any wakeup

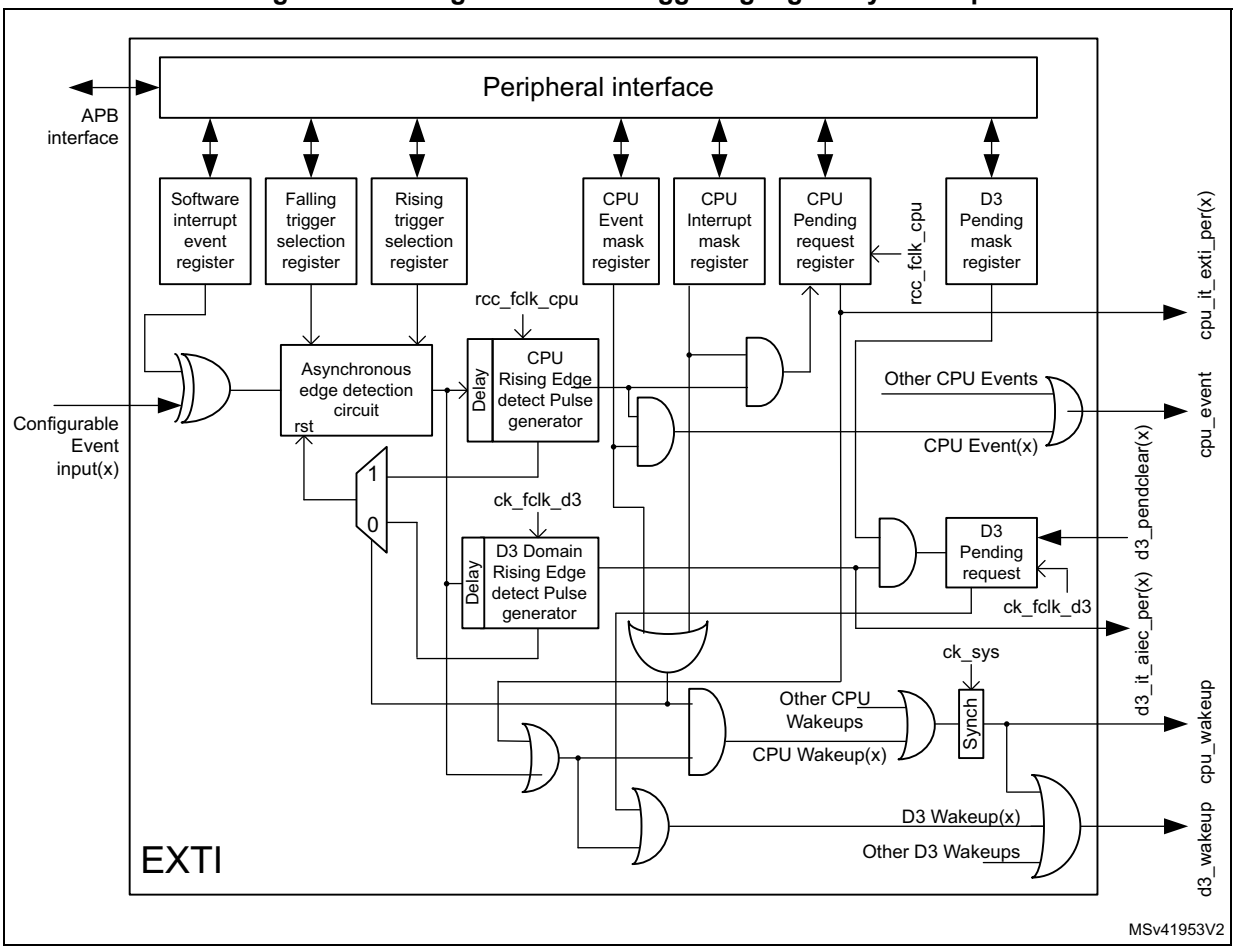
Figure 88 is a detailed representation of the logic associated with Configurable Event inputs that can wakeup D3 domain for autonomous Run mode and/or CPU (“Any” target). It provides the same functionality as the Configurable event input CPU wakeup, with additional functionality to wake up the D3 domain independently.

When all CPU interrupts and CPU events are disabled, the Asynchronous edge detection circuit is reset by the D3 domain clocked Delay and Rising edge detect pulse generator. This guarantees that the D3 domain clock is woken up before the Asynchronous edge detection circuit is reset.

Table 140. Configurable Event input Asynchronous Edge detector reset

EXTI_C1IMR	EXTI_C1EMR	Asynchronous Edge detector reset by
Both = 0		D3 domain clock rising edge detect pulse generator
At least one = 1		CPU clock rising edge detect pulse generator

Figure 88. Configurable event triggering logic Any wakeup



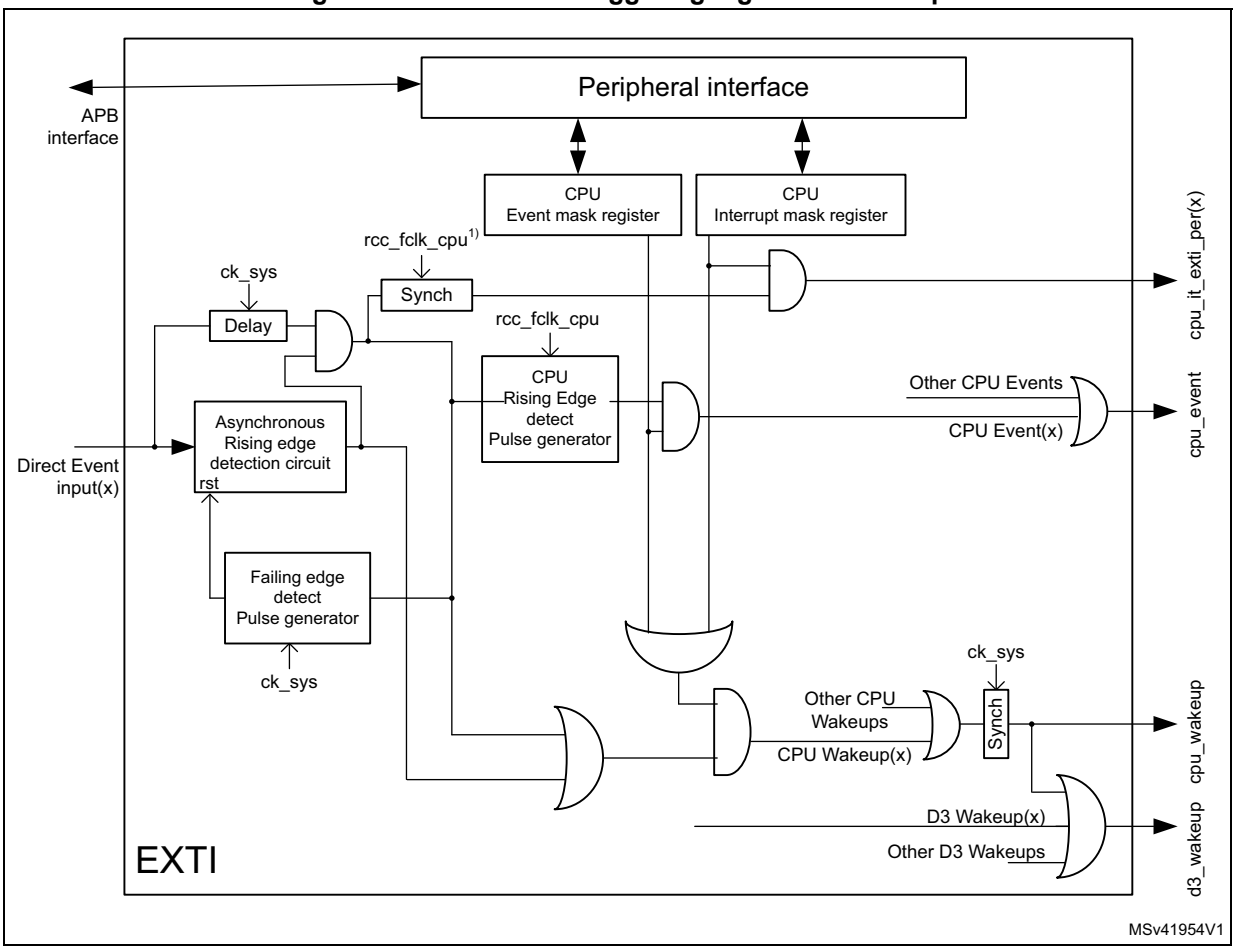
The event triggering logic for “Any” target has additional D3 Pending mask register *EXTI D3 pending mask register (EXTI_D3PMR1)*, *EXTI D3 pending mask register (EXTI_D3PMR2)*, *EXTI D3 pending mask register (EXTI_D3PMR3)*and D3 Pending request logic. The D3 Pending request logic will only be set for unmasked D3 Pending events. The D3 Pending request logic keeps the D3 domain in Run mode until the D3 Pending request logic is cleared by the selected D3 domain pendclear source.

19.3.3 EXTI direct event input CPU wakeup

Figure 89 is a detailed representation of the logic associated with Direct Event inputs waking up the CPU.

Direct events only provide CPU interrupt enable and CPU event enable functionality.

Figure 89. Direct event triggering logic CPU Wakeup

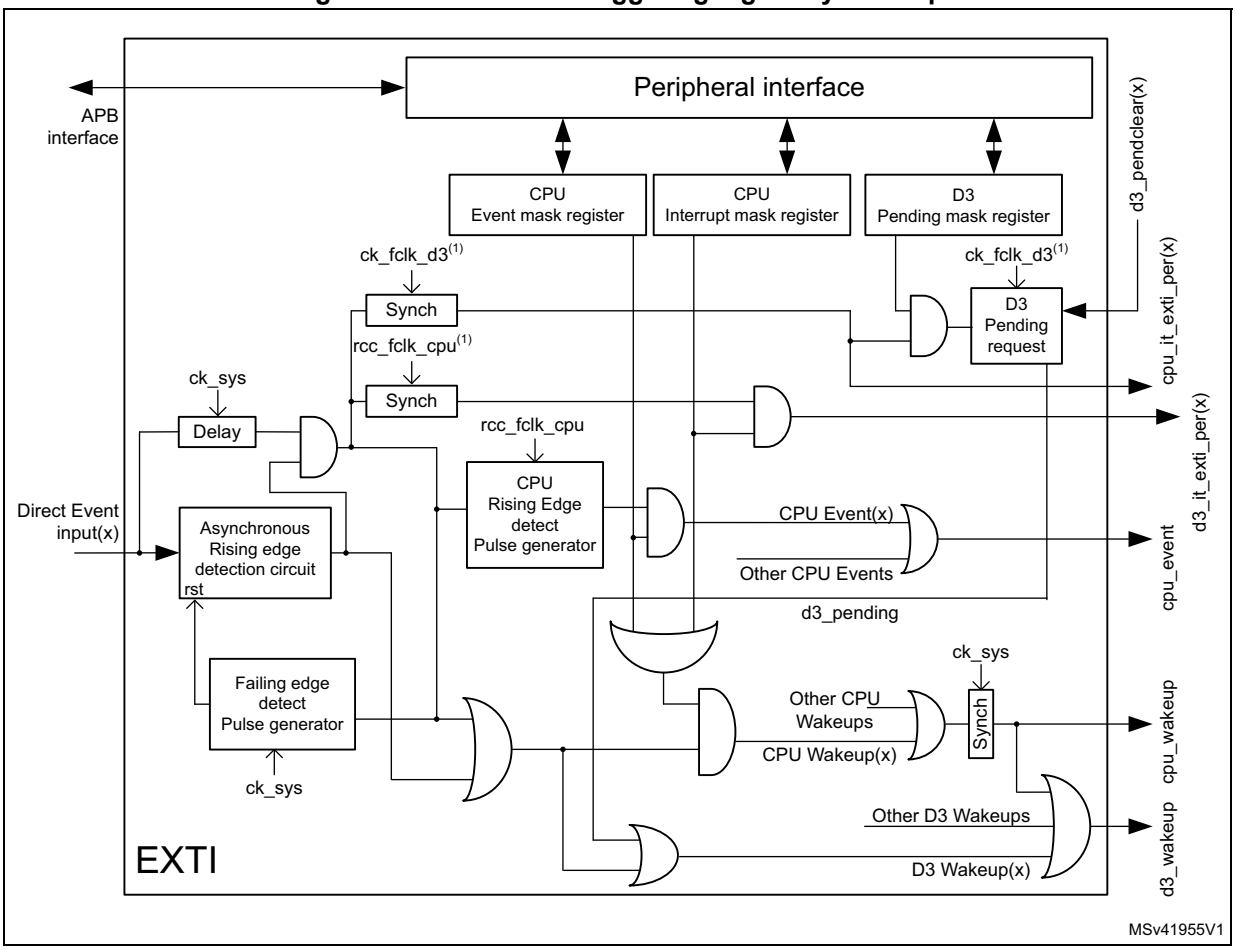


1. The CPU interrupt for asynchronous Direct Event inputs (peripheral Wakeup signals) is synchronized with the CPU clock. The synchronous Direct Event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU interrupt without resynchronization.

19.3.4 EXTI direct event input Any wakeup

Figure 90 is a detailed representation of the logic associated with Direct Event inputs waking up D3 domain for autonomous Run mode and/or CPU, (“Any” target). It provides the same functionality as the Direct event input CPU wakeup, plus additional functionality to wakeup the D3 domain independently.

Figure 90. Direct event triggering logic Any Wakeup



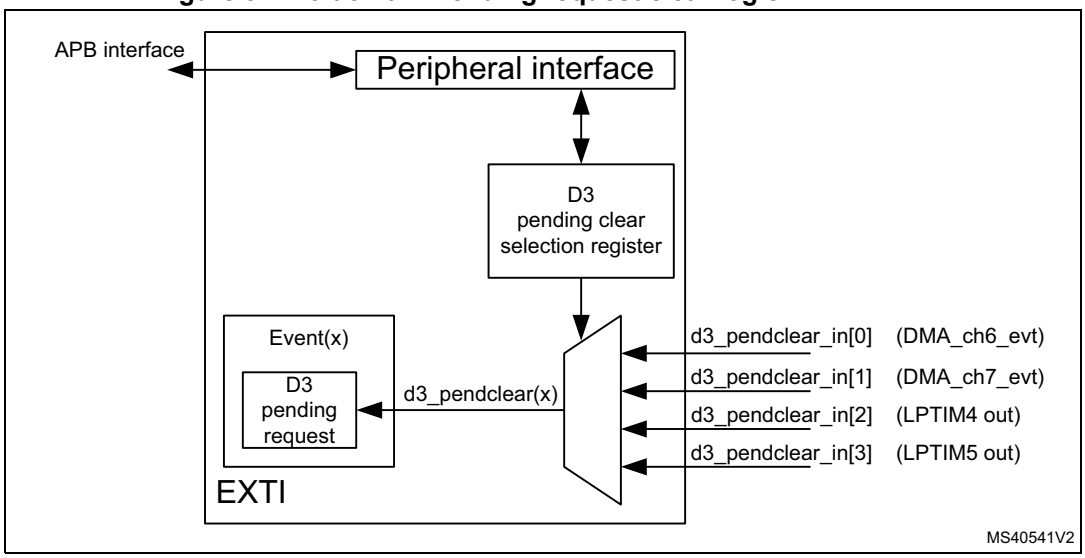
1. The CPU interrupt and D3 domain interrupt for asynchronous Direct Event inputs (peripheral Wakeup signals) are synchronized, respectively, with the CPU clock and the D3 domain clock. The synchronous Direct Event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU interrupt and the D3 domain interrupt without resynchronization in the EXTI.

19.3.5 EXTI D3 pending request clear selection

Event inputs able to wake up D3 domain for autonomous Run mode have D3 Pending request logic that can be cleared by the selected D3 pendclear source. For each D3 Pending request a D3 domain pendclear source can be selected from four different inputs.

Figure 91 is a detailed representation of the logic selecting the D3 pendclear source.

Figure 91. D3 domain Pending request clear logic



The D3 Pending request clear selection registers *EXTI D3 pending clear selection register low (EXTI_D3PCR1L)*, *EXTI D3 pending clear selection register high (EXTI_D3PCR1H)*, *EXTI D3 pending clear selection register low (EXTI_D3PCR2L)*, *EXTI D3 pending clear selection register high (EXTI_D3PCR2H)*, *EXTI D3 pending clear selection register low (EXTI_D3PCR3L)* and *EXTI D3 pending clear selection register high (EXTI_D3PCR3H)* allow the system to select the source to reset the D3 Pending request.

19.4 EXTI event input mapping

For the sixteen GPIO Event inputs the associated IOPORT pin has to be selected in the SYSCFG register SYSCFG_EXTICRn. The same pin from each IOPORT maps to the corresponding EXTI Event input.

The wakeup capabilities of each Event input are detailed in *Table 141*. An Event input can either wake up the CPU, and in the case of “Any” can also wake up D3 domain for autonomous Run mode.

The EXTI Event inputs with a connection to the CPU NVIC are indicated in the *Connection to NVIC* column. For the EXTI events not having a connection to the NVIC, the peripheral interrupt is directly connected to the NVIC in parallel with the connection to the EXTI.

All EXTI Event inputs are OR-ed together and connected to the CPU event input (rxev).

Table 141. EXTI Event input mapping

Event input	Source	Event input type	Wakeup target(s)	Connection to NVIC
0 - 15	EXTI[15:0]	Configurable	Any	Yes
16	PVD and AVD(1)	Configurable	CPU only	Yes
17	RTC alarms	Configurable	CPU only	Yes
18	RTC tamper, RTC timestamp, RCC LSECSS(2)	Configurable	CPU only	Yes
19	RTC wakeup timer	Configurable	Any	Yes
20	COMP1	Configurable	Any	Yes
21	COMP2	Configurable	Any	Yes
22	I2C1 wakeup	Direct	CPU only	Yes
23	I2C2 wakeup	Direct	CPU only	Yes
24	I2C3 wakeup	Direct	CPU only	Yes
25	I2C4 wakeup	Direct	Any	Yes
26	USART1 wakeup	Direct	CPU only	Yes
27	USART2 wakeup	Direct	CPU only	Yes
28	USART3 wakeup	Direct	CPU only	Yes
29	USART6 wakeup	Direct	CPU only	Yes
30	UART4 wakeup	Direct	CPU only	Yes
31	UART5 wakeup	Direct	CPU only	Yes
32	UART7 wakeup	Direct	CPU only	Yes
33	UART8 wakeup	Direct	CPU only	Yes
34	LPUART1 RX wakeup	Direct	Any	Yes
35	LPUART1 TX wakeup	Direct	Any	Yes
36	SPI1 wakeup	Direct	CPU only	Yes
37	SPI2 wakeup	Direct	CPU only	Yes
38	SPI3 wakeup	Direct	CPU only	Yes
39	SPI4 wakeup	Direct	CPU only	Yes
40	SPI5 wakeup	Direct	CPU only	Yes
41	SPI6 wakeup	Direct	Any	Yes
42	MDIO wakeup	Direct	CPU only	Yes
43	USB1 wakeup	Direct	CPU only	Yes
44	USB2 wakeup	Direct	CPU only	Yes
45	Reserved	-	-	-
46	Reserved	-	-	-
47	LPTIM1 wakeup	Direct	CPU only	Yes
48	LPTIM2 wakeup	Direct	Any	Yes

Table 141. EXTI Event input mapping (continued)

Event input	Source	Event input type	Wakeup target(s)	Connection to NVIC
49	LPTIM2 output	Configurable	Any	No(3)
50	LPTIM3 wakeup	Direct	Any	Yes
51	LPTIM3 output	Configurable	Any	No(3)
52	LPTIM4 wakeup	Direct	Any	Yes
53	LPTIM5 wakeup	Direct	Any	Yes
54	SWPMI wakeup	Direct	CPU only	Yes
55(4)	WKUP1	Direct	CPU only	Yes
56(4)	WKUP2	Direct	CPU only	Yes
57(4)	WKUP3	Direct	CPU only	Yes
58(4)	WKUP4	Direct	CPU only	Yes
59(4)	WKUP5	Direct	CPU only	Yes
60(4)	WKUP6	Direct	CPU only	Yes
61	RCC interrupt	Direct	CPU only	No(5)
62	I2C4 Event interrupt	Direct	CPU only	No(5)
63	I2C4 Error interrupt	Direct	CPU only	No(5)
64	LPUART1 global Interrupt	Direct	CPU only	No(5)
65	SPI6 interrupt	Direct	CPU only	No(5)
66	BDMA CH0 interrupt	Direct	CPU only	No(5)
67	BDMA CH1 interrupt	Direct	CPU only	No(5)
68	BDMA CH2 interrupt	Direct	CPU only	No(5)
69	BDMA CH3 interrupt	Direct	CPU only	No(5)
70	BDMA CH4 interrupt	Direct	CPU only	No(5)
71	BDMA CH5 interrupt	Direct	CPU only	No(5)
72	BDMA CH6 interrupt	Direct	CPU only	No(5)
73	BDMA CH7 interrupt	Direct	CPU only	No(5)
74	DMAMUX2 interrupt	Direct	CPU only	No(5)
75	ADC3 interrupt	Direct	CPU only	No(5)
76	SAI4 interrupt	Direct	CPU only	No(5)
77	Reserved	-	-	-
78	Reserved	-	-	-
79	Reserved	-	-	-
80	Reserved	-	-	-
81	Reserved	-	-	-
82	Reserved	-	-	-
83	Reserved	-	-	-

Table 141. EXTI Event input mapping (continued)

Event input	Source	Event input type	Wakeup target(s)	Connection to NVIC
84	Reserved	-	-	-
85	HDMI-CEC wakeup	Configurable	CPU only	Yes
86	ETHERNET wakeup	Configurable	CPU only	Yes
87	HSECSS interrupt	Direct	CPU only	No(5)
88	Reserved	-	-	-

1. PVD and AVD signals are OR-ed together on the same EXTI event input.
2. RTC Tamper, RTC timestamp and RCC LSECSS signals are OR-ed together on the same EXTI event input.
3. Not available on CPU NVIC, to be used for system wakeup only or CPU event input (rxev).
4. Signals of WKUP1 to WKUP6 correspond to WKUPn pin+1.
5. Available on CPU NVIC directly from the peripheral

19.5 EXTI functional behavior

The Direct event inputs are enabled in the respective peripheral generating the event. The Configurable events are enabled by enabling at least one of the trigger edges.

When in Stop mode an event will always wake up the D3 domain. In system Run and Stop modes an event will always generate an associated D3 domain interrupt. An event will only wake up the CPU when the event associated CPU interrupt is unmasked and/or the CPU event is unmasked.

Table 142. Masking functionality

CPU		Configurable event inputs PRx bits of EXTI_CPUPR	CPU			D3 domain wakeup
Interrupt enable MRx bits of EXTI_CPUIMR	Event enable MRx bits of EXTI_CPUEMR		Interrupt	Event	Wakeup	
0	0	No	Masked	Masked	Masked	Yes(1) / Masked(2)
0	1	No	Masked	Yes	Yes	Yes
1	0	Status latched	Yes	Masked	Yes	Yes
1	1	Status latched	Yes	Yes	Yes	Yes

1. Only for Event inputs that allow the system to wakeup D3 domain for autonomous Run mode (Any target).
2. For Event inputs that will always wake up CPU.

For Configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending PRx bit in EXTI_CPUPR is set and the CPU interrupt signal is activated. EXTI_CPUPR PRx pending bit shall be cleared by software writing it to ‘1’. This will clear the CPU interrupt.

For Direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit. When the associated CPU interrupt is unmasked the corresponding CPU interrupt signal is activated.

The CPU event has to be unmasked to generate an event. When the enabled edge(s) occur on the Event input a CPU event pulse is generated. There is no CPU Event pending bit.

Both a CPU interrupt and a CPU event may be enabled on the same Event input. They will both trigger the same Event input condition(s).

For the Configurable Event inputs an event input request can be generated by software when writing a ‘1’ in the software interrupt/event register EXTI_SWIER.

Whenever an Event input is enabled and a CPU interrupt and/or CPU event is unmasked, the Event input will also generate a D3 domain wakeup next to the CPU wakeup.

Some Event inputs are able to wakeup the D3 domain autonomous Run mode, in this case the CPU interrupt and CPU event are masked, preventing the CPU to be woken up. Two D3 domain autonomous Run mode wakeup mechanisms are supported:

- D3 domain wakeup without pending (EXTI_D3PMR = 0)
  - On a Configurable Event input this mechanism will wake up D3 domain and clear the D3 domain wakeup signal automatically after the Delay + Rising Edge detect Pulse generator.
  - On a Direct Event input this mechanism will wake up D3 domain and clear the D3 domain wakeup signal after the Direct Event input signal is cleared.
- D3 domain wakeup with pending (EXTI_D3PMR = 1)
  - On a Configurable Event input this mechanism will wake up D3 domain and clear the D3 domain wakeup signal after the Delay + Rising Edge detect Pulse generator and when the D3 Pending request is cleared.
  - On a Direct Event input this mechanism will wake up D3 domain and clear the D3 domain wakeup signal after the Direct Event input signal is cleared and when the D3 Pending request is cleared.

19.5.1 EXTI CPU interrupt procedure

- Unmask the Event input interrupt by setting the corresponding mask bits in the EXTI_CPUIMR register.
- For Configurable Event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- Enable the associated interrupt source in the CPU NVIC or use the SEVONPEND, so that an interrupt coming from the CPU interrupt signal is detectable by the CPU after a WFI/WFE instruction.
  - For Configurable event inputs the associated EXTI pending bit needs to be cleared.

19.5.2 EXTI CPU event procedure

- Unmask the Event input by setting the corresponding mask bits of the EXTI_CPUEMR register.
- For Configurable Event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- The CPU event signal is detected by the CPU after a WFE instruction.
  - For Configurable event inputs there is no EXTI pending bit to clear.

19.5.3 EXTI CPU wakeup procedure

- Unmask the Event input by setting at least one of the corresponding mask bits in the EXTI_CPUIMR and/or EXTI_CPUEMR registers. The CPU wakeup is generated at the same time as the unmasked CPU interrupt and/or CPU event.
- For Configurable Event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- Direct Events will automatically generate a CPU wakeup.

19.5.4 EXTI D3 domain wakeup for autonomous Run mode procedure

- Mask the Event input for waking up the CPU, by clearing both the corresponding mask bits in the EXTI_CPUIMR and/or EXTI_CPUEMR registers.
- For Configurable Event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- Direct Events will automatically generate a D3 domain wakeup.
- Select the D3 domain wakeup mechanism in EXTI_D3PMR.
  - When D3 domain wakeup without pending (EXTI_PMR = 0) is selected, the Wakeup will be cleared automatically following the clearing of the Event input.
  - When D3 domain wakeup with pending (EXTI_PMR = 1) is selected the Wakeup needs to be cleared by a selected D3 domain pendclear source.
    A pending D3 domain wakeup signal can also be cleared by FW clearing the associated EXTI_D3PMR register bit.
- After the D3 domain wakeup a D3 domain interrupt is generated.
  - Configurable Event inputs will generate a pulse on D3 domain interrupt.
  - Direct Event inputs will activate the D3 domain interrupt until the event input is cleared in the peripheral.

19.5.5 EXTI software interrupt/event trigger procedure

Any of the Configurable Event inputs can be triggered from the software interrupt/event register (the associated CPU interrupt and/or CPU event shall be enabled by their respective procedure).

- Enable the Event input by setting at least one of the corresponding edge trigger bits in the EXTI_RTSR and/or EXTI_FTSR registers.
- Unmask the software interrupt/event trigger by setting at least one of the corresponding mask bits in the EXTI_CPUIMR and/or EXTI_CPUEMR registers.
- Trigger the software interrupt/event by writing “1” to the corresponding bit in the EXTI_SWIER register.
- The Event input may be disabled by clearing the EXTI_RTSR and EXTI_FTSR register bits.

Note: An edge on the Configurable event input will also trigger an interrupt/event.

A software trigger can be used to set the D3 Pending request logic, keeping the D3 domain in Run until the D3 Pending request logic is cleared.

19.6 EXTI registers

Every register can only be accessed with 32-bit (word). A byte or half-word cannot be read or written.

19.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x00
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR21	TR20	TR19	TR18	TR17	TR16
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **TRx:** Rising trigger event configuration bit of Configurable Event input x.(1)

0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

1. The Configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the Configurable event input occurs during writing of the register, the associated pending bit will not be set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

19.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR21	TR20	TR19	TR18	TR17	TR16
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **TRx:** Falling trigger event configuration bit of Configurable Event input x.(1)

0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

1. The Configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the Configurable event input occurs during writing of the register, the associated pending bit will not be set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

19.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 21	SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **SWIERx:** Software interrupt on line x
Will alway return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x. This bit is auto cleared by HW.

19.6.4 EXTI D3 pending mask register (EXTI_D3PMR1)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MR25	Res.	Res.	Res.	MR21	MR20	MR19	Res.	Res.	Res.
						rw				rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MRx:** D3 Pending Mask on Event input x
0: D3 Pending request from Line x is masked. Writing this bit to 0 will also clear the D3 Pending request.
1: D3 Pending request from Line x is unmasked. The D3 domain pending signal when triggered will keep D3 domain wakeup active until cleared.

Bits 24:22 Reserved, must be kept at reset value.

Bits 21:19 **MRx:** D3 Pending Mask on Event input x

0: D3 Pending request from Line x is masked. Writing this bit to 0 will also clear the D3 Pending request.

1: D3 Pending request from Line x is unmasked. The D3 domain pending signal when triggered will keep D3 domain wakeup active until cleared.

Bits 18:16 Reserved, must be kept at reset value.

Bits 15:0 **MRx:** D3 Pending Mask on Event input x

0: D3 Pending request from Line x is masked. Writing this bit to 0 will also clear the D3 Pending request.

1: D3 Pending request from Line x is unmasked. The D3 domain pending signal when triggered will keep D3 domain wakeup active until cleared.

19.6.5 EXTI D3 pending clear selection register low (EXTI_D3PCR1L)

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS15		PCS14		PCS13		PCS12		PCS11		PCS10		PCS9		PCS8	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS7		PCS6		PCS5		PCS4		PCS3		PCS2		PCS1		PCS0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PCSx:** D3 Pending request clear input signal selection on Event input x = truncate (n/2)

00: DMA ch6 event selected as D3 domain pendclear source

01: DMA ch7 event selected as D3 domain pendclear source

10: LPTIM4 out selected as D3 domain pendclear source

11: LPTIM5 out selected as D3 domain pendclear source

19.6.6 EXTI D3 pending clear selection register high (EXTI_D3PCR1H)

Address offset: 0x14
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS25		Res.	Res.
												rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PCS21		PCS20		PCS19		Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw	rw	rw						

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:18 **PCSx:** D3 Pending request clear input signal selection on Event input x = truncate ((n+32)/2)
00: DMA ch6 event selected as D3 domain pendclear source
01: DMA ch7 event selected as D3 domain pendclear source
10: LPTIM4 out selected as D3 domain pendclear source
11: LPTIM5 out selected as D3 domain pendclear source

Bits 17:12 Reserved, must be kept at reset value.

Bits 11:6 **PCSx:** D3 Pending request clear input signal selection on Event input x = truncate ((n+32)/2)
00: DMA ch6 event selected as D3 domain pendclear source
01: DMA ch7 event selected as D3 domain pendclear source
10: LPTIM4 out selected as D3 domain pendclear source
11: LPTIM5 out selected as D3 domain pendclear source

Bits 5:0 Reserved, must be kept at reset value.

19.6.7 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x20
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR51	Res.	TR49	Res.
												rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TRx:** Rising trigger event configuration bit of Configurable Event input x+32.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bit 18 Reserved, must be kept at reset value.

Bit 17 **TRx:** Rising trigger event configuration bit of Configurable Event input x+32.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bits 16:0 Reserved, must be kept at reset value.

1. The Configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the Configurable event input occurs during writing of the register, the associated pending bit will not be set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

19.6.8 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x24
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR51	Res.	TR49	Res.
												rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TRx:** Falling trigger event configuration bit of Configurable Event input x+32.(1)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

Bit 18 Reserved, must be kept at reset value.

Bit 17 **TRx:** Falling trigger event configuration bit of Configurable Event input x+32.(1)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

Bits 16:0 Reserved, must be kept at reset value.

1. The Configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the Configurable event input occurs during writing of the register, the associated pending bit will not be set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

19.6.9 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x28
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 51	Res.	SWIER 49	Res.
												rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **SWIERx:** Software interrupt on line x+32
Will alway return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x. This bit is auto cleared by HW.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **SWIERx:** Software interrupt on line x+32
Will alway return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x. This bit is auto cleared by HW.

Bits 16:0 Reserved, must be kept at reset value.

19.6.10 EXTI D3 pending mask register (EXTI_D3PMR2)

Address offset: 0x2C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR53	MR52	MR51	MR50	MR49	MR48
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MR41	Res.	Res.	Res.	Res.	Res.	MR35	MR34	Res.	Res.
						rw						rw	rw		

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **MRx:** D3 Pending Mask on Event input x+32
0: D3 Pending request from Line x+32 is masked. Writing this bit to 0 will also clear the D3 Pending request.
1: D3 Pending request from Line x+32 is unmasked. The D3 domain pending signal when triggered will keep D3 domain wakeup active until cleared.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MRx:** D3 Pending Mask on Event input x+32
0: D3 Pending request from Line x+32 is masked. Writing this bit to 0 will also clear the D3 Pending request.
1: D3 Pending request from Line x+32 is unmasked. The D3 domain pending signal when triggered will keep D3 domain wakeup active until cleared.

Bits 8:4 Reserved, must be kept at reset value.

Bits 3:2 **MRx:** D3 Pending Mask on Event input x+32
0: D3 Pending request from Line x+32 is masked. Writing this bit to 0 will also clear the D3 Pending request.
1: D3 Pending request from Line x+32 is unmasked. The D3 domain pending signal when triggered will keep D3 domain wakeup active until cleared.

Bits 1:0 Reserved, must be kept at reset value.

19.6.11 EXTI D3 pending clear selection register low (EXTI_D3PCR2L)

Address offset: 0x30
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS41		Res.	Res.
												rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS35		PCS34		Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:18 **PCSx:** D3 Pending request clear input signal selection on Event input x = truncate ((n+64)/2)
00: DMA ch6 event selected as D3 domain pendclear source
01: DMA ch7 event selected as D3 domain pendclear source
10: LPTIM4 out selected as D3 domain pendclear source
11: LPTIM5 out selected as D3 domain pendclear source

Bits 17:8 Reserved, must be kept at reset value.

Bits 7:4 **PCSx:** D3 Pending request clear input signal selection on Event input x= truncate ((n+64)/2)
00: DMA ch6 event selected as D3 domain pendclear source
01: DMA ch7 event selected as D3 domain pendclear source
10: LPTIM4 out selected as D3 domain pendclear source
11: LPTIM5 out selected as D3 domain pendclear source

Bits 3:0 Reserved, must be kept at reset value.

19.6.12 EXTI D3 pending clear selection register high (EXTI_D3PCR2H)

Address offset: 0x34
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PCS53		PCS52		PCS51		PCS50		PCS49		PCS48	
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **PCSx:** D3 Pending request clear input signal selection on Event input x= truncate ((n+96)/2)
00: DMA ch6 event selected as D3 domain pendclear source
01: DMA ch7 event selected as D3 domain pendclear source
10: LPTIM4 out selected as D3 domain pendclear source
11: LPTIM5 out selected as D3 domain pendclear source

19.6.13 EXTI rising trigger selection register (EXTI_RTSR3)

Address offset: 0x40
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR86	TR85	TR84	Res.	TR82	Res.	Res.
									rw	rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **TRx:** Rising trigger event configuration bit of Configurable Event input x+64.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TRx:** Rising trigger event configuration bit of Configurable Event input x+64.(1)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Bits 17:0 Reserved, must be kept at reset value.

1. The Configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the Configurable event input occurs during writing of the register, the associated pending bit will not be set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

19.6.14 EXTI falling trigger selection register (EXTI_FTSR3)

Address offset: 0x44
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR86	TR85	TR84	Res.	TR82	Res.	Res.
									rw	rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **TRx:** Falling trigger event configuration bit of Configurable Event input x+64.(1)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TRx:** Falling trigger event configuration bit of Configurable Event input x+64.(1)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

Bits 17:0 Reserved, must be kept at reset value.

1. The Configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the Configurable event input occurs during writing of the register, the associated pending bit will not be set.
Rising and falling edge triggers can be set for the same Configurable Event input. In this case, both edges generate a trigger.

19.6.15 EXTI software interrupt event register (EXTI_SWIER3)

Address offset: 0x48
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 86	SWIER 85	SWIER 84	Res.	SWIER 82	Res.	Res.
									rw	rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **SWIERx:** Software interrupt on line x+64
Will alway return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x. This bit is auto cleared by HW.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SWIERx:** Software interrupt on line x+64
Will alway return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger an event on line x. This bit is auto cleared by HW.

Bits 17:0 Reserved, must be kept at reset value.

19.6.16 EXTI D3 pending mask register (EXTI_D3PMR3)

Address offset: 0x4C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR88	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MRx:** D3 Pending Mask on Event input x+64

0: D3 Pending request from Line x+64 is masked. Writing this bit to 0 will also clear the D3 Pending request.
1: D3 Pending request from Line x+64 is unmasked. The D3 domain pending signal when triggered will keep D3 domain wakeup active until cleared.

Bits 23:0 Reserved, must be kept at reset value.

19.6.17 EXTI D3 pending clear selection register low (EXTI_D3PCR3L)

Address offset: 0x50
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

19.6.18 EXTI D3 pending clear selection register high (EXTI_D3PCR3H)

Address offset: 0x54
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS88	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **PCSx:** D3 Pending request clear input signal selection on Event input x= truncate ((n+160)/2)

00: DMA ch6 event selected as D3 domain pendclear source
01: DMA ch7 event selected as D3 domain pendclear source
10: LPTIM4 out selected as D3 domain pendclear source
11: LPTIM5 out selected as D3 domain pendclear source

Bits 15:0 Reserved, must be kept at reset value.

19.6.19 EXTI interrupt mask register (EXTI_CPUIMR1)

Address offset: 0x80

Reset value: 0xFFC0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR31	MR30	MR29	MR28	MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20	MR19	MR18	MR17	MR16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 **MRx:** CPU interrupt Mask on Direct Event input x[(1)]

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is unmasked

Bits 21:0 **MRx:** CPU interrupt Mask on Configurable Event input x [(2)]

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is unmasked

1. The reset value for Direct Event inputs is set to ‘1’ in order to enable the interrupt by default.
2. The reset value for Configurable Event inputs is set to ‘0’ in order to disable the interrupt by default.

19.6.20 EXTI event mask register (EXTI_CPUEMR1)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR31	MR30	MR29	MR28	MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20	MR19	MR18	MR17	MR16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MRx:** CPU Event mask on Event input x

0: Event request from Line x is masked

1: Event request from Line x is unmasked

19.6.21 EXTI pending register (EXTI_CPUPR1)

Address offset: 0x88

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR21	PR21	PR19	PR18	PR17	PR16
										rc1	rc1	rc1	rc1	rc1	rc1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1	rc1

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **PRx:** Configurable event inputs x Pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

19.6.22 EXTI interrupt mask register (EXTI_CPUIMR2)

Address offset: 0x90

Reset value: 0xFFF5 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR63	MR62	MR61	MR60	MR59	MR58	MR57	MR56	MR55	MR54	MR53	MR52	MR51	MR50	MR49	MR48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR47	MR46	Res.	MR44	MR43	MR42	MR41	MR40	MR39	MR38	MR37	MR36	MR35	MR34	MR33	MR32
rw	rw	1	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 **MRx:** CPU Interrupt Mask on Direct Event input x+32(1)

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is unmasked

Bit 19 **MRx:** CPU interrupt Mask on Configurable Event input x+32 (2)

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is unmasked

Bit 18 **MRx:** CPU Interrupt Mask on Direct Event input x+32 (1)

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is unmasked

Bit 17 **MRx:** CPU interrupt Mask on Configurable Event input x+32 (2)

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is unmasked

Bits 16:14 **MRx:** CPU Interrupt Mask on Direct Event input x+32 (1)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bit 13 Reserved, must be kept at reset value (1).

Bits 12:0 **MRx:** CPU Interrupt Mask on Direct Event input x+32 (1)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

1. The reset value for Direct Event inputs is set to '1' in order to enable the interrupt by default.
2. The reset value for Configurable Event inputs is set to '0' in order to disable the interrupt by default.

19.6.23 EXTI event mask register (EXTI_CPUEMR2)

Address offset: 0x94

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR63	MR62	MR61	MR60	MR59	MR58	MR57	MR56	MR55	MR54	MR53	MR52	MR51	MR50	MR49	MR48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR47	MR46	Res.	MR44	MR43	MR42	MR41	MR40	MR39	MR38	MR37	MR36	MR35	MR34	MR33	MR32
rw	rw	0	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 **MRx:** CPU Event mask on Event input x+32
0: Event request from Line x is masked
1: Event request from Line x is unmasked

Bit 13 Reserved, must be kept at reset value.

Bits 12:0 **MRx:** CPU Event mask on Event input x+32
0: Event request from Line x is masked
1: Event request from Line x is unmasked

19.6.24 EXTI pending register (EXTI_CPUPR2)

Address offset: 0x98

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR51	Res.	PR49	Res.
												rc1		rc1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **PRx:** Configurable event inputs x+32 Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PRx:** Configurable event inputs x+32 Pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 16:0 Reserved, must be kept at reset value.

19.6.25 EXTI interrupt mask register (EXTI_CPUIMR3)

Address offset: 0xA0

Reset value: 0x018B FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR88	MR87	MR86	MR85	MR84	Res.	MR82	Res.	MR80
							rw	rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR79	MR78	MR77	MR76	MR75	MR74	MR73	MR72	MR71	MR70	MR69	MR68	MR67	MR66	MR65	MR64
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:23 **MRx:** CPU Interrupt Mask on Direct Event input x+64 (1)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bits 22:20 **MRx:** CPU interrupt Mask on Configurable Event input x+64 (2)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bit 19 Reserved, must be kept at reset value (1).

Bit 18 **MRx:** CPU interrupt Mask on Configurable Event input x+64 (2)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

Bit 17 Reserved, must be kept at reset value (1).

Bits 16:0 **MRx:** CPU Interrupt Mask on Direct Event input x+64 (1)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is unmasked

1. The reset value for Direct Event inputs is set to ‘1’ in order to enable the interrupt by default.
2. The reset value for Configurable Event inputs is set to ‘0’ in order to disable the interrupt by default.

19.6.26 EXTI event mask register (EXTI_CPUEMR3)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR88	MR87	MR86	MR85	MR84	Res.	MR82	Res.	MR80
							rw	rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR79	MR78	MR77	MR76	MR75	MR74	MR73	MR72	MR71	MR70	MR69	MR68	MR67	MR66	MR65	MR64
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **MRx:** CPU Event mask on Event input x+64

0: Event request from Line x is masked
1: Event request from Line x is unmasked

19.6.27 EXTI pending register (EXTI_CPUPR3)

Address offset: 0xA8

Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR86	PR85	PR84	Res.	PR82	Res.	Res.
									rc1	rc1	rc1		rc1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **PRx:** Configurable event inputs x+64 Pending bit

0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **PRx:** Configurable event inputs x+64 Pending bit

0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 17:0 Reserved, must be kept at reset value.

19.6.28 EXTI register map

The following table gives the EXTI register map and the reset values.

Table 143. Asynchronous interrupt/event controller register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**EXTI_RTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**EXTI_FTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**EXTI_SWIER1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**EXTI_D3PMR1**	Res.	Res.	Res.	Res.	Res.	Res.	MR[25]	Res.	Res.	Res.	MR[21:19]			Res.	Res.	Res.	MR[15:0]															
	Reset value							0				0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**EXTI_D3PCR1L**	PCS[15]		PCS[14]		PCS[13]		PCS[12]		PCS[11]		PCS[10]		PCS[9]		PCS[8]		PCS[7]		PCS[6]		PCS[5]		PCS[4]		PCS[3]		PCS[2]		PCS[1]		PCS[0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_D3PCR1H**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS[25]		Res.	Res.	Res.	Res.	Res.	Res.	PCS[21]		PCS[20]		PCS[19]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0							0	0	0	0	0	0						
0x20	**EXTI_RTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[51]	Res.	TR[49]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0		0																	
0x24	**EXTI_FTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[51]	Res.	TR[49]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0		0																	
0x28	**EXTI_SWIER2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER[51]	Res.	SWIER[49]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0		0																	
0x2C	**EXTI_D3PMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR[53:48]						Res.	Res.	Res.	Res.	Res.	Res.	MR[41]	Res.	Res.	Res.	Res.	Res.	MR[35]	MR[34]	Res.	Res.
	Reset value											0	0	0	0	0	0							0						0	0		
0x30	**EXTI_D3PCR2L**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS[41]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS[35]		PCS[34]		Res.	Res.	Res.	Res.
	Reset value													0	0											0	0	0	0				
0x34	**EXTI_D3PCR2H**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS[53]		PCS[52]		PCS[51]		PCS[50]		PCS[49]		PCS[48]	
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x40	**EXTI_RTSR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[86]	TR[85]	TR[84]	Res.	TR[82]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0		0																		

Table 143. Asynchronous interrupt/event controller register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44	**EXTI_FTSR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR[86]	TR[85]	TR[84]	Res.	TR[82]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0		0																		
0x48	**EXTI_SWIER3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER[86]	SWIER[85]	SWIER[84]	Res.	SWIER[82]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0		0																		
0x4C	**EXTI_D3PMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR[88]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0																								
0x50	**EXTI_D3PCR3L**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x54	**EXTI_D3PCR3H**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCS[88]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0																
0x58-0x7C	Reserved																																
0x80	**EXTI_CPUIMR1**	MR[31:22]										MR[21:0]																					
	Reset value	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	**EXTI_CPUEMR1**	MR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	**EXTI_CPUPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	**EXTI_CPUIMR2**	MR[63:52]												MR[51]	MR[50]	MR[49]	MR[48:46]			Res.	MR[44:32]												
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1
0x94	**EXTI_CPUEMR2**	MR[63:46]																		Res.	MR[44:32]												
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	**EXTI_CPUPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[51]	Res.	PR[49]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0		0																	
0xA0	**EXTI_CPUIMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR[88]	MR[87]	MR[86]	MR[85]	MR[84]	Res.	MR[82]	Res.	MR[80:64]																
	Reset value								1	1	0	0	0		0		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0xA4	**EXTI_CPUEMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR[88:84]					Res.	MR[82]	Res.	MR[80:64]																
	Reset value								0	0	0	0	0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 143. Asynchronous interrupt/event controller register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA8	**EXTI_CPUPR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[86]	PR[85]	PR[84]	Res.	PR[82]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0		0																		
0xAC-0xBC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

20 Cyclic redundancy check calculation unit (CRC)

20.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

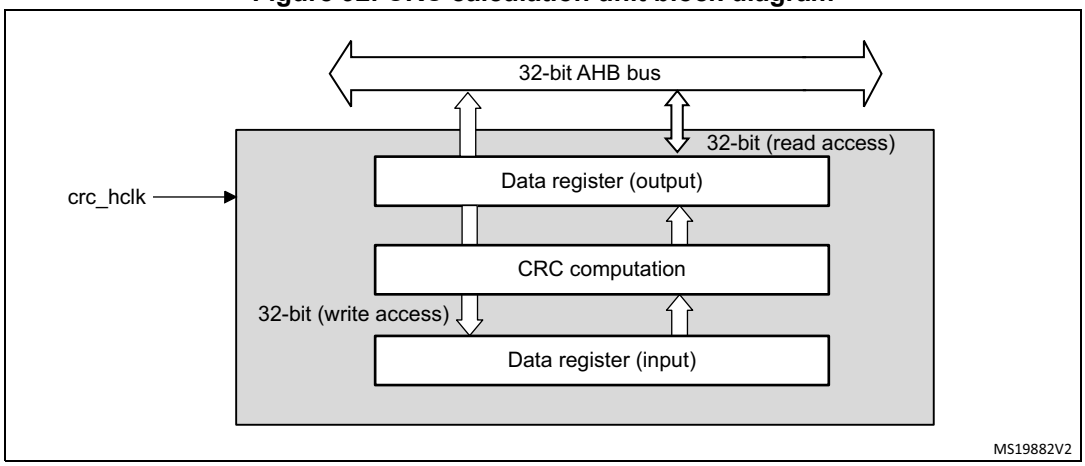
20.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data

20.3 CRC functional description

20.3.1 CRC block diagram

Figure 92. CRC calculation unit block diagram



20.3.2 CRC internal signals

Table 144. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

20.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

An input buffer allows to immediately write a second data without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

20.4 CRC registers

20.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.
It holds the previous CRC calculation result when it is read.
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

20.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

20.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **REV_OUT**: Reverse output data
This bit controls the reversal of the bit order of the output data.
0: Bit order not affected
1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data
These bits control the reversal of the bit order of the input data
00: Bit order not affected
01: Bit reversal done by byte
10: Bit reversal done by half-word
11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size
These bits control the size of the polynomial.
00: 32 bit polynomial
01: 16 bit polynomial
10: 8 bit polynomial
11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit
This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

20.4.4 Initial CRC value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]**: *Programmable initial CRC value*
This register is used to write the CRC initial value.

20.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

20.4.6 CRC register map

Table 145. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CRC_DR**	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**CRC_IDR**	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**CRC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	**CRC_INIT**	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	**CRC_POL**	Polynomial coefficients																															
	Reset value	0x04C11DB7																															

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

21 Flexible memory controller (FMC)

The Flexible memory controller (FMC) includes three memory controllers:

- The NOR/PSRAM memory controller
- The NAND memory controller
- The Synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) controller

21.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, SDRAM memories, and NAND flash memory. Its main purposes are:

- to translate AXI transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC controller are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR Flash memory/OneNAND Flash memory
  - PSRAM (4 memory banks)
  - NAND Flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) memories
- Burst mode support for faster access to synchronous devices such as NOR Flash memory, PSRAM and SDRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16- or 32-bit wide data bus
- Independent Chip Select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM and SDRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth

  The Write FIFO is common to all memory controllers and consists of:
  - a Write Data FIFO which stores the data to be written to the memory
  - a Write Address FIFO which stores the address (up to 28 bits) plus the data size (up to 2 bits). When operating in Burst mode, only the start address is stored except when crossing a page boundary (for PSRAM and SDRAM). In this case, the burst is broken into two FIFO entries.
- Cacheable Read FIFO with 6 x64-bit depth (6 x14-bit address tag) for SDRAM controller.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.

The FMC registers that define the external device type and associated characteristics are set at boot time and do not change until the next reset or power-up. However, only a few bits can be changed on-the-fly:

- ECCEN and PBKEN bits in the FMC_PCR register
- IFS, IRS and ILS bits in the FMC_SR register
- MODE[2:0], CTB1/CTB2, NRFS and MRD bits in the FMC_SDCMR register
- REIE and CRE bits in the FMC_SDRTR register.

Follow the below sequence to modify parameters while the FMC is enabled:

1. First disable the FMC controller to prevent further accesses to any memory controller while the register is modified.
2. Update all required configurations.
3. Enable the FMC controller again.

When the SDRAM controller is used, if the SDCLK Clock ratio or refresh rate has to be modified after initialization phase, the following procedure must be followed.

1. Put the SDRAM device in Self-refresh mode.
2. Disable the FMC controller by resetting the FMCEN bit in the FMC_BCR1 register.
3. Update the required parameters.
4. Enable the FMC controller once all parameters have been updated.
5. Then, send the Clock Configuration Enable command to exit Self-fresh mode.

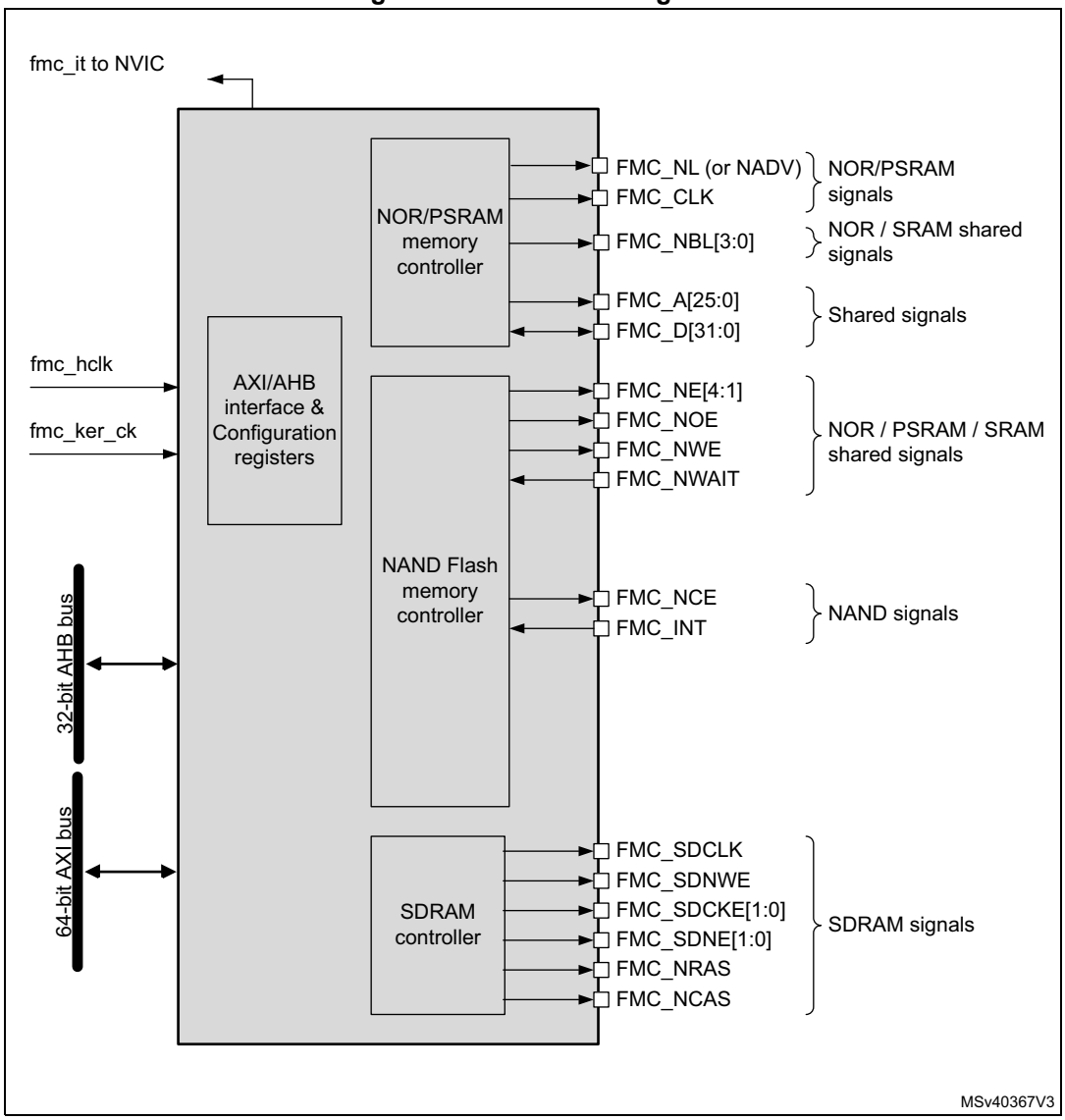
21.2 FMC block diagram

The FMC consists of the following main blocks:

- The NOR Flash/PSRAM/SRAM controller
- The NAND controller
- The SDRAM controller
- The AXI interface
- The AHB interface (including the FMC configuration registers)

The block diagram is shown in the figure below.

Figure 93. FMC block diagram



21.3 FMC internal signals

Table 146 gives the list of FMC internal signals. FMC pins (or external signals) are described in *Section 21.7.1: External memory interface signals*.

Table 146. FMC pins

Names	Signal type	Description
fmc_it	Digital output	FMC interrupt
fmc_ker_ck	Digital input	FMC kernel clock
fmc_hclk	Digital input	FMC interface clock

21.4 AHB interface

The AHB slave interface allows internal CPUs to configure the FMC registers.

The AHB clock (fmc_hclk) is the reference clock for the FMC register accesses.

21.5 AXI interface

The AXI slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AXI transactions are translated into the external device protocol. As the AXI data bus is 64-bit wide, the AXI transactions might be split into several consecutive 32-, 16- or 8-bit accesses according to data size accesses. The FMC Chip Select (FMC_NEx) does not toggle between consecutive accesses except in case of accesses in Mode D when the Extended mode is enabled.

The FMC generates an AXI slave error when one of the following conditions is met:

- Reading or writing to an FMC bank (Bank 1 to 4) which is not enabled.
- Reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FMC_BCRx register.
- Writing to a write protected SDRAM bank (WP bit set in the SDRAM_SDCRx register).
- Violation of the SDRAM address range (access to reserved address range)
- Attempting to read/write access from/to SDRAM bank when it is not yet initialized

The FMC generates an AXI decoder error when ADDR[31:28] address bits are not supported by the FMC bank base address following the BMAP[1:0] bits configuration.

The kernel clock for the FMC controller is the asynchronous fmc_ker_ck clock (refer to *Section 7: Reset and Clock Control (RCC)* for fmc_ker_ck clock source selection).

21.5.1 Supported memories and transactions

General transaction rules

The requested AXI transaction data size can be 8-, 16-, 32- or 64-bit wide whereas the accessed external device has a fixed data width. The AXI transaction size depends on memory region MPU attributes. This may lead to inconsistent transfers.

Therefore, some simple transaction rules can be followed, depending on AXI transaction size versus memory data size:

- AXI transaction size and memory data size are equal to prevent issues from occurring.
- AXI transaction size is greater than the memory size:

  In this case, the FMC splits the AXI transaction into smaller consecutive memory accesses to meet the external data width.

  The memory region MPU attribute must be configured as “Device” or “strongly-ordered” in order to prevent issues on devices that do not have the byte select feature. As an example, if byte accesses are performed to 16-bit memories with the MPU attribute set to “Normal memory”, the AXI transaction size is 64 bits and the FMC splits the transaction into four consecutive accesses while only one access is expected instead of four.
- AXI transaction size is smaller than the memory size:

  The transfer may or not be consistent depending on the type of external device:
  - Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM, SDRAM)

    In this case, the FMC allows read/write transactions and accesses the right data through its NBL[3:0] byte lanes.

    Bytes to be written are addressed by NBL[3:0].

    All memory bytes are read (NBL[3:0] are driven low during read transaction) and the useless ones are discarded.
  - Accesses to devices that do not have the byte select feature (NOR and NAND Flash memories)

    This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the Flash memory), write transactions are not allowed while read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Wrap support for NOR Flash/PSRAM and SDRAM

The synchronous memories must be configured in Linear burst mode of undefined length as not all masters can issue wrap transactions.

If a master generates a wrap transaction:

- The read is split into two linear burst transactions.
- The write is split into two linear burst transactions if the write FIFO is enabled and into several linear burst transactions if the write FIFO is disabled.

Configuration registers

The FMC can be configured through a set of registers. Refer to *Section 21.7.6*, for a detailed description of the NOR Flash/PSRAM controller registers. Refer to *Section 21.8.7*,

for a detailed description of the NAND Flash registers and to *Section 21.9.5* for a detailed description of the SDRAM controller registers.

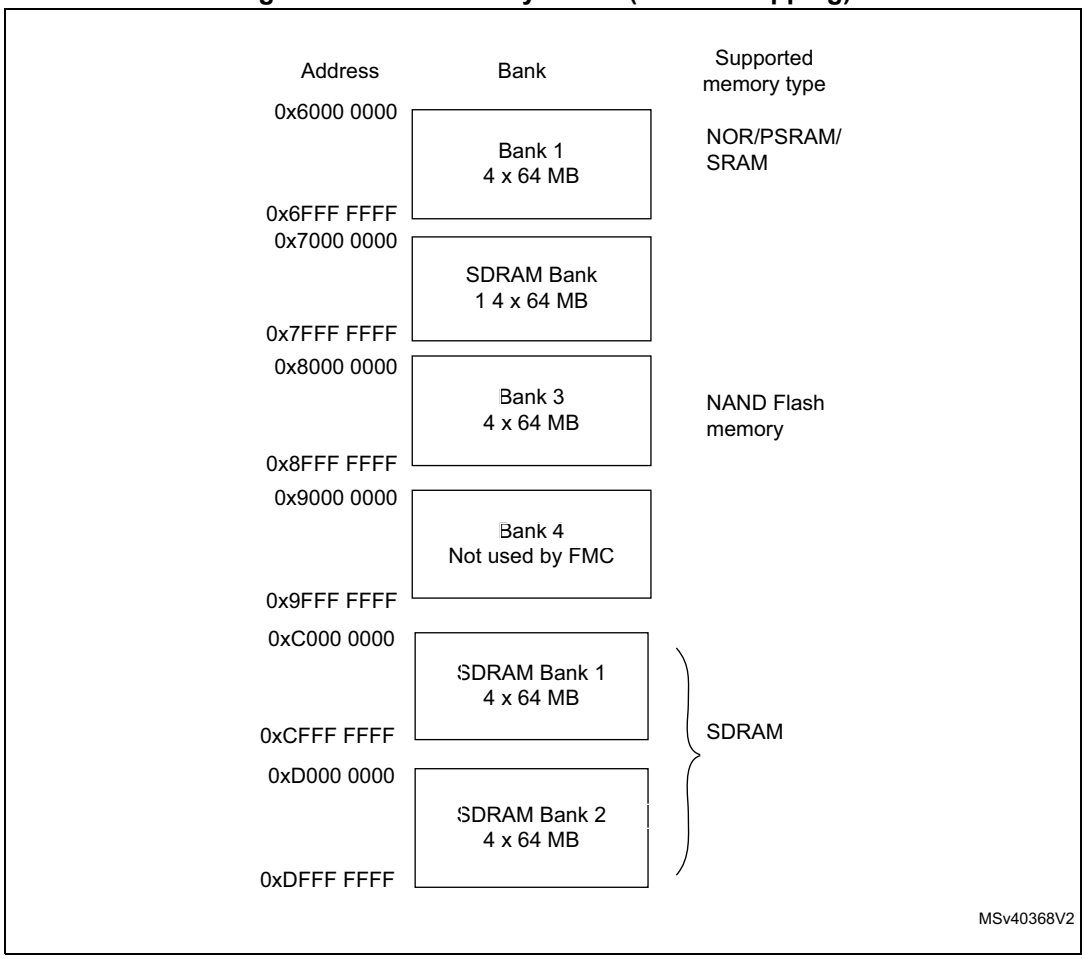
21.6 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see *Figure 94*):

- Bank 1 is used to address up to 4 NOR Flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated Chip Selects, as follows:
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 2 is used for SDRAM device, SDRAM bank 1 or SDRAM bank 2 depending on BMAP bits configuration.
- Bank 3 is used to address NAND Flash memory devices.The MPU memory attribute for this space must be reconfigured by software to Device.
- Bank 5 and 6 are used to address SDRAM devices (1 device per bank).

For each bank the type of memory to be used can be configured by the user application through the Configuration register.

Figure 94. FMC memory banks (default mapping)



The FMC bank mapping can be modified through the BMAP[1:0] bits in the FMC_BCR1 register. *Table 147* shows the configuration to swap the NOR/PSRAM bank with SDRAM banks or remap the SDRAM bank2, thus allowing to access the SDRAM banks at two different address mapping.

Table 147. FMC bank mapping options

Start -End address	BMAP[1:0]=00 (Default mapping)	BMAP[1:0]=01 NOR/PSRAM and SDRAM banks swapped	BMAP[1:0]=10 SDRAM bank 2 remapped
0x6000 0000 - 0x6FFF FFFF	NOR/PSRAM bank	SDRAM bank1	NOR/PSRAM bank
0x7000 0000 - 0x7FFF FFFF	SDRAM bank1	SDRAM bank2	SDRAM bank2
0x8000 0000 - 0x8FFF FFFF	NAND bank	NAND bank	NAND bank
0x9000 0000 - 0x9FFF FFFF	Not used by FMC		
0xC000 0000 - 0xCFFF FFFF	SDRAM bank1	NOR/PSRAM bank	SDRAM bank1
0xD000 0000 - 0xDFFF FFFF	SDRAM bank2	SDRAM bank2	SDRAM bank2

21.6.1 NOR/PSRAM address mapping

ADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 148*.

Table 148. NOR/PSRAM bank selection

ADDR[27:26](1)	Selected bank
00	Bank 1 - NOR/PSRAM 1
01	Bank 1 - NOR/PSRAM 2
10	Bank 1 - NOR/PSRAM 3
11	Bank 1 - NOR/PSRAM 4

1. ADDR are internal address lines that are translated to external memory.

The ADDR[25:0] bits contain the external memory address. Since ADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 149. NOR/PSRAM External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	ADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	ADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit
32-bit	ADDR[25:2] >> 2	64 Mbytes/4 x 32 = 512 Mbit

1. In case of a 16-bit external memory width, the FMC will internally use ADDR[25:1] to generate the address for external memory FMC_A[24:0]. In case of a 32-bit memory width, the FMC will internally use ADDR[25:2] to generate the external address.
Whatever the external memory width, FMC_A[0] should be connected to external memory address A[0].

21.6.2 NAND Flash memory address mapping

The NAND bank is divided into memory areas as indicated in *Table 150*.

Table 150. NAND memory mapping and timing registers

Start address	End address	FMC bank	Memory space	Timing register
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FMC_PATT (0x8C)
0x8000 0000	0x83FF FFFF		Common	FMC_PMEM (0x88)

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (see in *Table 151* below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

Table 151. NAND bank selection

Section name	ADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFF

The application software uses the 3 sections to access the NAND Flash memory:

- **To send a command to NAND Flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND Flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data,** the software reads or writes the data from/to any memory location in the data section.

Since the NAND Flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

21.6.3 SDRAM address mapping

Two SDRAM banks are available as indicated in *Table 152*.

Table 152. SDRAM bank selection

Selected bank	Control register	Timing register
SDRAM Bank1	FMC_SDCR1	FMC_SDTR1
SDRAM Bank2	FMC_SDCR2	FMC_SDTR2

Table 153 shows SDRAM mapping for a 13-bit row and an 11-bit column configuration.

Table 153. SDRAM address mapping

Memory width(1)	Internal bank	Row address	Column address(2)	Maximum memory capacity (Mbytes)
8-bit	ADDR[25:24]	ADDR[23:11]	ADDR[10:0]	64 Mbytes: 4 x 8K x 2K
16-bit	ADDR[26:25]	ADDR[24:12]	ADDR[11:1]	128 Mbytes: 4 x 8K x 2K x 2
32-bit	ADDR[27:26]	ADDR[25:13]	ADDR[12:2]	256 Mbytes: 4 x 8K x 2K x 4

1. When interfacing with a 16-bit memory, the FMC internally uses the ADDR[11:1] internal address lines to generate the external address. When interfacing with a 32-bit memory, the FMC internally uses ADDR[12:2] lines to generate the external address. Whatever the memory width, FMC_A[0] has to be connected to the external memory address A[0].
2. The AutoPrecharge is not supported. FMC_A[10] must be connected to the external memory address A[10] but it will be always driven low.

The ADDR[27:0] bits are translated into an external SDRAM address depending on the SDRAM controller configuration:

- Data size:8, 16 or 32 bits
- Row size:11, 12 or 13 bits
- Column size: 8, 9, 10 or 11 bits
- Number of internal banks: two or four internal banks

The following tables show the SDRAM address mapping versus the SDRAM controller configuration.

Table 154. SDRAM address mapping with 8-bit data bus width(1)(2)

<table>
<tr><th rowspan="2">Row size configuration</th><th colspan="28">ADDR(Internal Address Lines)</th></tr>
<tr><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="4">11-bit row size configuration</td><td colspan="8">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[10:0]</td><td colspan="8">Column[7:0]</td></tr>
<tr><td colspan="7">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[10:0]</td><td colspan="8">Column[8:0]</td></tr>
<tr><td colspan="6">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[10:0]</td><td colspan="8">Column[9:0]</td></tr>
<tr><td colspan="5">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[10:0]</td><td colspan="8">Column[10:0]</td></tr>
<tr><td rowspan="4">12-bit row size configuration</td><td colspan="7">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[11:0]</td><td colspan="8">Column[7:0]</td></tr>
<tr><td colspan="6">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[11:0]</td><td colspan="8">Column[8:0]</td></tr>
<tr><td colspan="5">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[11:0]</td><td colspan="8">Column[9:0]</td></tr>
<tr><td colspan="4">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[11:0]</td><td colspan="8">Column[10:0]</td></tr>
<tr><td rowspan="4">13-bit row size configuration</td><td colspan="6">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[12:0]</td><td colspan="8">Column[7:0]</td></tr>
<tr><td colspan="5">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[12:0]</td><td colspan="8">Column[8:0]</td></tr>
<tr><td colspan="4">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[12:0]</td><td colspan="8">Column[9:0]</td></tr>
<tr><td colspan="3">Res.</td><td colspan="2">Bank [1:0]</td><td colspan="10">Row[12:0]</td><td colspan="8">Column[10:0]</td></tr>
</table>

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved (Res.) address range generates an AXI slave error.

Table 155. SDRAM address mapping with 16-bit data bus width(1)(2)

Row size Configuration	ADDR(address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]											Column[7:0]								BM0(3)
	Res.							Bank [1:0]				Row[10:0]									Column[8:0]							BM0
	Res.								Bank [1:0]					Row[10:0]								Column[9:0]						BM0
	Res.									Bank [1:0]						Row[10:0]							Column[10:0]					BM0
12-bit row size configuration	Res.							Bank [1:0]				Row[11:0]									Column[7:0]							BM0
	Res.								Bank [1:0]					Row[11:0]								Column[8:0]						BM0
	Res.									Bank [1:0]						Row[11:0]							Column[9:0]					BM0
	Res.										Bank [1:0]							Row[11:0]						Column[10:0]				BM0
13-bit row size configuration	Res.								Bank [1:0]							Row[12:0]								Column[7:0]				BM0
	Res.									Bank [1:0]								Row[12:0]							Column[8:0]			BM0
	Res.										Bank [1:0]									Row[12:0]							Column[9:0]		BM0
	Res.											Bank [1:0]										Row[12:0]						Column[10:0]	BM0

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved space (Res.) generates an AXI Slave error.
3. BM0: is the byte mask for 16-bit access.

Table 156. SDRAM address mapping with 32-bit data bus width(1)(2)

Row size configuration	ADDR(address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.					Bank [1:0]		Row[10:0]											Column[7:0]								BM[1:0](3)	
	Res.						Bank [1:0]		Row[10:0]										Column[8:0]								BM[1:0	
	Res.							Bank [1:0]		Row[10:0]									Column[9:0]								BM[1:0	
	Res.								Bank [1:0]		Row[10:0]								Column[10:0]								BM[1:0	
12-bit row size configuration	Res.						Bank [1:0]		Row[11:0]										Column[7:0]								BM[1:0	
	Res.							Bank [1:0]		Row[11:0]									Column[8:0]								BM[1:0	
	Res.								Bank [1:0]		Row[11:0]								Column[9:0]								BM[1:0	
	Res.									Bank [1:0]		Row[11:0]							Column[10:0]								BM[1:0	
13-bit row size configuration	Res.							Bank [1:0]		Row[12:0]									Column[7:0]								BM[1:0	
	Res.								Bank [1:0]		Row[12:0]								Column[8:0]								BM[1:0	
	Res.									Bank [1:0]		Row[12:0]							Column[9:0]								BM[1:0	
	Bank [1:0]										Row[12:0]								Column[10:0]								BM[1:0	

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved space (Res.) generates an AXI slave error.
3. BM[1:0]: is the byte mask for 32-bit access.

21.7 NOR Flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
  - 8 bits
  - 16 bits
  - 32 bits
- PSRAM (Cellular RAM)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR Flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique Chip Select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC output Clock (FMC_CLK) is a sub-multiple of the fmc_ker_ck clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (FMC_CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in Synchronous mode (see *Section 21.7.6: NOR/PSRAM controller registers*). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AXI data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency will be changed depending on AXI data transaction (refer to *Section 21.7.5: Synchronous transactions* for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see *Section 21.7.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 157*) and support for wait management (for PSRAM and NOR Flash memory accessed in Burst mode).

Table 157. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	FMC clock cycle (fmc_ker_ck)	1	15
Data setup	Duration of the data setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	1	256
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	FMC clock cycle (fmc_ker_ck)	0	15
Clock divide ratio	Number of FMC clock cycles (fmc_ker_ck) to build one memory clock cycle (CLK)	Synchronous	FMC clock cycle (fmc_ker_ck)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (fmc_ker_ck)	2	17

21.7.1 External memory interface signals

Table 158, *Table 159* and *Table 160* list the signals that are typically used to interface with NOR Flash memory, SRAM and PSRAM.

Note: The prefix "N" identifies the signals which are active low.

NOR Flash memory, non-multiplexed I/Os

Table 158. Non-multiplexed I/O NOR Flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[31:0]	I/O	Bidirectional data bus
NE[x]	O	Chip Select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR Flash memory, 16-bit multiplexed I/Os

Table 159. 16-bit multiplexed I/O NOR Flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/SRAM, non-multiplexed I/Os

Table 160. Non-multiplexed I/Os PSRAM/SRAM

FMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[31:0]	I/O	Data bidirectional bus
NE[x]	O	Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[3:0]	O	Byte lane output. Byte 0 to Byte 3 control (Upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 161. 16-Bit multiplexed I/O PSRAM

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

21.7.2 Supported memories and transactions

Table 162 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR Flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 162. NOR Flash/PSRAM: Example of supported memories and transactions(1)

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NOR Flash (muxed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32/64	16	Y	
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32/64	16	Y	
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32/64	16	Y	

Table 162. NOR Flash/PSRAM: Example of supported memories and transactions(1)

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
SRAM and ROM	Asynchronous	R	8/16	16	Y	
	Asynchronous	W	8/16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses Use of byte lanes NBL[1:0]
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses Use of byte lanes NBL[1:0]

1. NBL[1:0] are also driven by AXI write strobes.

21.7.3 General timing rules

Signal synchronization is performed as follows:

- All controller output signals change on the rising edge of the fmc_ker_ck clock.
- In Synchronous read and write modes, all output signals change on the rising edge of fmc_ker_ck clock. Whatever the CLKDIV value, all outputs change as follows:
  - NOEL/NWEL/ NEL/NADVL/ NADVH /NBLL/ Address valid outputs change on the falling edge of FMC_CLK clock.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

21.7.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous transactions on static memories (NOR Flash memory, PSRAM, SRAM) are performed as follows:

- Signals are synchronized by the internal clock. This clock is not issued to the memory.
- The FMC always samples the data before de-asserting the Chip Select signal. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the Extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the Extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in Mode1 or Mode2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 95. Mode 1 read access waveforms

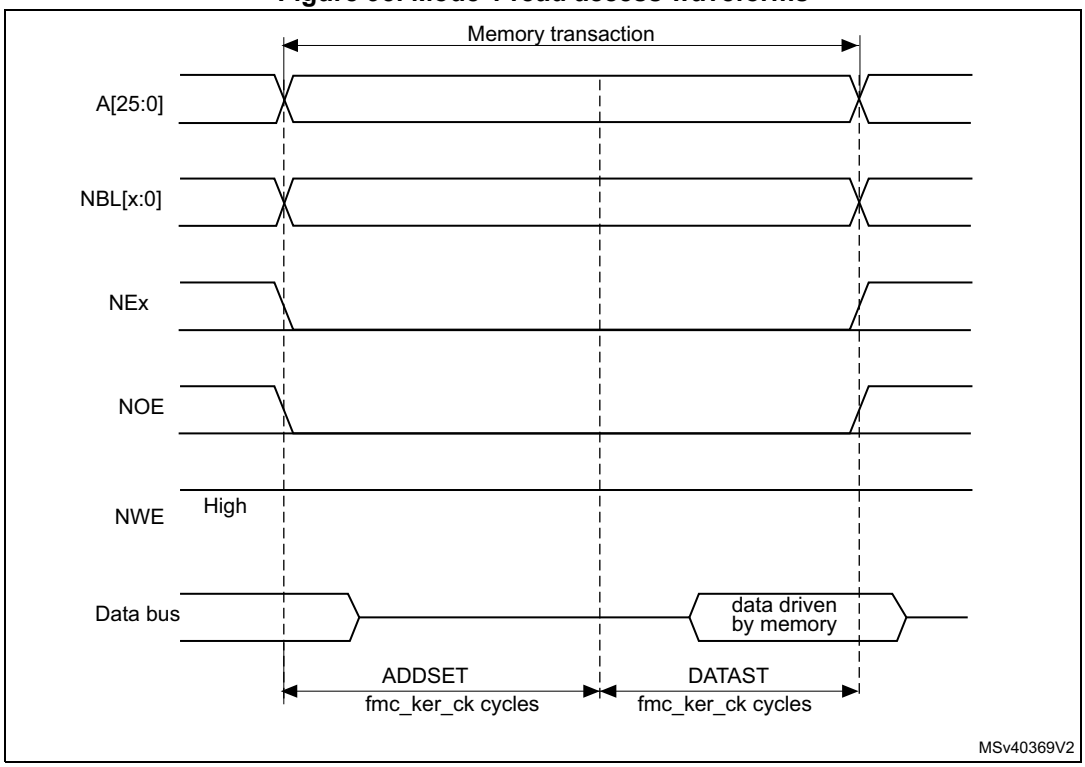
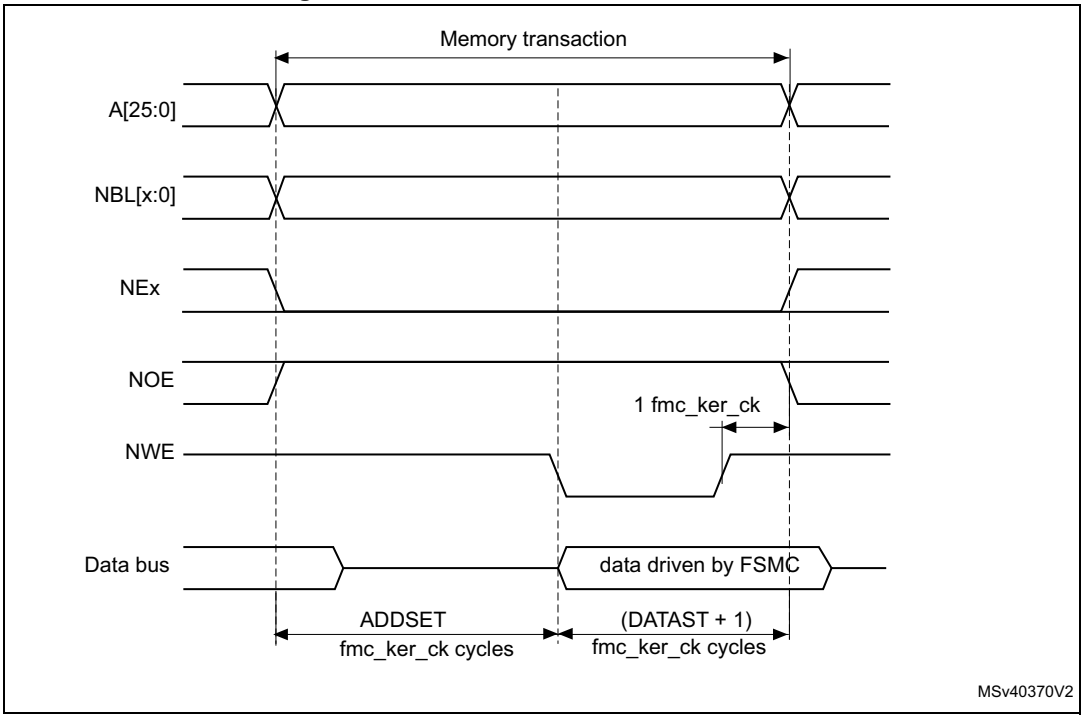


Figure 96. Mode 1 write access waveforms



The fmc_ker_ck cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this fmc_ker_ck cycle, the DATAST value must be greater than zero (DATAST > 0).

Table 163. FMC_BCRx bit fields

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 163. FMC_BCRx bit fields (continued)

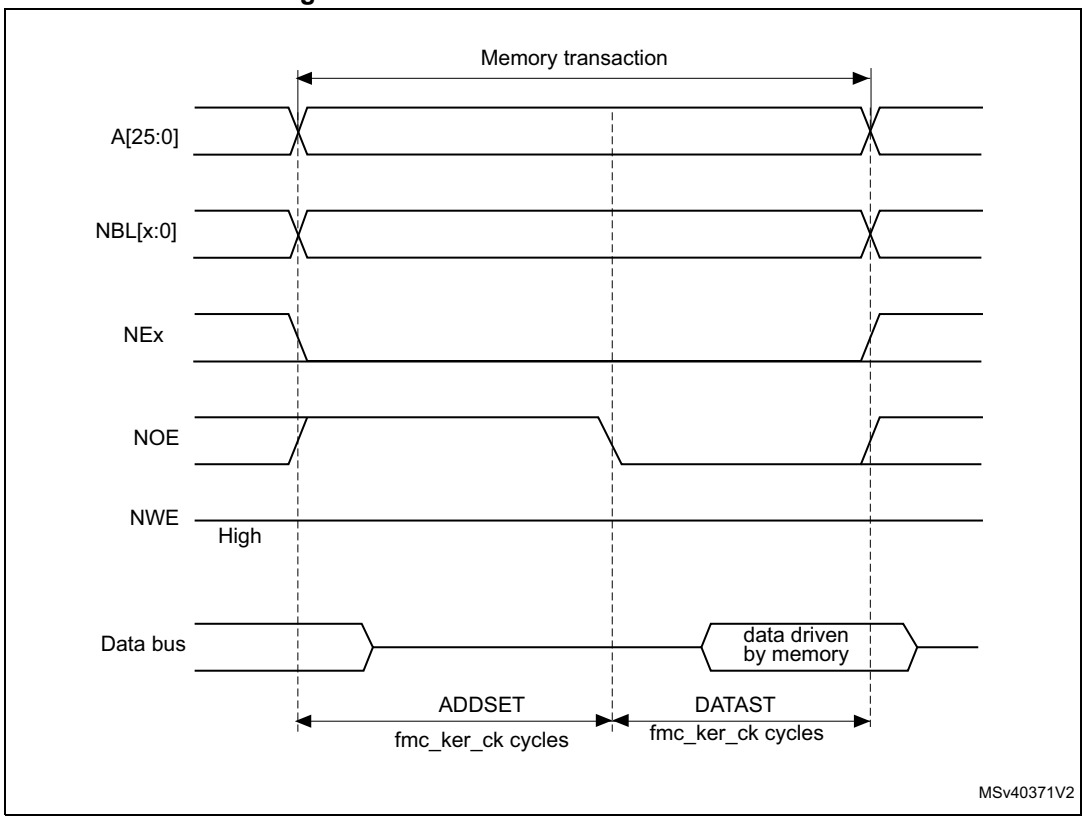
Bit number	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 164. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	Don’t care
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST+1 fmc_ker_ck cycles for write accesses, DATAST fmc_ker_ck cycles for read accesses).
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 0.

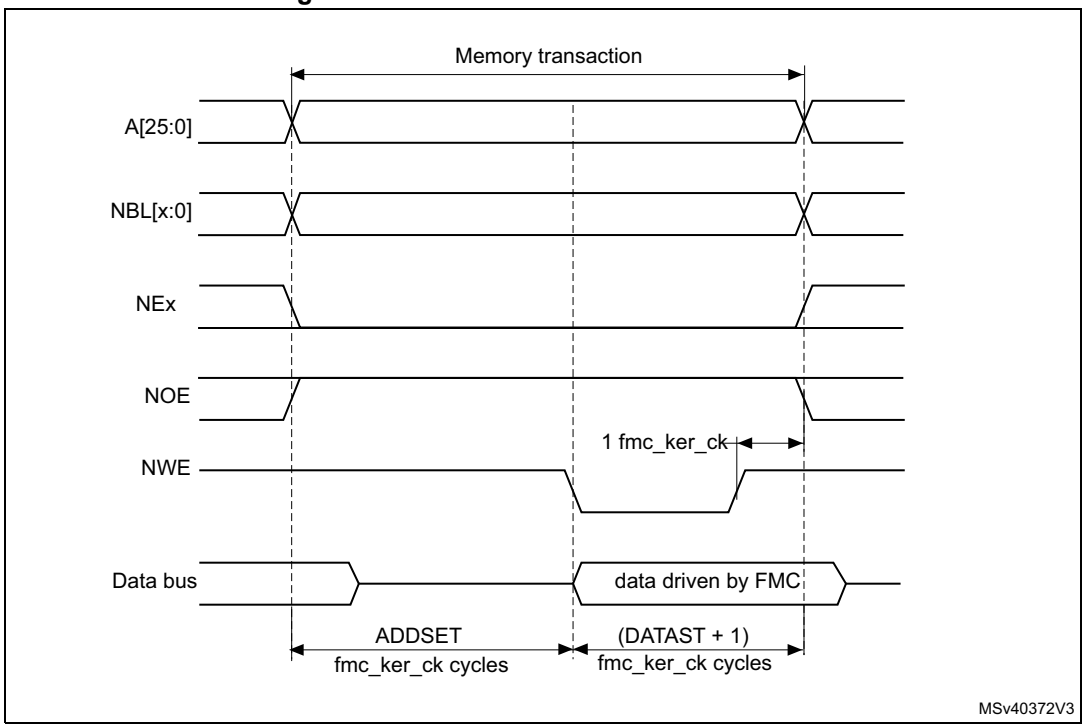
Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 97. Mode A read access waveforms



1. NBL[3:0] are driven low during the read access

Figure 98. Mode A write access waveforms



The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

Table 165. FMC_BCRx bit fields

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 165. FMC_BCRx bit fields (continued)

Bit number	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don't care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 166. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 167. FMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR Flash

Figure 99. Mode 2 and mode B read access waveforms

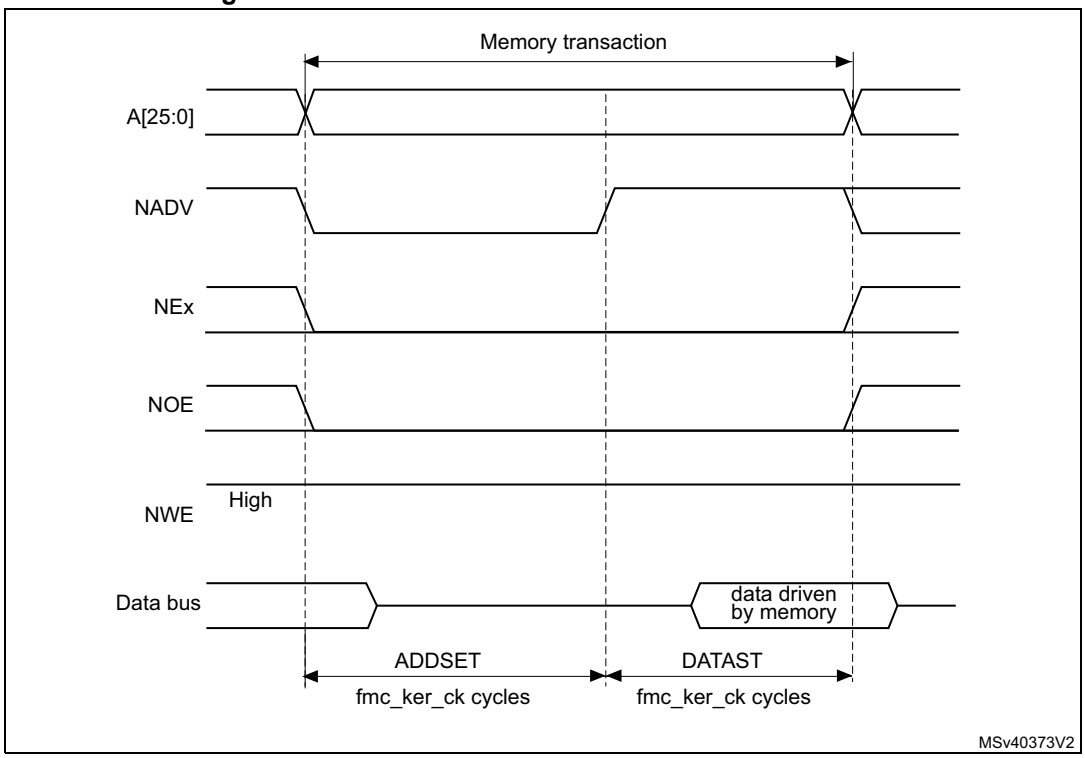


Figure 100. Mode 2 write access waveforms

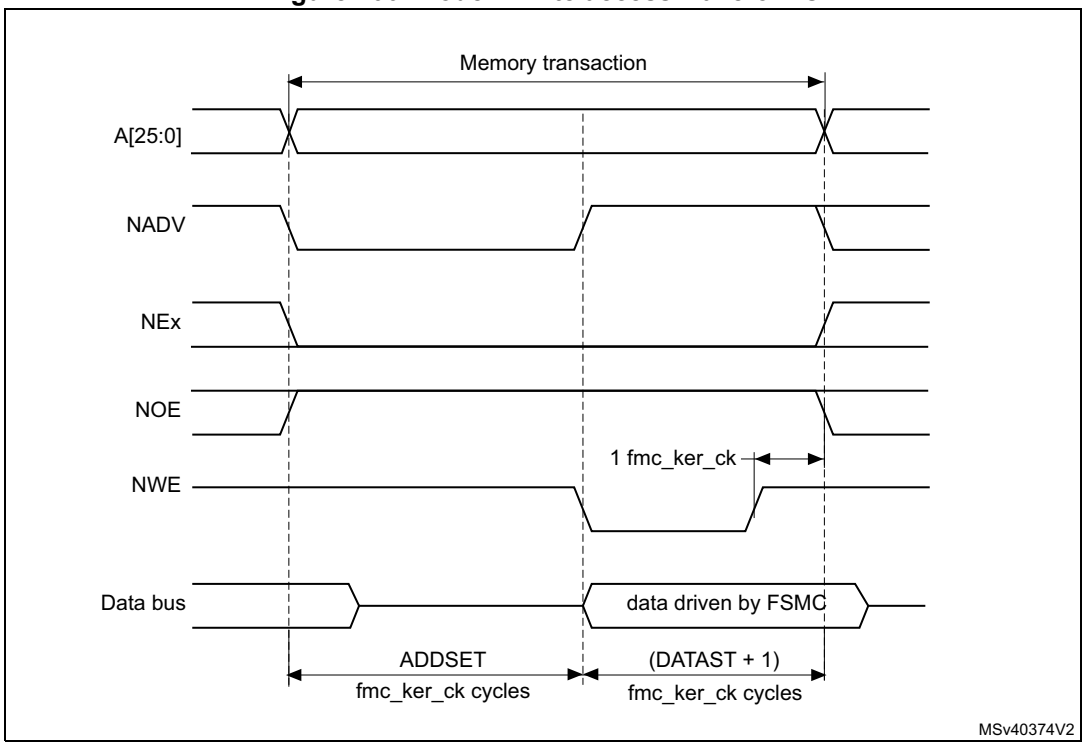
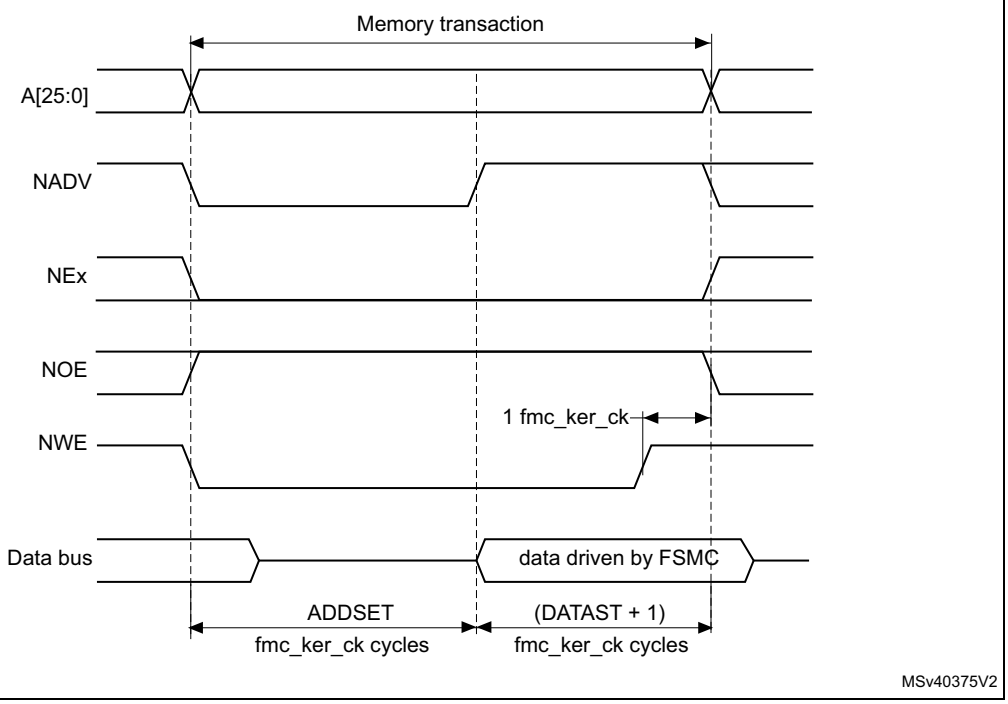


Figure 101. Mode B write access waveforms



The differences with Mode1 are the toggling of NWE and the independent read and write timings when extended mode is set (Mode B).

Table 168. FMC_BCRx bit fields

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 168. FMC_BCRx bit fields (continued)

Bit number	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 169. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 170. FMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the Extended mode is set (mode B), otherwise its content is don't care.

Mode C - NOR Flash - OE toggling

Figure 102. Mode C read access waveforms

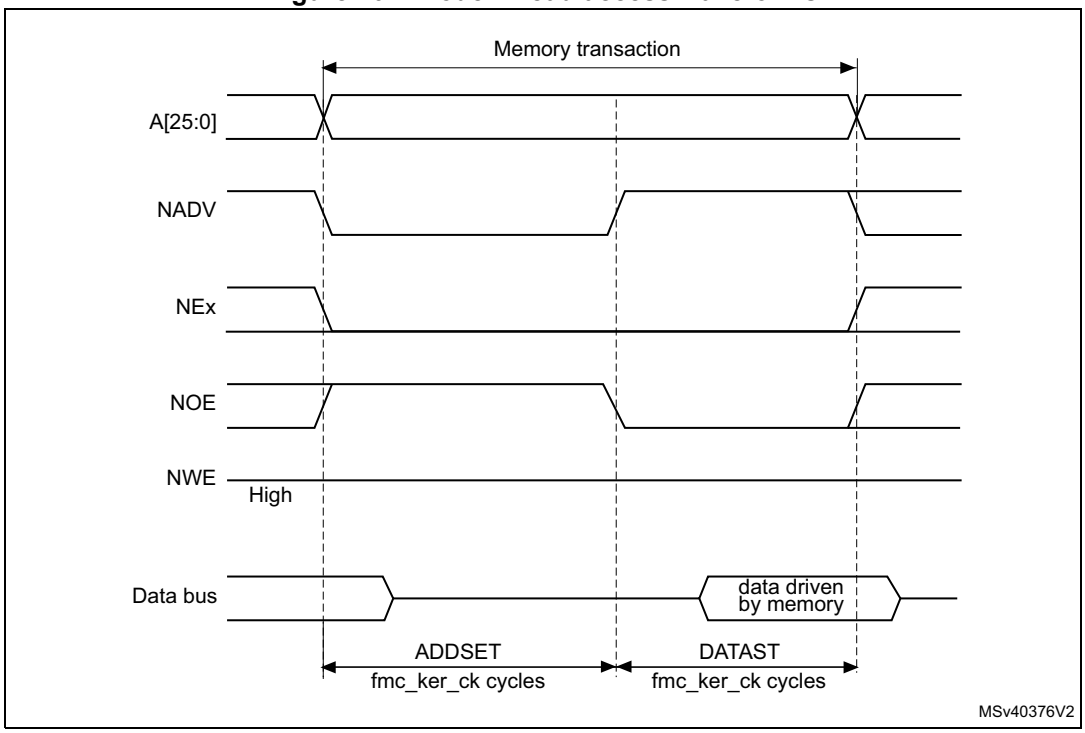
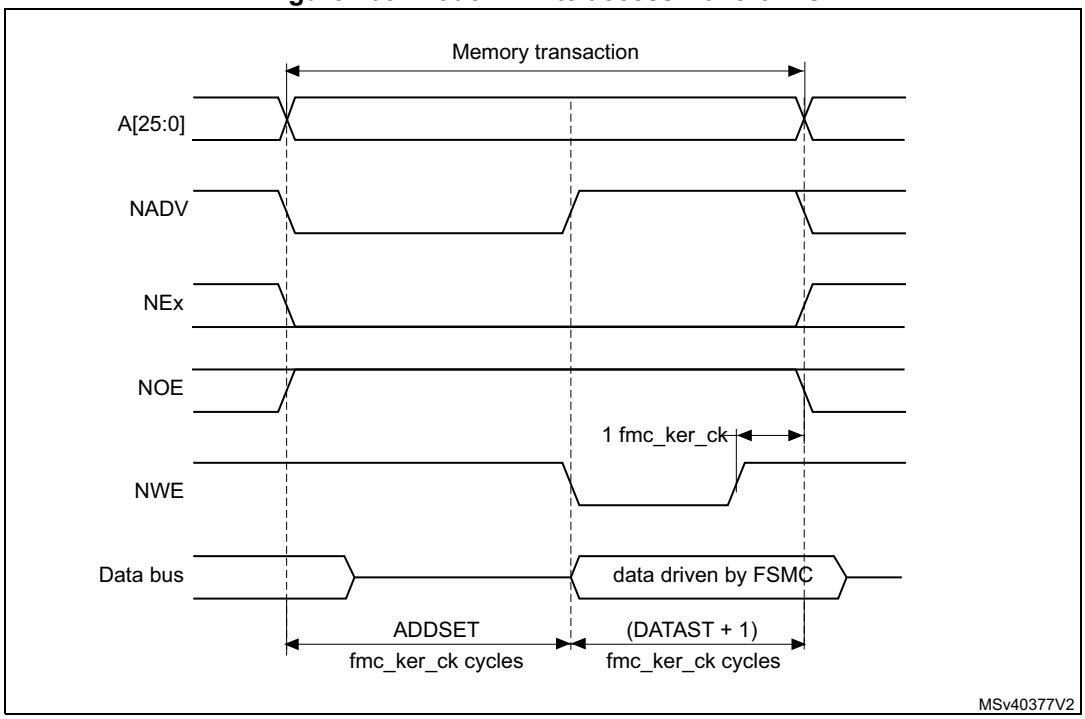


Figure 103. Mode C write access waveforms



The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

Table 171. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x02 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 172. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x2
27:24	DATLAT	0x0
23:20	CLKDIV	0x0
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 173. FMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x2
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 104. Mode D read access waveforms

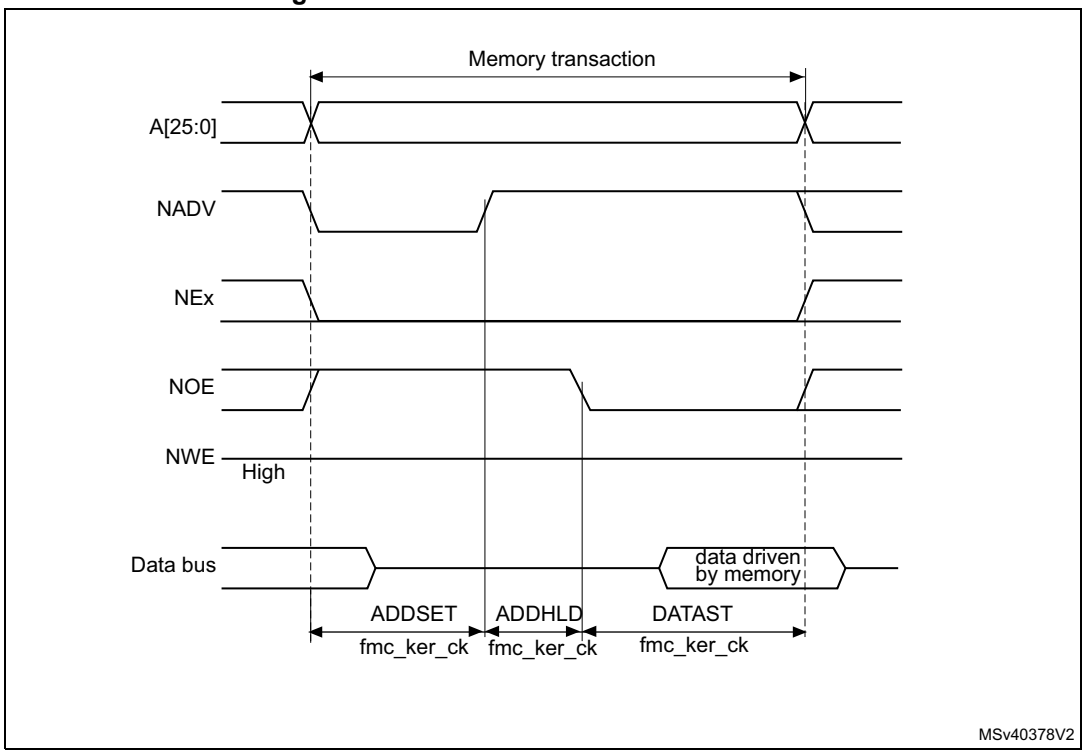
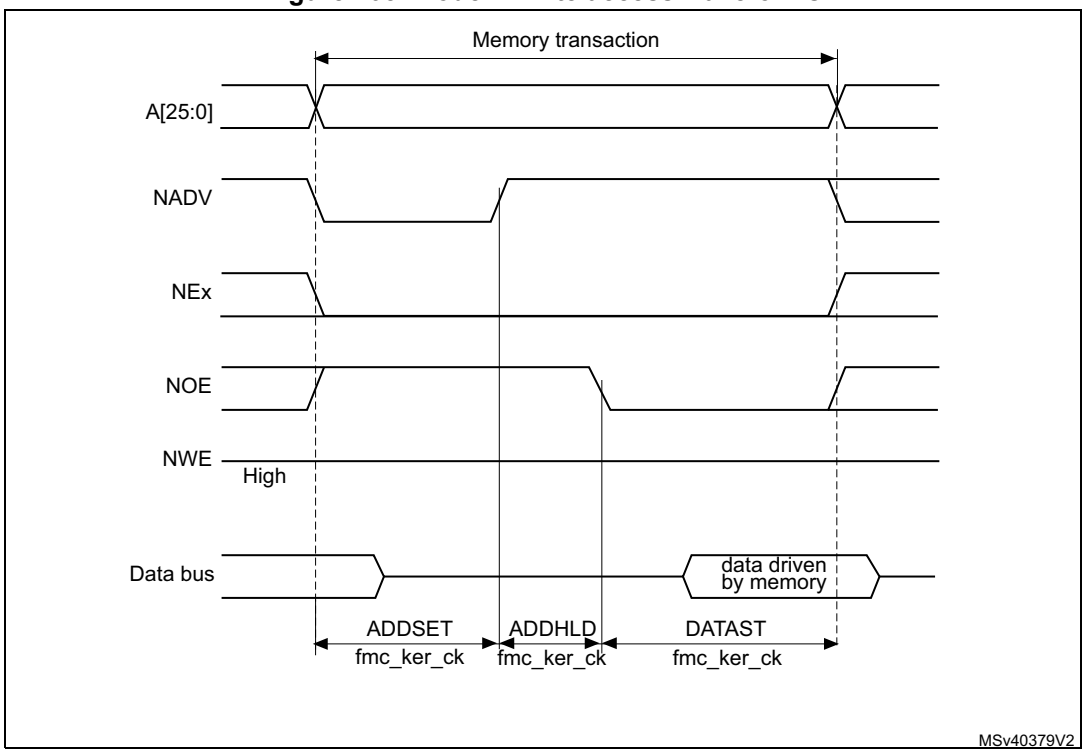


Figure 105. Mode D write access waveforms



The differences with Mode1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 174. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 175. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.

Table 175. FMC_BTRx bit fields (continued)

Bit number	Bit name	Value to set
7:4	ADDHLD	Duration of the middle phase of the read access (ADDHLD fmc_ker_ck cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 1.

Table 176. FMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST + 1 fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Duration of the middle phase of the write access (ADDHLD fmc_ker_ck cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR Flash memory

Figure 106. Muxed read access waveforms

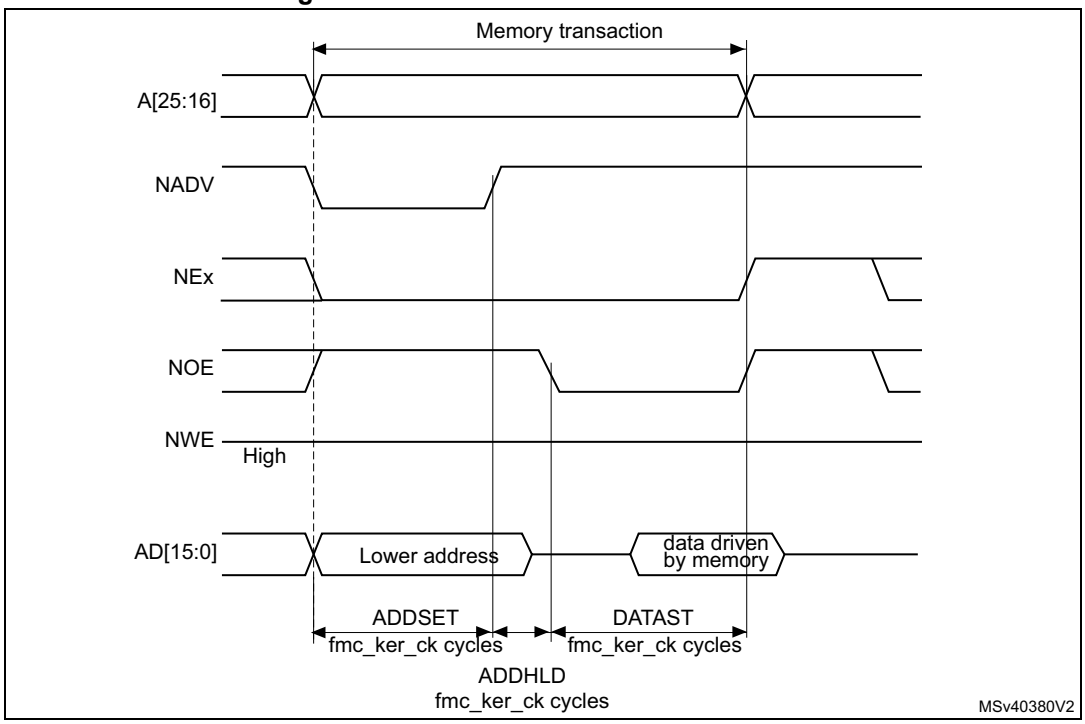
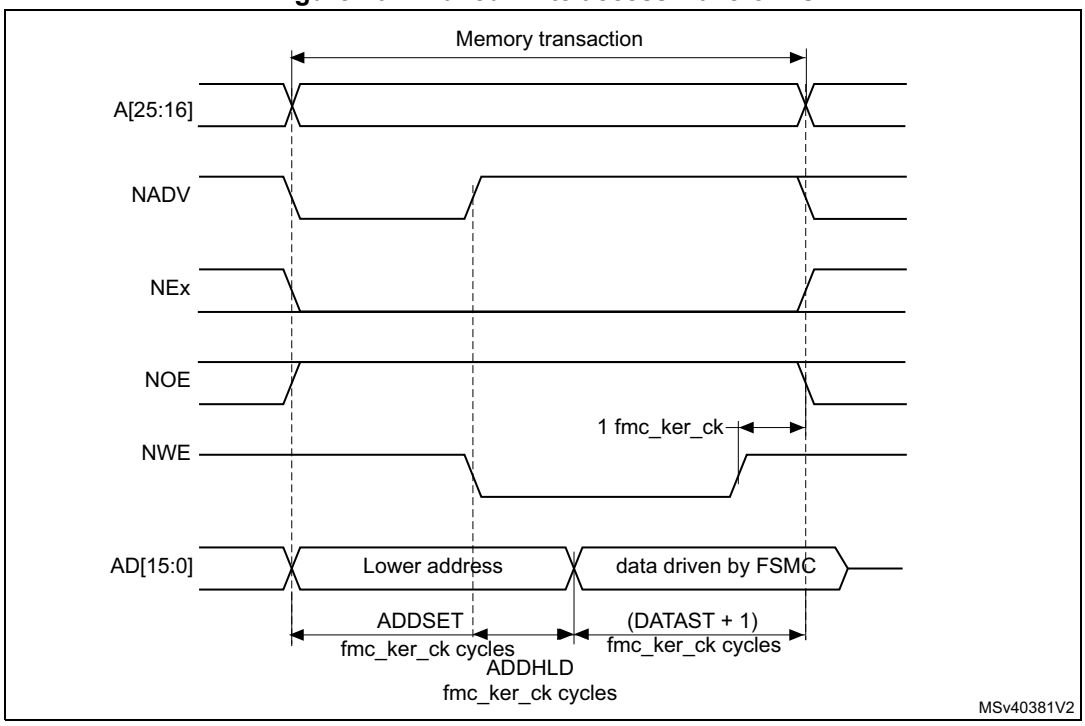


Figure 107. Muxed write access waveforms



The difference with Mode D is the drive of the lower address byte(s) on the data bus.

Table 177. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR Flash memory)
1	MUXEN	0x1
0	MBKEN	0x1

Table 178. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles for read accesses and DATAST+1 fmc_ker_ck cycles for write accesses).

Table 178. FMC_BTRx bit fields (continued)

Bit number	Bit name	Value to set
7:4	ADDHLD	Duration of the middle phase of the access (ADDHLD fmc_ker_ck cycles).
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 fmc_ker_ck cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$DATAST \geq (4 \times FMC_CLK) + max_wait_assertion_time$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   if

$$max_wait_assertion_time > address_phase + hold_phase$$

   then:

$$DATAST \geq (4 \times FMC_CLK) + (max_wait_assertion_time - address_phase - hold_phase$$

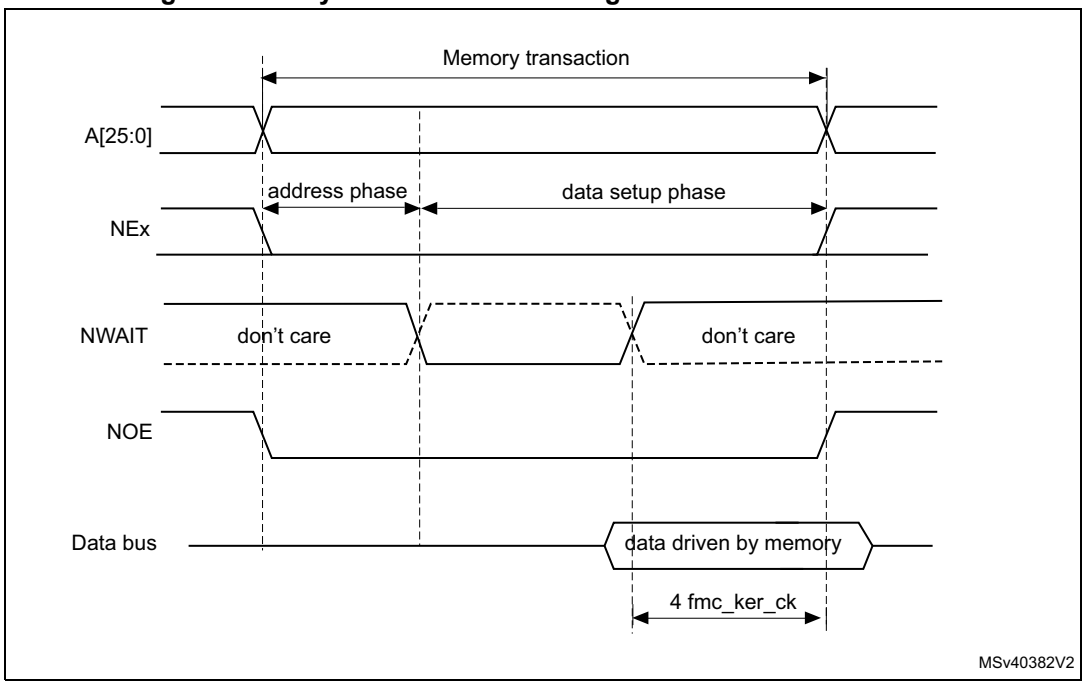
   otherwise

$$DATAST \geq (4 \times FMC_CLK)$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

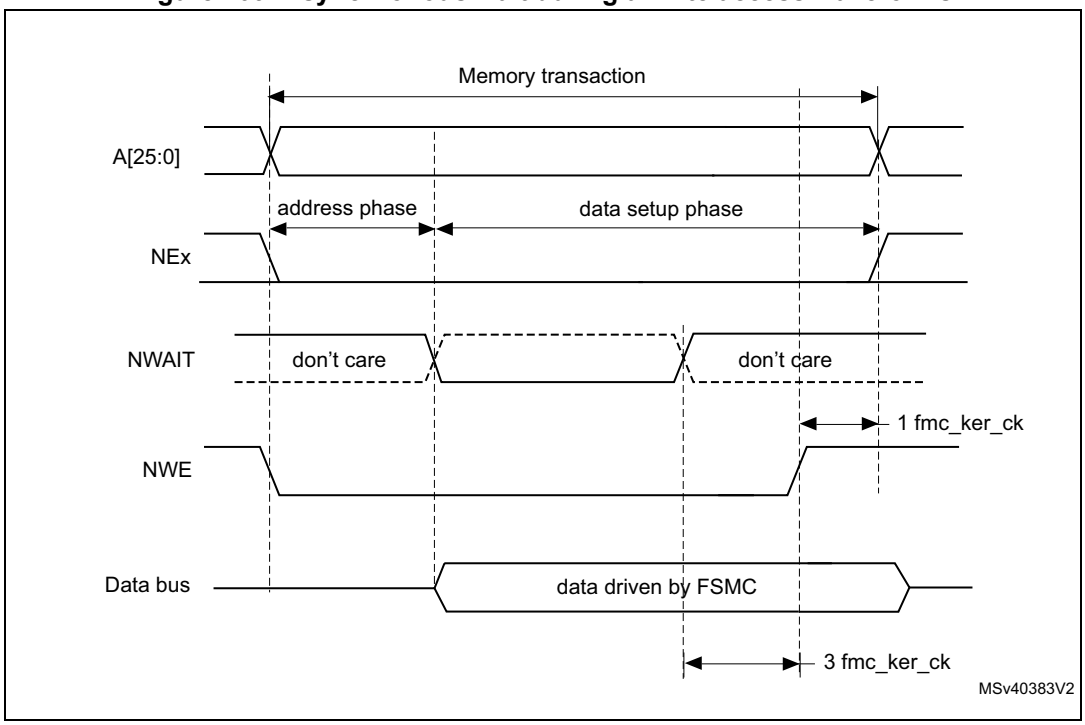
Figure 108 and *Figure 109* show the number of fmc_ker_ck clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 108. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 109. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

21.7.5 Synchronous transactions

The memory clock, FMC_CLK, is a sub-multiple of fmc_ker_ck. It depends on the value of CLKDIV and the MWID/ AXI data size, following the formula given below:

$$\text{FMC_CLK divider ratio} = \max(\text{CLKDIV} + 1, \text{MWID}(\text{AXI data size}))$$

If MWID is 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

If MWID is 32-bit, the FMC_CLK divider ratio depends also on AXI data size.

Example:

- If CLKDIV=1, MWID = 32 bits, AXI data size=8 bits, FMC_CLK=fmc_ker_ck/4.
- If CLKDIV=1, MWID = 16 bits, AXI data size=8 bits, FMC_CLK=fmc_ker_ck/2.

NOR Flash memories specify a minimum time from NADV assertion to FMC_CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

For some PSRAM memories which must be configured to Synchronous mode, during the BCR register writing, the memory attribute space must be configured to device or strongly-ordered. Once PSRAM BCR register is configured, the memory attribute of PSRAM address space can be programmed to cacheable.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR Flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR Flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR Flash latency and the FMC DATLAT parameter can be either:

- NOR Flash latency = (DATLAT + 2) FMC_CLK clock cycles
- or NOR Flash latency = (DATLAT + 3) FMC_CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in Burst mode for synchronous accesses, if for example a single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AXI transfer is 16 bits), or length 2 (if the AXI transfer is 32 bits) and de-assert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Cross boundary page for Cellular RAM 1.5

Cellular RAM 1.5 does not allow burst access to cross the page boundary. The FMC controller allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR Flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) FMC_CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In Burst mode, there are two timing configurations for the NOR Flash NWAIT signal:

- The Flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The Flash memory asserts the NWAIT signal during the wait state

The FMC supports both NOR Flash wait state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).

Figure 110. Wait configuration waveforms

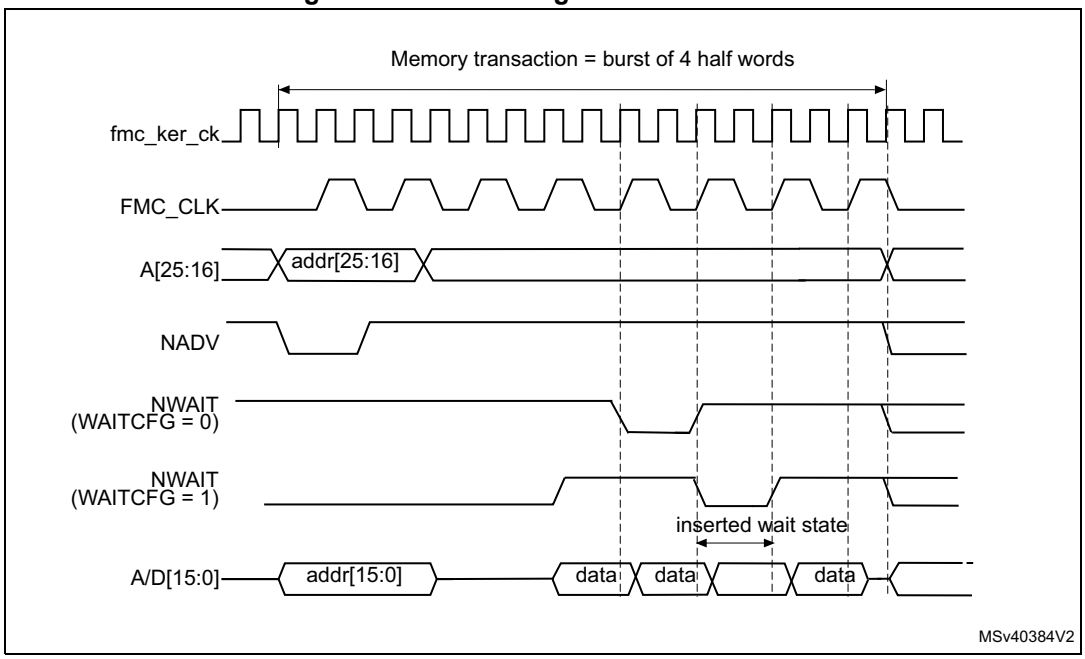
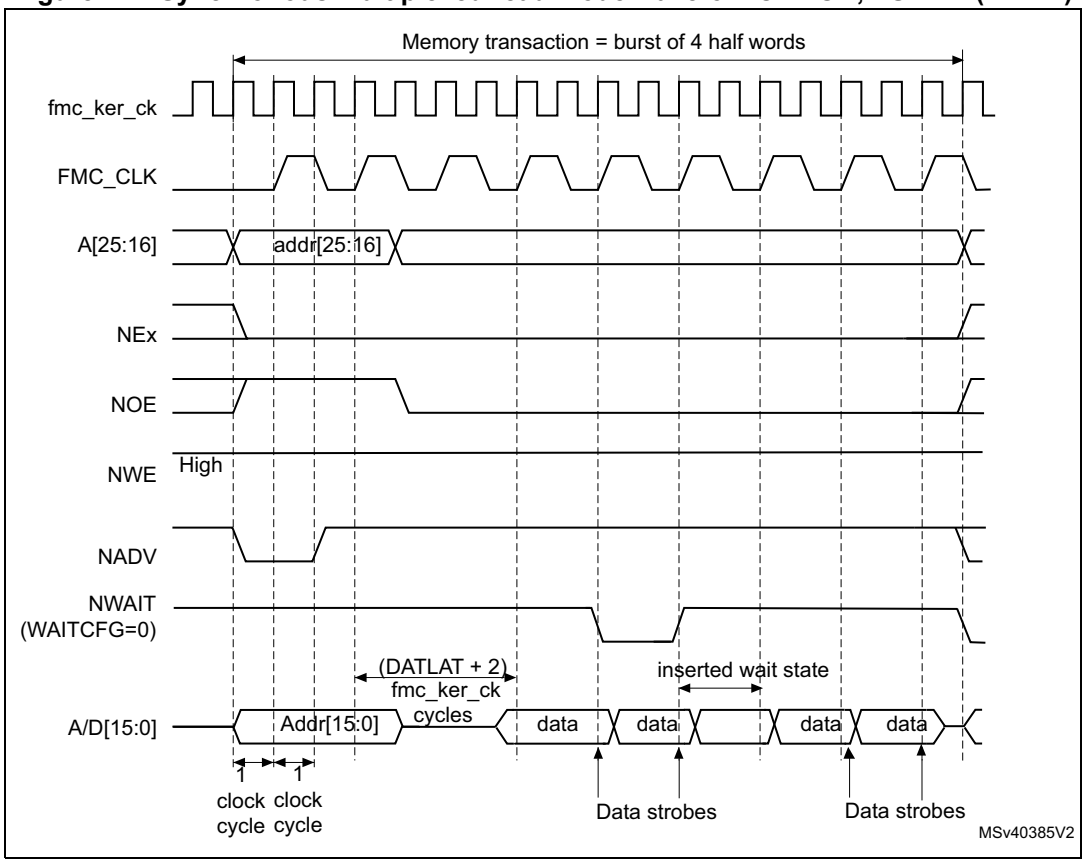


Figure 111. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)



1. Byte lane outputs (NBL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.

Table 179. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31	MC	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Set according to memory support (NOR Flash memory)
5:4	MWID	As needed
3:2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

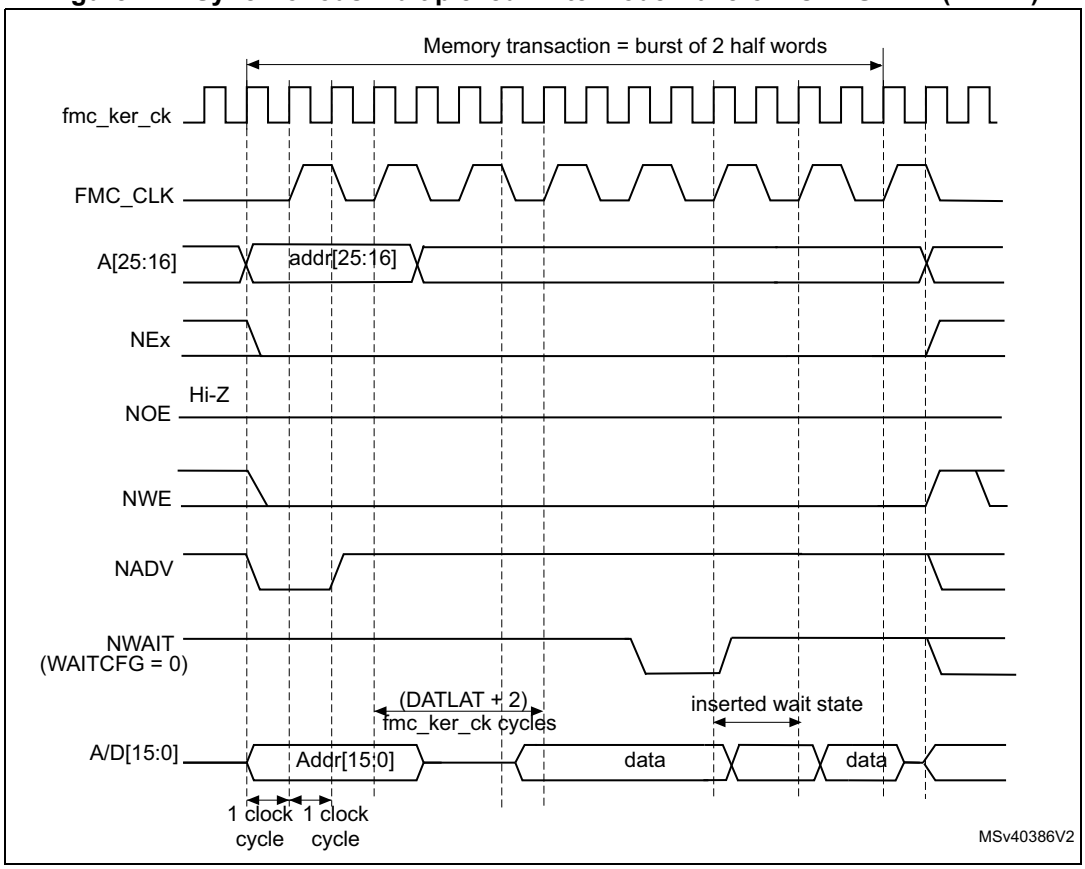
Table 180. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Data latency
27:24	DATLAT	Data latency
23:20	CLKDIV	0x0 to get CLK = fmc_ker_ck (Not supported) 0x1 to get CLK = 2 × fmc_ker_ck ..
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Don’t care

Table 180. FMC_BTRx bit fields (continued)

Bit number	Bit name	Value to set
7:4	ADDHLD	Don't care
3:0	ADDSET	Don't care

Figure 112. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 181. FMC_BCRx bit fields

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read

Table 181. FMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
12	WREN	0x1
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	Reserved	0x1
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 182. FMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Data latency
23:20	CLKDIV	0x0 to get CLK = fmc_ker_ck (not supported) 0x1 to get CLK = 2 × fmc_ker_ck
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Don’t care
7:4	ADDHLD	Don’t care
3:0	ADDSET	Don’t care

21.7.6 NOR/PSRAM controller registers

SRAM/NOR-Flash chip-select control registers for bank x (FMC_BCRx)

Address offset: 8 * (x - 1), (x = 1 to 4)

Reset value: Block 1: 0x0000 30DB

Reset value: Block 2: 0x0000 30D2

Reset value: Block 3: 0x0000 30D2

Reset value: Block 4: 0x0000 30D2

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR Flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FMCEN	Res.	Res.	Res.	Res.	Res.	BMAP[1:0]		Res.	Res.	WFDIS	CCLKEN	CBURSTRW	CPSIZE[2:0]		
rw						rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID		MTYP		MUXEN	MBKEN
rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **FMCEN:** FMC controller Enable

This bit enables/disables the FMC controller.

0: Disable the FMC controller

1: Enable the FMC controller

Note: The FMCEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **BMAP[1:0]:** FMC bank mapping

These bits allow different remap or swap of the FMC NOR/PSRAM and SDRAM banks (refer to *Table 147*).

00: Default mapping (refer to *Figure 94* and *Table 147*).

01: NOR/PSRAM bank and SDRAM bank 1/bank2 are swapped.

10: SDRAM Bank2 remapped on FMC bank2 and still accessible at default mapping

11: Reserved.

Note: The BMAP bits of the FMC_BCR2..4 registers are don't care. It is only enabled through the FMC_BCR1 register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **WFDIS:** Write FIFO Disable

This bit disables the Write FIFO used by the FMC controller.

0: Write FIFO enabled (Default after reset)

1: Write FIFO disabled

Note: The WFDIS bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bit 20 **CCLKEN:** Continuous Clock Enable

This bit enables the FMC_CLK clock output to external memory devices.

0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).

1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The CCLKEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in Synchronous mode to generate the FMC_CLK continuous clock.

If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don't care.

If the Synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 **CBURSTRW:** Write burst enable

For PSRAM (CRAM) operating in Burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.

0: Write operations are always performed in Asynchronous mode

1: Write operations are performed in Synchronous mode.

Bits 18:16 **CPSIZE[2**:0]: CRAM Page Size

These are used for Cellular RAM 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FMC controller splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).

000: No burst split when crossing page boundary (default after reset).

001: 128 bytes

010: 256 bytes

100: 1024 bytes

Other configuration: reserved.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)

1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 **EXTMOD**: Extended mode enable.

This bit enables the FMC to program the write timings for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

0: values inside FMC_BWTR register are not taken into account (default after reset)

1: values inside FMC_BWTR register are taken into account

Note: When the Extended mode is disabled, the FMC can operate in Mode1 or Mode2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP =0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).*

Bit 13 **WAITE**N: Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in Synchronous mode.

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)

1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset)

Bit 12 **WREN**: Write enable bit

This bit indicates whether write operations are enabled/disabled in the bank by the FMC:

0: Write operations are disabled in the bank by the FMC, an AXI slave error is reported,

1: Write operations are enabled for the bank by the FMC (default after reset).

Bit 11 **WAITCF**G: Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in Synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:

0: NWAIT signal is active one data cycle before wait state (default after reset),

1: NWAIT signal is active during wait state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPO**L: Wait signal polarity bit

This bit defines the polarity of the wait signal from memory used for either in Synchronous or Asynchronous mode:

0: NWAIT active low (default after reset),

1: NWAIT active high.

Bit 8 **BURS**TEN: Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in Burst mode:

0: Burst mode disabled (default after reset). Read accesses are performed in Asynchronous mode.

1: Burst mode enable. Read accesses are performed in Synchronous mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCE**N: Flash access enable

This bit enables NOR Flash memory access operations.

0: Corresponding NOR Flash memory access is disabled

1: Corresponding NOR Flash memory access is enabled (default after reset)

Bits 5:4 **MWID[1:0]:** Memory data bus width

Defines the external memory device width, valid for all type of memories.

00: 8 bits

01: 16 bits (default after reset)

10: 32 bits

11: reserved

Bits 3:2 **MTYP[1:0]:** Memory type

These bits define the type of external memory attached to the corresponding memory bank:

00: SRAM (default after reset for Bank 2...4)

01: PSRAM (CRAM)

10: NOR Flash/OneNAND Flash (default after reset for Bank 1)

11: reserved

Bit 1 **MUXEN:** Address/data multiplexing enable bit

When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:

0: Address/Data non-multiplexed

1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory bank enable bit

This bit enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AXI bus.

0: Corresponding memory bank is disabled

1: Corresponding memory bank is enabled

SRAM/NOR-Flash chip-select timing registers for bank x (FMC_BTRx)

Address offset: 0x04 + 8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR Flash memories.If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ACCMOD		DATLAT				CLKDIV				BUSTURN			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST								ADDHLD				ADDSET			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ACCMOD[1:0]:** Access mode

These bits specify the Asynchronous access modes as shown in the timing diagrams. They are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:24 **DATLAT[3:0]:** (see note below bit descriptions): Data latency for synchronous memory

For synchronous access with read/write Burst mode enabled (BURSTEN / CBURSTRW bits set), these bits define the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:

This timing parameter is not expressed in fmc_ker_ck periods, but in FMC_CLK periods.

For asynchronous access, this value is don't care.

0000: Data latency of 2 FMC_CLK clock cycles for first burst access

1111: Data latency of 17 FMC_CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)

These bits define the period of FMC_CLK clock output signal, expressed in number of fmc_ker_ck cycles:

0000: Reserved

0001: FMC_CLK period = 2 × fmc_ker_ck periods

0010: FMC_CLK period = 3 × fmc_ker_ck periods

1111: FMC_CLK period = 16 × fmc_ker_ck periods (default value after reset)

In asynchronous NOR Flash, SRAM or PSRAM accesses, this value is don't care.

Note: Refer to Section 21.7.5: Synchronous transactions for FMC_CLK divider ratio formula)

Bits 19:16 **BUSTURN:** Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write-to-read (and read-to-write) transaction. This delay allows to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (t_{EHQZ}). The programmed bus turnaround delay is inserted between an asynchronous read (muxed or mode D) or write transaction and any other asynchronous /synchronous read or write to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different except for muxed or mode D.

In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed

as follows:

- The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes.
- There is a bus turnaround delay of 1 FMC clock cycle between:
  - –Two consecutive asynchronous read transfers to the same static memory bank except for muxed and D modes.
  - –An asynchronous read to an asynchronous or synchronous write to any static bank or dynamic bank except for muxed and D modes.
  - –An asynchronous (modes 1, 2, A, B or C) read and a read from another static bank.
- There is a bus turnaround delay of 2 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to the same bank.
  - –A synchronous write (burst or single) access and an asynchronous write or read transfer to or from static memory bank (the bank can be the same or different for the case of read.
  - –Two consecutive synchronous reads (burst or single) followed by any synchronous/asynchronous read or write from/to another static memory bank.
- There is a bus turnaround delay of 3 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to different static bank.
  - –A synchronous write (burst or single) access and a synchronous read from the same or a different bank.

0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 15 x fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 95* to *Figure 107*), used in asynchronous accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × fmc_ker_ck clock cycles

0000 0010: DATAST phase duration = 2 × fmc_ker_ck clock cycles

...

1111 1111: DATAST phase duration = 255 × fmc_ker_ck clock cycles (default value after reset)

For each memory type and access mode data-phase duration, please refer to the respective figure (*Figure 95* to *Figure 107*).

Example: Mode1, write access, DATAST = 1: Data-phase duration = DATAST+1 = 1 x fmc_ker_ck clock cycles.

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 95* to *Figure 107*), used in mode D or multiplexed accesses:
0000: Reserved
0001: ADDHLD phase duration =1 × fmc_ker_ck clock cycle
0010: ADDHLD phase duration = 2 × fmc_ker_ck clock cycle
...
1111: ADDHLD phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)
For each access mode address-hold phase duration, please refer to the respective figure (*Figure 95* to *Figure 107*).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 **ADDSET:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 95* to *Figure 107*), used in SRAMs, ROMs and asynchronous NOR Flash:
0000: ADDSET phase duration = 0 × fmc_ker_ck clock cycle
...
1111: ADDSET phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)
For each access mode address setup phase duration, please refer to the respective figure (refer to *Figure 95* to *Figure 107*).

Note: In synchronous accesses, this value is don't care.
In Muxed mode or Mode D, the minimum value for ADDSET is 1.
In mode 1 and PSRAM memory, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to extend the latency as needed.
On PSRAMs (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).

SRAM/NOR-Flash write timing registers for bank x (FMC_BWTRx)

Address offset: 0x104 + 8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRAMs and NOR Flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN			
		rw	rw									rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST								ADDHLD				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ACCMOD:** Access mode.

These bits specify the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (t_{EHEL} from ENx high to ENx low):

(BUSTRUN + 1) fmc_ker_ck period ≥ $t_{EHELmin}$.

The programmed bus turnaround delay is inserted between an asynchronous write transfer and any other asynchronous /synchronous read or write transfer to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different expect for muxed or mode D.

In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed as follows:

- The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes.
- There is a bus turnaround delay of 2 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to the same bank.
  - –A synchronous write (burst or single) transfer and an asynchronous write or read transfer to or from static memory bank.
- There is a bus turnaround delay of 3 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to different static bank.
  - –A synchronous write (burst or single) transfer and a synchronous read from the same or a different bank.

0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 15 fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to *Figure 95* to *Figure 107*), used in asynchronous SRAM, PSRAM and NOR Flash memory accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × fmc_ker_ck clock cycles

0000 0010: DATAST phase duration = 2 × fmc_ker_ck clock cycles

...

1111 1111: DATAST phase duration = 255 × fmc_ker_ck clock cycles (default value after reset)

Bits 7:4 **ADDHLD:** Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 95* to *Figure 107*), used in asynchronous multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × fmc_ker_ck clock cycle

0010: ADDHLD phase duration = 2 × fmc_ker_ck clock cycle

...

1111: ADDHLD phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is not used, the address hold phase is always 1 Flash clock period duration.

Bits 3:0 **ADDSET:** Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in fmc_ker_ck cycles (refer to *Figure 95* to *Figure 107*), used in asynchronous accesses:

0000: ADDSET phase duration = 0 × fmc_ker_ck clock cycle

...

1111: ADDSET phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, the address setup phase is always 1 Flash clock period duration. In muxed mode, the minimum ADDSET value is 1.

21.8 NAND Flash controller

The FMC generates the appropriate signal timings to drive 8- and 16-bit NAND Flash memories.

The NAND bank is configured through dedicated registers (*Section 21.8.7*). The programmable memory parameters include access timings (shown in *Table 183*) and ECC configuration.

Table 183. Programmable NAND Flash access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (fmc_ker_ck) required to set up the address before the command assertion	Read/Write	AHB clock cycle (fmc_ker_ck)	1	255
Memory wait	Minimum duration (in fmc_ker_ck clock cycles) of the command assertion	Read/Write	AHB clock cycle (fmc_ker_ck)	2	255
Memory hold	Number of clock cycles (fmc_ker_ck) during which the address must be held (as well as the data if a write access is performed) after the command de-assertion	Read/Write	AHB clock cycle (fmc_ker_ck)	1	254
Memory databus high-Z	Number of clock cycles (fmc_ker_ck) during which the data bus is kept in high-Z state after a write access has started	Write	AHB clock cycle (fmc_ker_ck)	0	254

21.8.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND Flash memories.

Note: The prefix “N” identifies the signals which are active low.

8-bit NAND Flash memory

Table 184. 8-bit NAND Flash memory

FMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCE	O	Chip Select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND Flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND Flash memory

Table 185. 16-bit NAND Flash memory

FMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE	O	Chip Select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND Flash ready/busy input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

21.8.2 NAND Flash supported memories and transactions

Table 186 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash controller are shown in gray.

Table 186. Supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	-
	Asynchronous	W	8	8	Y	-
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses
	Asynchronous	R	32	8	Y	Split into 8 FMC accesses
	Asynchronous	W	32	8	Y	Split into 8 FMC accesses

Table 186. Supported memories and transactions (continued)

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 16-bit	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	32	16	Y	Split into 4 FMC accesses
	Asynchronous	W	32	16	Y	Split into 4 FMC accesses

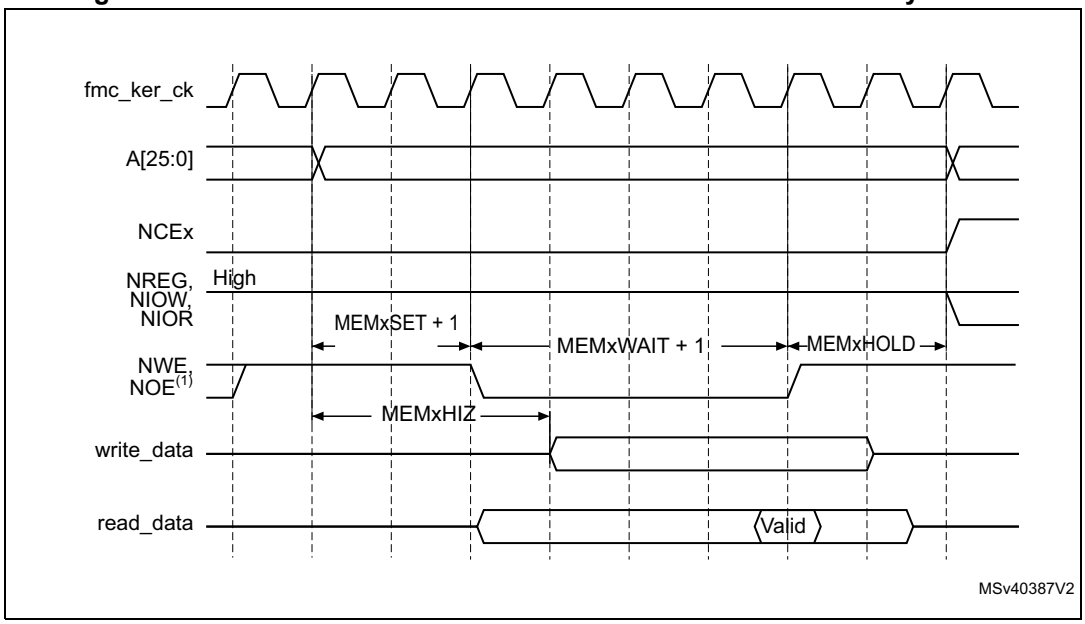
21.8.3 Timing diagrams for NAND Flash memories

The NAND Flash memory bank is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_ECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

Each timing configuration register contains three parameters used to define the number of fmc_ker_ck cycles for the three phases of any NAND Flash access, plus one parameter that defines the timing to start driving the data bus when a write access is performed. *Figure 113* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 113. NAND Flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. For write accesses, the hold phase delay is (MEMHOLD) fmc_ker_ck cycles and for read access is (MEMHOLD + 1) fmc_ker_ck cycles.

21.8.4 NAND Flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash memory device are driven by address signals from the FMC controller. This means that to send a command or an address to the NAND Flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND Flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 21.8.5: NAND Flash prewait feature*) registers according to the characteristics of the NAND Flash memory (PWID bits for the data bus width of the NAND Flash memory, PWAITEN = 0 or 1 as needed, see *Section 21.6.2: NAND Flash memory address mapping* for timing configuration).
2. The CPU performs a byte write to the common memory space, with data byte equal to one Flash command byte (for example 0x00 for Samsung NAND Flash devices). The LE input of the NAND Flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND Flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND Flash memories) in the common memory or attribute space. The ALE input of the NAND Flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used

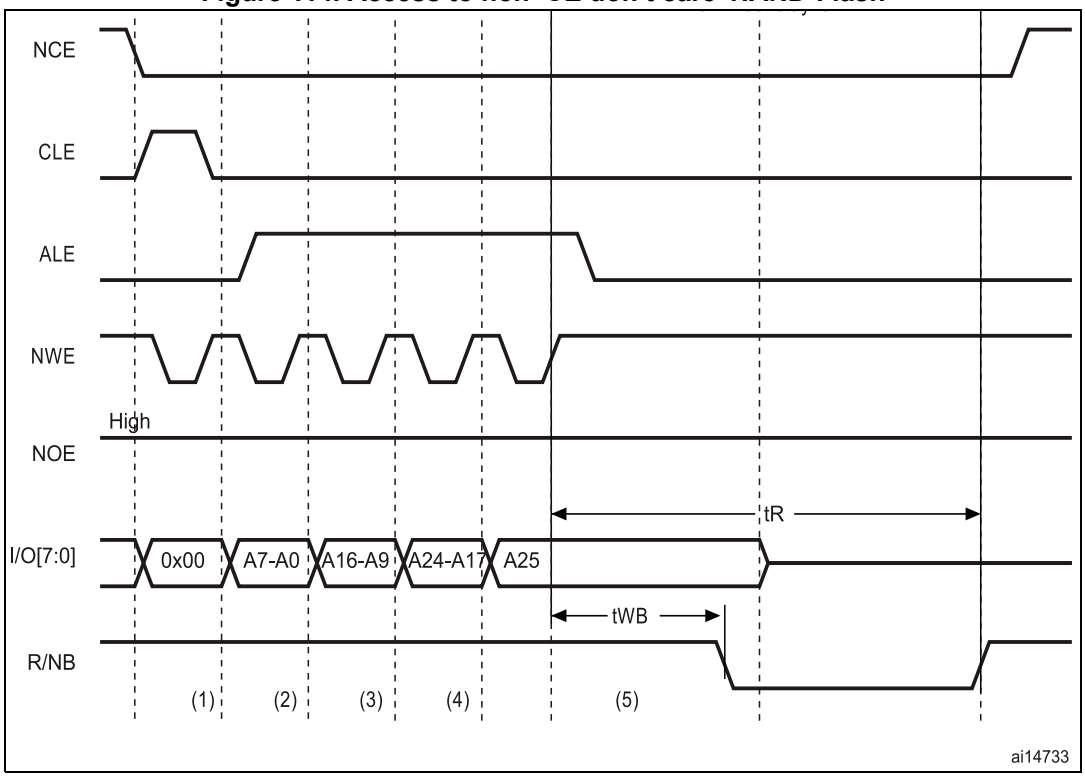
to implement the prewait functionality needed by some NAND Flash memories (see details in *Section 21.8.5: NAND Flash prewait feature*).

4. The controller waits for the NAND Flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).
5. The CPU can then perform byte read operations from the common memory space to read the NAND Flash page (data field + Spare field) byte by byte.
6. The next NAND Flash page can be read without any CPU command or address write operation. This can be done in three different ways:
   – by simply performing the operation described in step 5
   – a new random address can be accessed by restarting the operation at step 3
   – a new command can be sent to the NAND Flash device by restarting at step 2

21.8.5 NAND Flash prewait feature

Some NAND Flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see *Figure 114*).

Figure 114. Access to non 'CE don't care' NAND-Flash



1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7~A0 at address 0x7002 0000.
3. CPU wrote byte A16~A9 at address 0x7002 0000.
4. CPU wrote byte A24~A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x8802 0000: FMC performs a write access using FMC_PATT2 timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × fmc_ker_ck = 112 ns > t_{WB} max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND Flash memories where NCE is not don't care).

When this function is required, it can be performed by programming the MEMHOLD value to meet the t_{WB} timing. However, any CPU read access to NAND Flash memory has a hold delay of (MEMHOLD + 1) fmc_ker_ck cycles, and any CPU write access has a hold delay of (MEMHOLD) fmc_ker_ck cycles that is inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t_{WB} timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND Flash read and write accesses, except when writing the last address byte to the NAND Flash device, where the CPU must write to the attribute memory space.

21.8.6 Computation of the error correction code (ECC) in NAND Flash memory

The FMC controller includes an error correction code computation hardware block. It reduces the host CPU workload when processing the ECC by software.The ECC block is associated with NAND bank.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1 024, 2 048, 4 096 or 8 192 bytes read or written from/to the NAND Flash memory. It is based on the BCH8 coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND Flash data bus and read/write signals (NCE and NWE) each time the NAND Flash memory bank is active.

The ECC operates as follows:

- When accessing NAND Flash bank, the data present on the D[15:0] bus is latched and used for ECC computation.
- When accessing any other address in NAND Flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND Flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND Flash memory by the host CPU, the FMC_ECCR registers must be read to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to one in the FMC_PCR registers.

Execute below the sequence to perform an ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND Flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Wait until the ECC code is ready (FIFO empty).
4. Read the ECC value available in the FMC_ECCR register and store it in a variable.
5. Clear the ECCEN bit and then enable it in the FMC_PCR register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
6. Read the new ECC value available in the FMC_ECCR register.
7. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.

21.8.7 NAND Flash controller registers

NAND Flash control registers (FMC_PCR)

Address offset: 0x80

Reset value: 0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS			TAR3
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR2	TAR1	TAR0	TCLR				Res.	Res.	ECCEN	PWID		Res.	PBKEN	PWAITEN	Res.
rw	rw	rw	rw	rw	rw	rw			rw	rw	rw		rw	rw	

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:17 **ECCPS:** ECC page size.
These bits define the page size for the extended ECC:
000: 256 bytes
001: 512 bytes
010: 1024 bytes
011: 2048 bytes
100: 4096 bytes
101: 8192 bytes

Bits 16:13 **TAR:** ALE to RE delay.
These bits set time from ALE low to RE low in number of fmc_ker_ck clock cycles.
Time is: t_ar = (TAR + SET + 2) × $t_{fmc_ker_ck}$ where $t_{fmc_ker_ck}$ is the FMC clock period
0000: 1 x fmc_ker_ck cycle (default)
1111: 16 x fmc_ker_ck cycles
Note: Set is MEMSET or ATTSET according to the addressed space.

Bits 12:9 **TCLR:** CLE to RE delay.
These bits set time from CLE low to RE low in number of fmc_ker_ck clock cycles. The time is give by the following formula:
t_clr = (TCLR + SET + 2) × $t_{fmc_ker_ck}$ where $t_{fmc_ker_ck}$ is the fmc_ker_ck clock period
0000: 1 x fmc_ker_ck cycle (default)
1111: 16 x fmc_ker_ck cycles
Note: Set is MEMSET or ATTSET according to the addressed space.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **ECCEN:** ECC computation logic enable bit
0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.

Bits 5:4 **PWID:** Data bus width.
These bits define the external memory device width.
00: 8 bits
01: 16 bits (default after reset).
10: reserved.
11: reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PBKEN:** NAND Flash memory bank enable bit.
This bit enables the memory bank. Accessing a disabled memory bank causes an ERROR on AXI bus
0: Corresponding memory bank is disabled (default after reset)
1: Corresponding memory bank is enabled

Bit 1 **PWAITEN:** Wait feature enable bit.
This bit enables the Wait feature for the NAND Flash memory bank:
0: disabled
1: enabled

Bit 0 Reserved, must be kept at reset value.

FIFO status and interrupt register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data.

This is used to quickly write to the FIFO and free the AXI bus for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
									r	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **FEMPT:** FIFO empty.

Read-only bit that provides the status of the FIFO
0: FIFO not empty
1: FIFO empty

Bit 5 **IFEN:** Interrupt falling edge detection enable bit

0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 4 **ILEN:** Interrupt high-level detection enable bit

0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 3 **IREN:** Interrupt rising edge detection enable bit

0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

Bit 2 **IFS:** Interrupt falling edge status

The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred

Note: If this bit is written by software to 1 it will be set.

Bit 1 **ILS:** Interrupt high-level status

The flag is set by hardware and reset by software.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IRS:** Interrupt rising edge status

The flag is set by hardware and reset by software.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

Note: If this bit is written by software to 1 it will be set.

Common memory space timing register (FMC_PMEM)

Address offset: Address: 0x88

Reset value: 0xFCFC FCFC

The FMC_PMEM read/write register contains the timing information for NAND Flash memory bank. This information is used to access either the common memory space of the NAND Flash for command, address write access and data read/write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEMHIZ								MEMHOLD							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMWAIT								MEMSET							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **MEMHIZ:** Common memory x data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept Hi-Z after the start of a NAND Flash write access to common memory space. This is only valid for write transactions:
0000 0000: 0 x fmc_ker_ck cycle
1111 1110: 254 x fmc_ker_ck cycles
1111 1111: reserved.

Bits 23:16 **MEMHOLD:** Common memory hold time

These bits define the number of fmc_ker_ck clock cycles for write accesses and fmc_ker_ck+1 clock cycles for read accesses during which the address is held (and data for write accesses) after the command is de-asserted (NWE, NOE), for NAND Flash read or write access to common memory space:
0000 0000: reserved.
0000 0001: 1 fmc_ker_ck cycle for write access / 3 fmc_ker_ck cycle for read access
1111 1110: 254 fmc_ker_ck cycles for write access / 257 fmc_ker_ck cycles for read access
1111 1111: reserved.

Bits 15:8 **MEMWAIT:** Common memory wait time

These bits define the minimum number of fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), for NAND Flash read or write access to common memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:
0000 0000: reserved
0000 0001: x fmc_ker_ck cycles (+ wait cycle introduced by de-asserting NWAIT)
1111 1110: 255 x fmc_ker_ck cycles (+ wait cycle introduced by de-asserting NWAIT)
1111 1111: reserved.

Bits 7:0 **MEMSET:** Common memory x setup time

These bits define the number of fmc_ker_ck (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for NAND Flash read or write access to common memory space:
0000 0000: fmc_ker_ck cycles
1111 1110: 255 x fmc_ker_ck cycles
1111 1111: reserved

Attribute memory space timing registers (FMC_PATT)

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FMC_PATT read/write register contains the timing information for NAND Flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND Flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to *Section 21.8.5: NAND Flash prewait feature*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATTHIZ								ATTHOLD							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATTWAIT								ATTSET							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ATTHIZ:** Attribute memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept in Hi-Z after the start of a NAND Flash write access to attribute memory space on socket. Only valid for writ transaction:
0000 0000: 0 x fmc_ker_ck cycle
1111 1110: 254 x fmc_ker_ck cycles
1111 1111: reserved.

Bits 23:16 **ATTHOLD:** Attribute memory hold time

These bits define the number of fmc_ker_ck clock cycles during which the address is held (and data for write access) after the command de-assertion (NWE, NOE), for NAND Flash read or write access to attribute memory space:
0000 0000: reserved
0000 0001: 1 x fmc_ker_ck cycle
1111 1110: 254 x fmc_ker_ck cycles
1111 1111: reserved.

Bits 15:8 **ATTWAIT:** Attribute memory wait time

These bits define the minimum number of x fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), for NAND Flash read or write access to attribute memory space. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:
0000 0000: reserved
0000 0001: 2 x fmc_ker_ck cycles (+ wait cycle introduced by de-assertion of NWAIT)
1111 1110: 255 x fmc_ker_ck cycles (+ wait cycle introduced by de-asserting NWAIT)
1111 1111: reserved.

Bits 7:0 **ATTSET:** Attribute memory setup time

These bits define the number of fmc_ker_ck (+1) clock cycles to set up address before the command assertion (NWE, NOE), for NAND Flash read or write access to attribute memory space:
0000 0000: 1 x fmc_ker_ck cycle
1111 1110: 255 x fmc_ker_ck cycles
1111 1111: reserved.

ECC result registers (FMC_ECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FMC NAND controller. When the CPU reads/writes the data from a NAND Flash memory page at the correct address (refer to *Section 21.8.6: Computation of the error correction code (ECC) in NAND Flash memory*), the data read/written from/to the NAND Flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCR registers), the CPU must read the computed ECC value from the FMC_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECCR register should be cleared after being read by setting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to ’1’.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ECC[31:0]:** ECC result

This field contains the value computed by the ECC computation logic. *Table 187* describes the contents of these bit fields.

Table 187. ECC result relevant bits

ECCPS[2:0]	Page size in bytes	ECC bits
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

21.9 SDRAM controller

21.9.1 SDRAM controller main features

The main features of the SDRAM controller are the following:

- Two SDRAM banks with independent configuration
- 8-bit, 16-bit, 32-bit data bus width
- 13-bits Address Row, 11-bits Address Column, 4 internal banks: 4x16Mx32bit (256 MB), 4x16Mx16bit (128 MB), 4x16Mx8bit (64 MB)
- Word, half-word, byte access
- SDRAM clock can be fmc_ker_ck/2 or fmc_ker_ck/3
- Automatic row and bank boundary management
- Multibank ping-pong access
- Programmable timing parameters
- Automatic Refresh operation with programmable Refresh rate
- Self-refresh mode
- Power-down mode
- SDRAM power-up initialization by software
- CAS latency of 1,2,3
- Cacheable Read FIFO with depth of 6 lines x32-bit (6 x14-bit address tag)

21.9.2 SDRAM External memory interface signals

At startup, the SDRAM I/O pins used to interface the FMC SDRAM controller with the external SDRAM devices must configured by the user application. The SDRAM controller I/O pins which are not used by the application, can be used for other purposes.

Table 188. SDRAM signals

SDRAM signal	I/O type	Description	Alternate function
SDCLK	O	SDRAM clock	-
SDCKE[1:0]	O	SDCKE0: SDRAM Bank 1 Clock Enable SDCKE1: SDRAM Bank 2 Clock Enable	-
SDNE[1:0]	O	SDNE0: SDRAM Bank 1 Chip Enable SDNE1: SDRAM Bank 2 Chip Enable	-
A[12:0]	O	Address	FMC_A[12:0]
D[31:0]	I/O	Bidirectional data bus	FMC_D[31:0]
BA[1:0]	O	Bank Address	FMC_A[15:14]
NRAS	O	Row Address Strobe	-
NCAS	O	Column Address Strobe	-
SDNWE	O	Write Enable	-
NBL[3:0]	O	Output Byte Mask for write accesses (memory signal name: DQM[3:0]	FMC_NBL[3:0]

21.9.3 SDRAM controller functional description

All SDRAM controller outputs (signals, address and data) change on the falling edge of the memory clock (FMC_SDCLK).

SDRAM initialization

The initialization sequence is managed by software. If the two banks are used, the initialization sequence must be generated simultaneously to Bank 1and Bank 2 by setting the Target Bank bits CTB1 and CTB2 in the FMC_SDCMR register:

1. Program the memory device features into the FMC_SDCRx register. The SDRAM clock frequency, RBURST and RPIPE must be programmed in the FMC_SDCR1 register.
2. Program the memory device timing into the FMC_SDTRx register. The TRP and TRC timings must be programmed in the FMC_SDTR1 register.
3. Set MODE bits to ‘001’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to start delivering the clock to the memory (SDCKE is driven high).
4. Wait during the prescribed delay period. Typical delay is around 100 µs (refer to the SDRAM datasheet for the required delay after power-up).
5. Set MODE bits to ‘010’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Precharge All” command.
6. Set MODE bits to ‘011’, and configure the Target Bank bits (CTB1 and/or CTB2) as well as the number of consecutive Auto-refresh commands (NRFS) in the FMC_SDCMR register. Refer to the SDRAM datasheet for the number of Auto-refresh commands that should be issued. Typical number is 8.
7. Configure the MRD field, set the MODE bits to ‘100’, and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Load Mode Register” command and program the SDRAM device. In particular the Burst Length (BL) has to be set to ‘1’) and the CAS latency has to be selected. If the Mode Register is not the same for both SDRAM banks, this step has to be repeated twice, once for each bank and the Target Bank bits set accordingly. For mobile SDRAM devices, the MRD field is also used to configure the extended mode register while issuing the Load Mode Register”
8. Program the refresh rate in the FMC_SDRTR register

   The refresh rate corresponds to the delay between refresh cycles. Its value must be adapted to SDRAM devices.

At this stage the SDRAM device is ready to accept commands. If a system reset occurs during an ongoing SDRAM access, the data bus might still be driven by the SDRAM device. Therefor the SDRAM device must be first reinitialized after reset before issuing any new access by the NOR Flash/PSRAM/SRAM or NAND Flash controller.

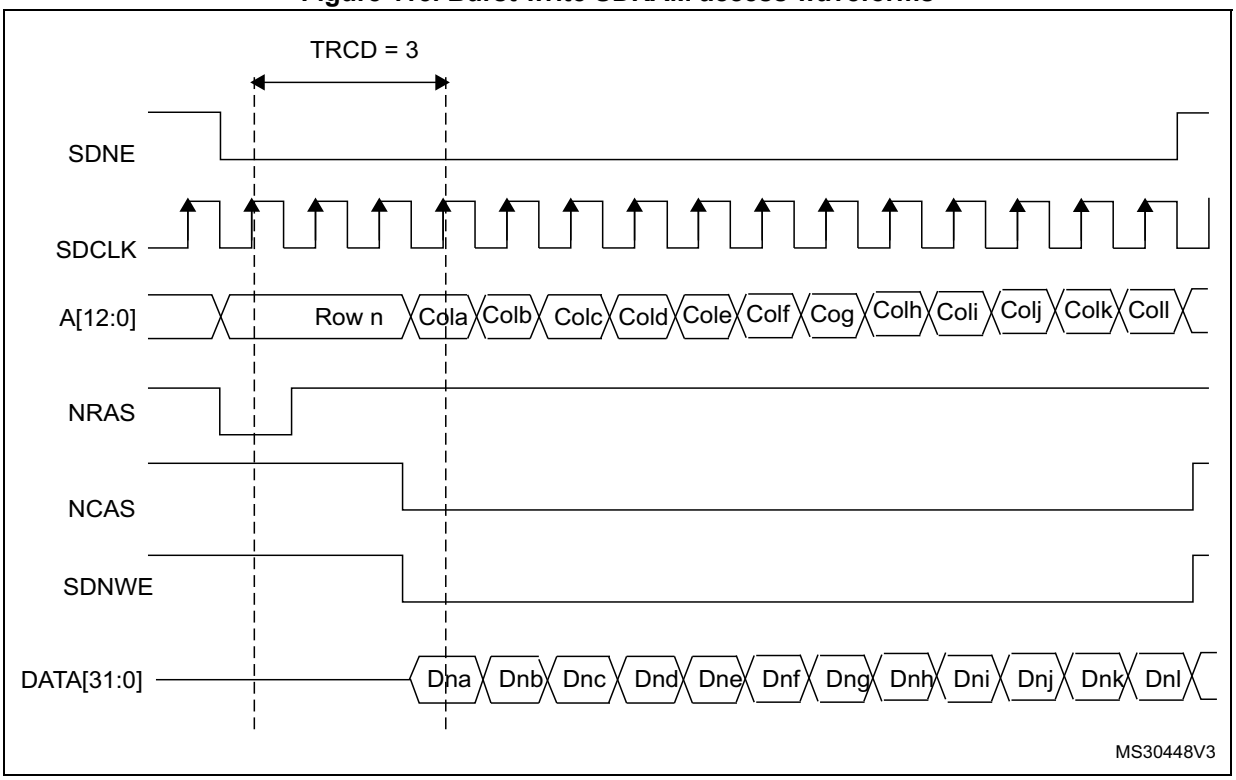
Note: If two SDRAM devices are connected to the FMC, all the accesses performed at the same time to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for SDRAM Bank 1 (TMRD andTRAS timings) in the FMC_SDTR1 register.

SDRAM controller write cycle

The SDRAM controller accepts single and burst write requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row for each bank to be able to perform consecutive write accesses to different banks (Multibank ping-pong access).

Before performing any write access, the SDRAM bank write protection must be disabled by clearing the WP bit in the FMC_SDCRx register.

Figure 115. Burst write SDRAM access waveforms



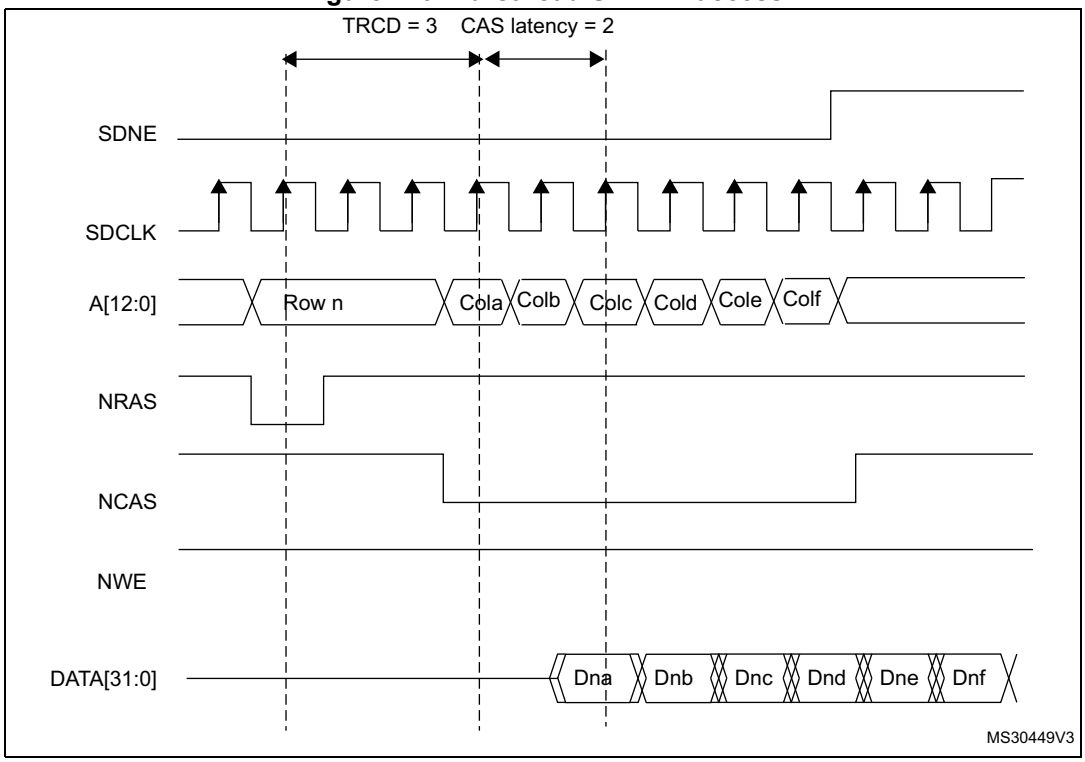
The SDRAM controller always checks the next access.

- If the next access is in the same row or in another active row, the write operation is carried out,
- if the next access targets another row (not active), the SDRAM controller generates a precharge command, activates the new row and initiates a write command.

SDRAM controller read cycle

The SDRAM controller accepts single and burst read requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row in each bank to be able to perform consecutive read accesses in different banks (Multibank ping-pong access).

Figure 116. Burst read SDRAM access



The FMC SDRAM controller features a Cacheable read FIFO (6 lines x 32 bits). It is used to store data read in advance during the CAS latency period (up to 3 memory clock cycles, programmed FMC_SDCRx) and during the RPIPE delay when set to 2xfmc_ker_ck clock cycles as configured in FMC_SDCR1) following this formula: CAS Latency + 1 + (RPIPE DIV2). The RBURST bit must be set in the FMC_SDCR1 register to anticipate the next read access.

Examples:

- CAS=3, RPIPE= 2xfmc_ker_ck. In this case, 5 data (not committed) are stored in the FIFO (4 data during CAS latency and 1 data during RPIPE delay)
- CAS=3, RPIPE= 1xfmc_ker_ck. In this case, 4 data (not committed) are stored in the FIFO (4 data during CAS latency)

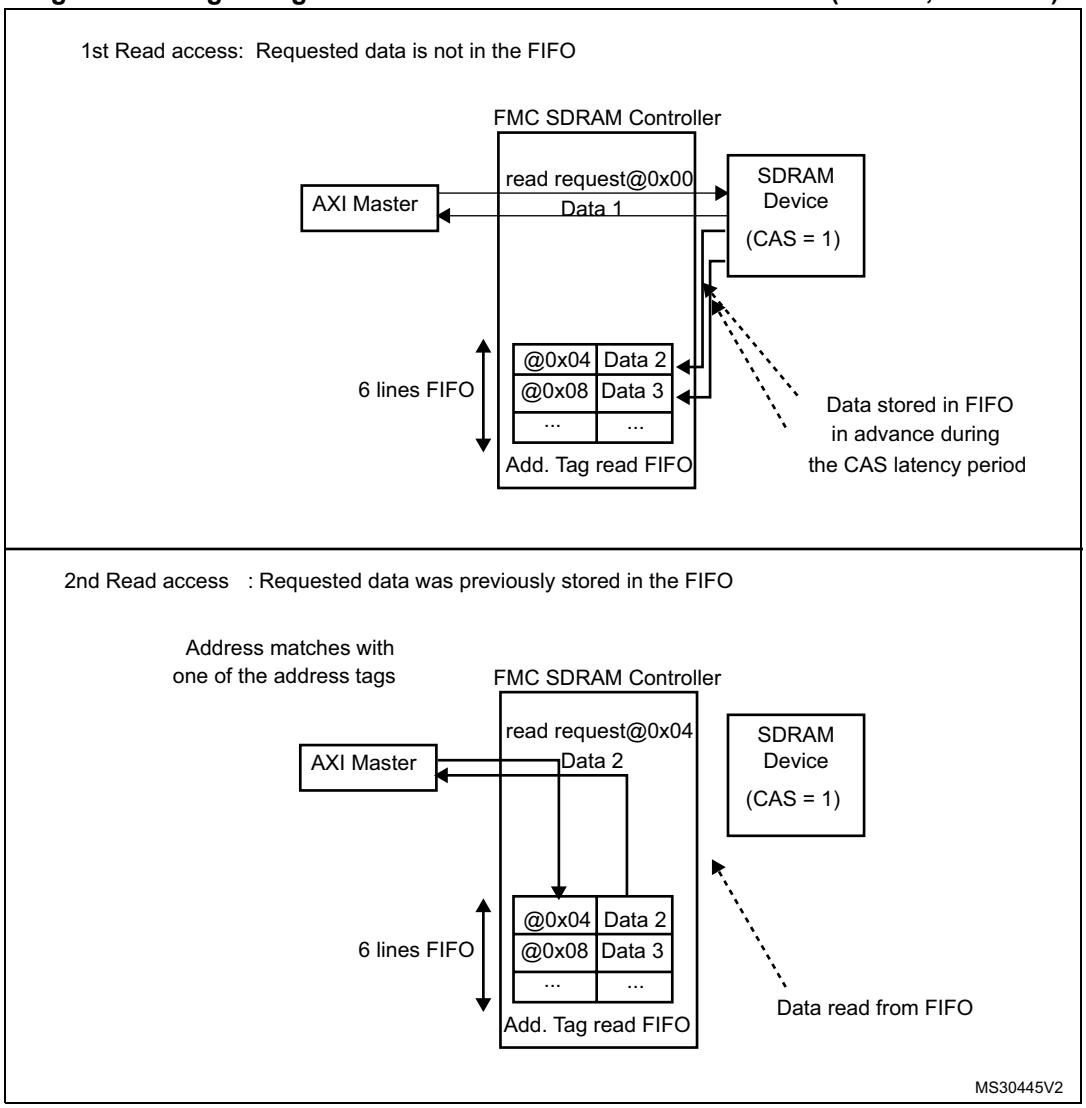
The read FIFO features a 14-bit address tag to each line to identify its content: 11 bits for the column address, 2 bits to select the internal bank and the active row, and 1 bit to select the SDRAM device

When the end of the row is reached in advance during an burst read transaction, the data read in advance (not committed) are not stored in the read FIFO. For single read access, data are correctly stored in the FIFO.

Each time a read request occurs, the SDRAM controller checks:

- If the address matches one of the address tags, data are directly read from the FIFO and the corresponding address tag/ line content is cleared and the remaining data in the FIFO are compacted to avoid empty lines.
- Otherwise, a new read command is issued to the memory and the FIFO is updated with new data. If the FIFO is full, the older data are lost.

Figure 117. Logic diagram of Read access with RBURST bit set (CAS=2, RPIPE=0)



During a write access or a Precharge command, the read FIFO is flushed and ready to be filled with new data.

After the first read request, if the current access was not performed to a row boundary, the SDRAM controller anticipates the next read access during the CAS latency period and the RPIPE delay (if configured). This is done by incrementing the memory address. The following condition must be met:

- RBURST control bit should be set to ‘1’ in the FMC_SDCR1 register.

The address management depends on the next AXI request:

- Next request is sequential (Burst access)

  In this case, the SDRAM controller increments the address.
- Next request is not sequential
  - If the new read request targets the same row or another active row, the new address is passed to the memory and the master is stalled for the CAS latency period, waiting for the new data from memory.
  - If the new read request does not target an active row, the SDRAM controller generates a Precharge command, activates the new row, and initiates a read command.

If the RBURST is reset, the read FIFO is not used.

Row and bank boundary management

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, the SDRAM controller executes the following operations:

1. Precharge of the active row,
2. Activation of the new row
3. Start of a read/write command.

At a row boundary, the automatic activation of the next row is supported for all columns and data bus width configurations.

If necessary, the SDRAM controller inserts additional clock cycles between the following commands:

- Between Precharge and Active commands to match TRP parameter (only if the next access is in a different row in the same bank),
- Between Active and Read commands to match the TRCD parameter.

These parameters are defined into the FMC_SDTRx register.

Refer to *Figure 115* and *Figure 116* for read and burst write access crossing a row boundary.

Figure 118. Read access crossing row boundary

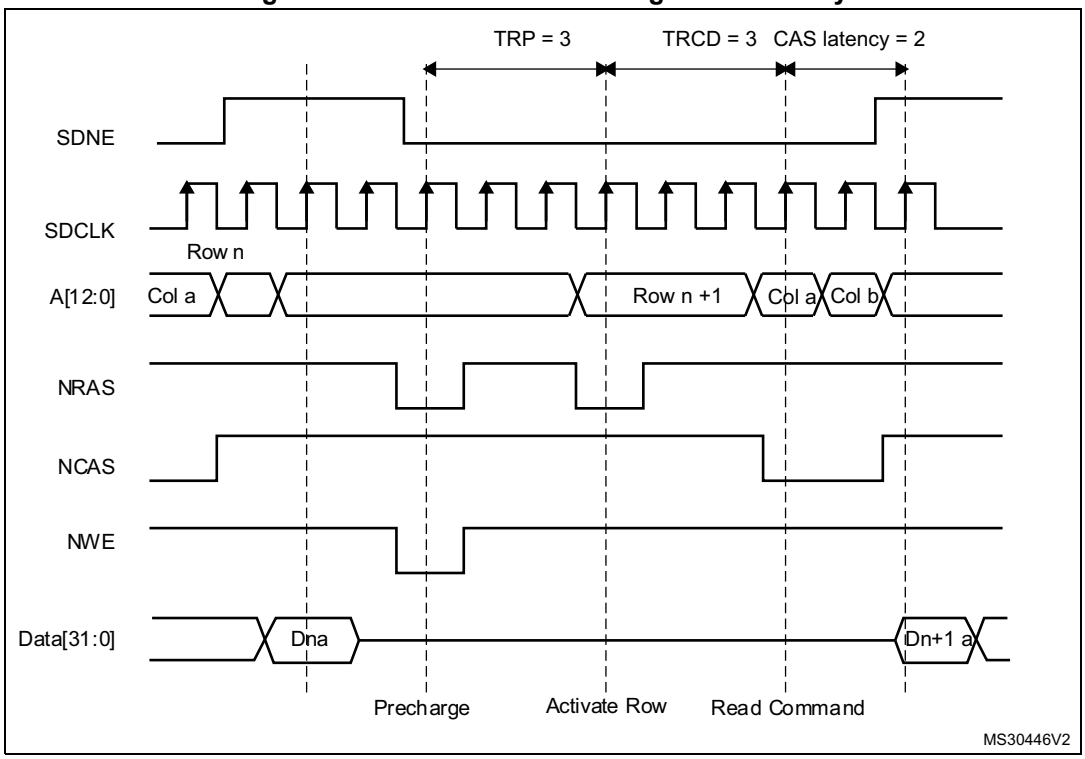
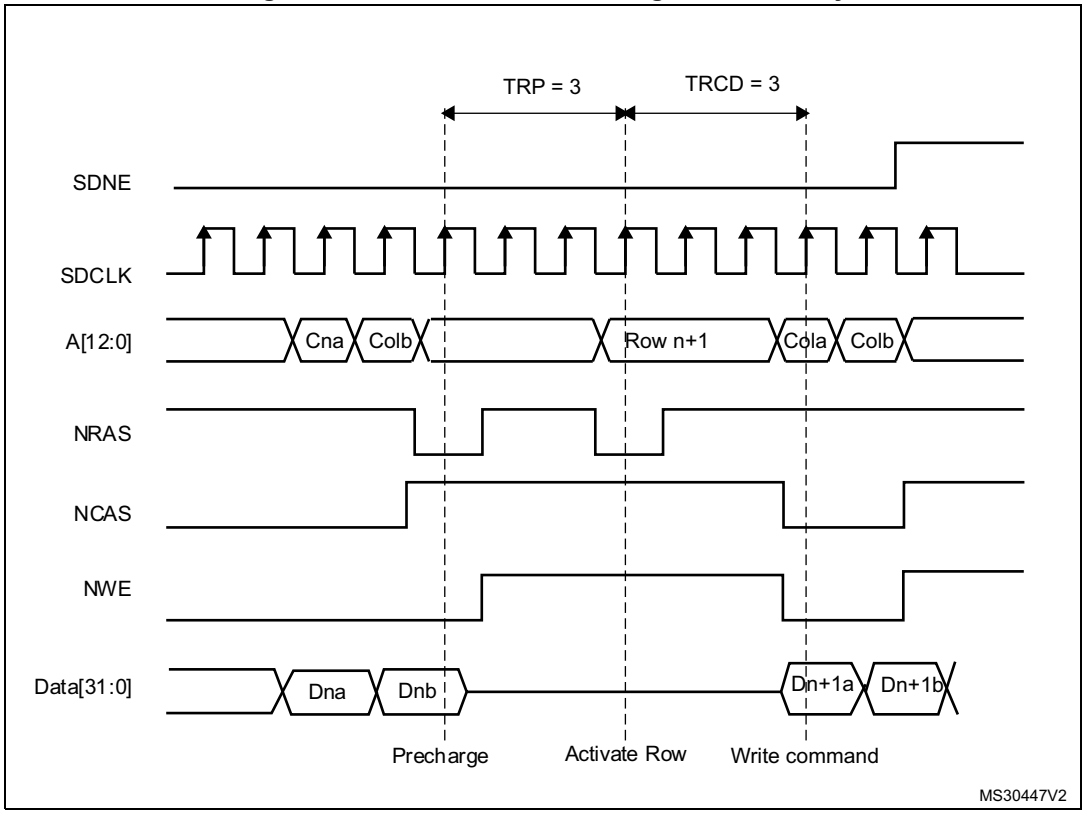


Figure 119. Write access crossing row boundary



If the next access is sequential and the current access crosses a bank boundary, the SDRAM controller activates the first row in the next bank and initiates a new read/write command. Two cases are possible:

- If the current bank is not the last one, the active row in the new bank must be precharged.At a bank boundary, the automatic activation of the next row is supported for all rows/columns and data bus width configuration.
- If the current bank is the last one, the automatic activation of the next row is supported only when addressing 13-bit rows, 11-bit columns, 4 internal banks and 32-bit data bus SDRAM devices. Otherwise, the SDRAM address range is violated and an AXI slave error is generated.
- In case of 13-bit row address, 11-bit column address, 4 internal banks and bus width 32-bit SDRAM memories, at boundary bank, the SDRAM controller continues to read/write from the second SDRAM device (assuming it has been initialized):
  a) The SDRAM controller activates the first row (after precharging the active row, if there is already an active row in the first internal bank, and initiates a new read/write command.
  b) If the first row is already activated, the SDRAM controller just initiates a read/write command.

SDRAM controller refresh cycle

The Auto-refresh command is used to refresh the SDRAM device content. The SDRAM controller periodically issues auto-refresh commands. An internal counter is loaded with the COUNT value in the register FMC_SDRTR. This value defines the number of memory clock cycles between the refresh cycles (refresh rate). When this counter reaches zero, an internal pulse is generated.

If a memory access is ongoing, the auto-refresh request is delayed. However, if the memory access and the auto-refresh requests are generated simultaneously, the auto-refresh request takes precedence.

If the memory access occurs during an auto-refresh operation, the request is buffered and processed when the auto-refresh is complete.

If a new auto-refresh request occurs while the previous one was not served, the RE (Refresh Error) bit is set in the Status register. An Interrupt is generated if it has been enabled (REIE = ‘1’).

If SDRAM lines are not in idle state (not all row are closed), the SDRAM controller generates a PALL (Precharge ALL) command before the auto-refresh.

If the Auto-refresh command is generated by the FMC_SDCMR Command Mode register (Mode bits = ‘011’), a PALL command (Mode bits =’ 010’) must be issued first.

21.9.4 Low-power modes

Two low-power modes are available:

- Self-refresh mode

  The auto-refresh cycles are performed by the SDRAM device itself to retain data without external clocking.
- Power-down mode

  The auto-refresh cycles are performed by the SDRAM controller.

Self-refresh mode

This mode is selected by setting the MODE bits to ‘101’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

The SDRAM clock stops running after a TRAS delay and the internal refresh timer stops counting only if one of the following conditions is met:

- A Self-refresh command is issued to both devices
- One of the devices is not activated (SDRAM bank is not initialized).

Before entering Self-Refresh mode, the SDRAM controller automatically issues a PALL command.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Self-refresh mode and the BUSY status flag remains set.

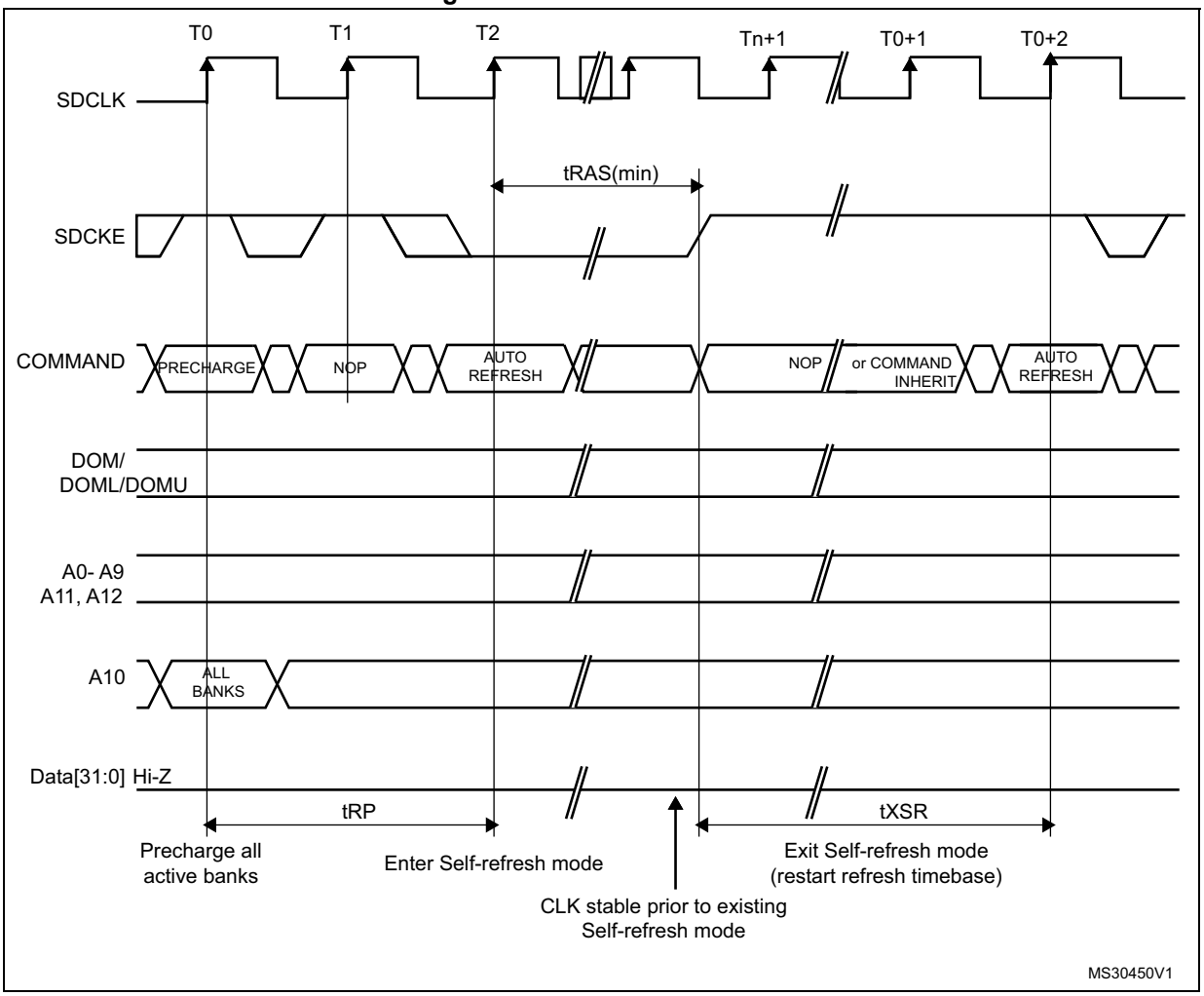
In Self-refresh mode, all SDRAM device inputs become don’t care except for SDCKE which remains low.

The SDRAM device must remain in Self-refresh mode for a minimum period of time of TRAS and can remain in Self-refresh mode for an indefinite period beyond that. To guarantee this minimum period, the BUSY status flag remains high after the Self-refresh activation during a TRAS delay.

As soon as an SDRAM device is selected, the SDRAM controller generates a sequence of commands to exit from Self-refresh mode. After the memory access, the selected device remains in Normal mode.

To exit from Self-refresh, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.

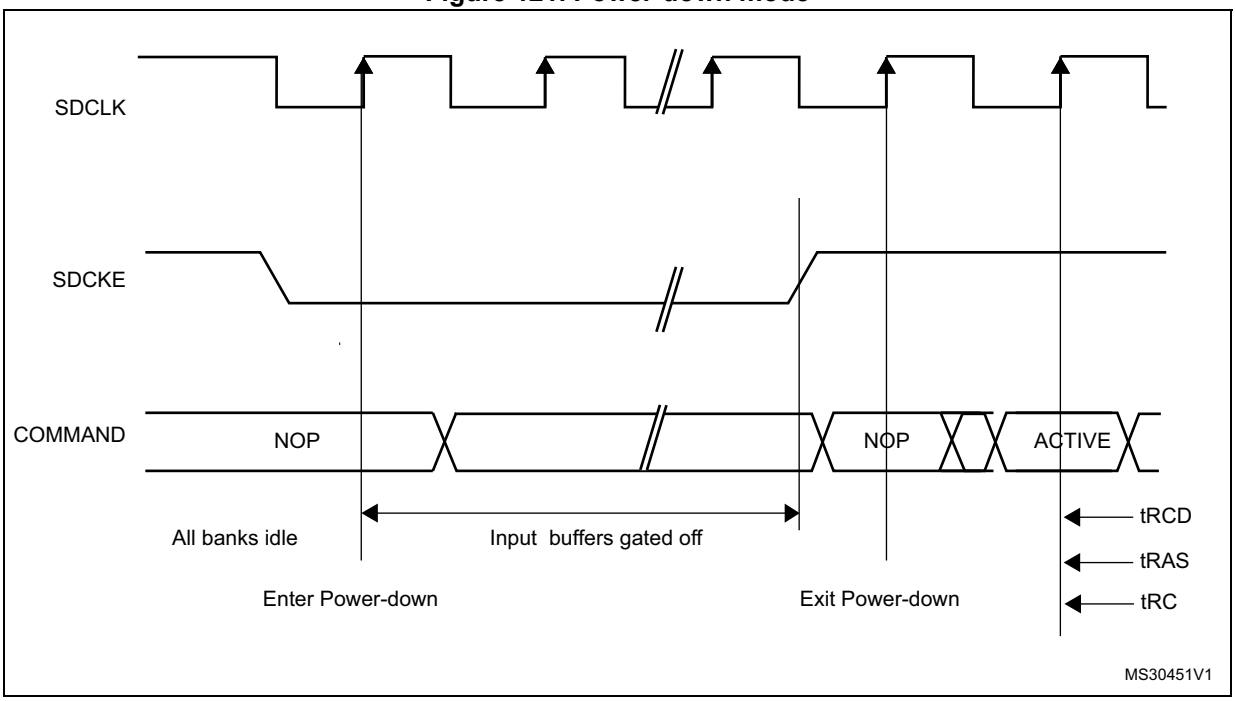
Figure 120. Self-refresh mode



Power-down mode

This mode is selected by setting the MODE bits to ‘110’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

Figure 121. Power-down mode



If the Write data FIFO is not empty, all data are sent to the memory before activating the Power-down mode.

As soon as an SDRAM device is selected, the SDRAM controller exits from the Power-down mode. After the memory access, the selected SDRAM device remains in Normal mode.

During Power-down mode, all SDRAM device input and output buffers are deactivated except for the SDCKE which remains low.

The SDRAM device cannot remain in Power-down mode longer than the refresh period and cannot perform the Auto-refresh cycles by itself. Therefore, the SDRAM controller carries out the refresh operation by executing the operations below:

1. Exit from Power-down mode and drive the SDCKE high
2. Generate the PALL command only if a row was active during Power-down mode
3. Generate the auto-refresh command
4. Drive SDCKE low again to return to Power-down mode.

To exit from Power-down mode, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.

21.9.5 SDRAM controller registers

SDRAM Control registers for SDRAM memory bank x (FMC_SDCRx)

Address offset: 0x140+ 4* (x - 1), (x = 1 to 2)

Reset value: 0x0000 02D0

This register contains the control parameters for each SDRAM memory bank

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RPIPE[1:0]		RBURST	SDCLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:13 **RPIPE[1:0]:** Read pipe

These bits define the delay, in fmc_ker_ck clock cycles, for reading data after CAS latency.
00: No fmc_ker_ck clock cycle delay
01: One fmc_ker_ck clock cycle delay
10: Two fmc_ker_ck clock cycle delay
11: reserved.

Note: The corresponding bits in the FMC_SDCR2 register is read only.

Bit 12 **RBURST:** Burst read

This bit enables Burst read mode. The SDRAM controller anticipates the next read commands during the CAS latency and stores data in the Read FIFO.
0: single read requests are not managed as bursts
1: single read requests are always managed as bursts

Note: The corresponding bit in the FMC_SDCR2 register is read only.

Bits 11:10 **SDCLK[1:0]:** SDRAM clock configuration

These bits define the SDRAM clock period for both SDRAM banks and allow disabling the clock before changing the frequency. In this case the SDRAM must be re-initialized.
00: SDCLK clock disabled
01: Reserved
10: SDCLK period = 2 x fmc_ker_ck periods
11: SDCLK period = 3 x fmc_ker_ck periods

Note: The corresponding bits in the FMC_SDCR2 register is read only.

Bit 9 **WP:** Write protection

This bit enables Write mode access to the SDRAM bank.
0: Write accesses allowed
1: Write accesses ignored

Bits 8:7 **CAS[1:0]:** CAS Latency
This bits sets the SDRAM CAS latency in number of memory clock cycles
00: reserved.
01: 1 cycle
10: 2 cycles
11: 3 cycles

Bit 6 **NB:** Number of internal banks
This bit sets the number of internal banks.
0: Two internal Banks
1: Four internal Banks

Bits 5:4 **MWID[1:0]:** Memory data bus width.
These bits define the memory device width.
00: 8 bits
01: 16 bits
10: 32 bits
11: reserved.

Bits 3:2 **NR[1:0]:** Number of row address bits
These bits define the number of bits of a row address.
00: 11 bit
01: 12 bits
10: 13 bits
11: reserved.

Bits 1:0 **NC[1:0]:** Number of column address bits
These bits define the number of bits of a column address.
00: 8 bits
01: 9 bits
10: 10 bits
11: 11 bits.

Note: Before modifying the RBURST or RPIPE settings or disabling the SDCLK clock, the user must first send a PALL command to make sure ongoing operations are complete.

SDRAM Timing registers for SDRAM memory bank x (FMC_SDTRx)

Address offset: 0x148 + 4 * (x - 1), (x = 1 to 2)

Reset value: 0x0FFF FFFF

This register contains the timing parameters of each SDRAM bank

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TRCD				TRP				TWR			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRC				TRAS				TXSR				TMRD			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TRCD[3:0]:** Row to column delay

These bits define the delay between the Activate command and a Read/Write command in number of memory clock cycles.
0000: 1 cycle.
0001: 2 cycles
....
1111: 16 cycles

Bits 23:20 **TRP[3:0]:** Row precharge delay

These bits define the delay between a Precharge command and another command in number of memory clock cycles. The TRP timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP must be programmed with the timing of the slowest device.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: The corresponding bits in the FMC_SDTR2 register are don't care.

Bits 19:16 **TWR[3:0]:** Recovery delay

These bits define the delay between a Write and a Precharge command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: TWR must be programmed to match the write recovery time (t_{WR}) defined in the SDRAM datasheet, and to guarantee that:

TWR ³ TRAS - TRCD and TWR ³TRC - TRCD - TRP

Example: TRAS= 4 cycles, TRCD= 2 cycles. So, TWR >= 2 cycles. TWR must be programmed to 0x1.

If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TWR timing corresponding to the slowest SDRAM device.

If only one SDRAM device is used, the TWR timing must be kept at reset value (0xF) for the not used bank.

Bits 15:12 **TRC[3:0]:** Row cycle delay

These bits define the delay between the Refresh command and the Activate command, as well as the delay between two consecutive Refresh commands. It is expressed in number of memory clock cycles. The TRC timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRC must be programmed with the timings of the slowest device.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: TRC must match the TRC and TRFC (Auto Refresh period) timings defined in the SDRAM device datasheet.

Note: The corresponding bits in the FMC_SDTR2 register are don't care.

Bits 11:8 **TRAS[3:0]:** Self refresh time

These bits define the minimum Self-refresh period in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Bits 7:4 **TXSR[3:0]:** Exit Self-refresh delay

These bits define the delay from releasing the Self-refresh command to issuing the Activate command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TXSR timing corresponding to the slowest SDRAM device.

Bits 3:0 **TMRD[3:0]:** Load Mode Register to Active

These bits define the delay between a Load Mode Register command and an Active or Refresh command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: If two SDRAM devices are connected, all the accesses performed simultaneously to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.

The TRP and TRC timings are only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP and TRC timings must be programmed with the timings of the slowest device.

SDRAM Command mode register (FMC_SDCMR)

Address offset: 0x150

Reset value: 0x0000 0000

This register contains the command issued when the SDRAM device is accessed. This register is used to initialize the SDRAM device, and to activate the Self-refresh and the Power-down modes. As soon as the MODE field is written, the command will be issued only to one or to both SDRAM banks according to CTB1 and CTB2 command bits. This register is the same for both SDRAM banks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRD							NRFS				CTB1	CTB2	MODE		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:9 **MRD[13:0]:** Mode Register definition
This 14-bit field defines the SDRAM Mode Register content. The Mode Register is programmed using the Load Mode Register command. The MRD[13:0] bits are also used to program the extended mode register for mobile SDRAM.

Bits 8:5 **NRFS[3:0]:** Number of Auto-refresh
These bits define the number of consecutive Auto-refresh commands issued when MODE = '011'.
0000: 1 Auto-refresh cycle
0001: 2 Auto-refresh cycles
....
1110: 15 Auto-refresh cycles
1111: 16 Auto-refresh cycles

Bit 4 **CTB1:** Command Target Bank 1
This bit indicates whether the command will be issued to SDRAM Bank 1 or not.
0: Command not issued to SDRAM Bank 1
1: Command issued to SDRAM Bank 1

Bit 3 **CTB2:** Command Target Bank 2
This bit indicates whether the command will be issued to SDRAM Bank 2 or not.
0: Command not issued to SDRAM Bank 2
1: Command issued to SDRAM Bank 2

Bits 2:0 **MODE[2:0]:** Command mode
These bits define the command issued to the SDRAM device.
000: Normal Mode
001: Clock Configuration Enable
010: PALL ("All Bank Precharge") command
011: Auto-refresh command
100: Load Mode Register
101: Self-refresh command
110: Power-down command
111: Reserved

Note: When a command is issued, at least one Command Target Bank bit (CTB1 or CTB2) must be set otherwise the command will be ignored.

Note: If two SDRAM banks are used, the Auto-refresh and PALL command must be issued simultaneously to the two devices with CTB1 and CTB2 bits set otherwise the command will be ignored.

Note: If only one SDRAM bank is used and a command is issued with it's associated CTB bit set, the other CTB bit of the unused bank must be kept to 0.

SDRAM refresh timer register (FMC_SDRTR)

Address offset: 0x154

Reset value: 0x0000 0000

This register sets the refresh rate in number of SDCLK clock cycles between the refresh cycles by configuring the Refresh Timer Count value.

$$\text{Refresh rate} = (\text{COUNT} + 1) \times \text{SDRAM clock frequency}$$

$$\text{COUNT} = (\text{SDRAM refresh period} / \text{Number of rows}) - 20$$

Below an example of refresh rate calculation:

$$\text{Refresh rate} = 64\ \text{ms} / (8196\text{rows}) = 7.81\mu\text{s}$$

where 64 ms is the SDRAM refresh period.

$$7.81\mu\text{s} \times 60\text{MHz} = 468.6$$

The refresh rate must be increased by 20 SDRAM clock cycles (as in the above example) to obtain a safe margin if an internal refresh request occurs when a read request has been accepted. It corresponds to a COUNT value of '0000111000000' (448).

This 13-bit field is loaded into a timer which is decremented using the SDRAM clock. This timer generates a refresh pulse when zero is reached. The COUNT value must be set at least to 41 SDRAM clock cycles.

As soon as the FMC_SDRTR register is programmed, the timer starts counting. If the value programmed in the register is '0', no refresh is carried out. This register must not be reprogrammed after the initialization procedure to avoid modifying the refresh rate.

Each time a refresh pulse is generated, this 13-bit COUNT field is reloaded into the counter.

If a memory access is in progress, the Auto-refresh request is delayed. However, if the memory access and Auto-refresh requests are generated simultaneously, the Auto-refresh takes precedence. If the memory access occurs during a refresh operation, the request is buffered to be processed when the refresh is complete.

This register is common to SDRAM bank 1 and bank 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REIE	COUNT													CRE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w

Bits 31: 15 Reserved, must be kept at reset value.

Bit 14 **REIE:** RES Interrupt Enable
0: Interrupt is disabled
1: An Interrupt is generated if RE = 1

Bits 13:1 **COUNT[12:0]:** Refresh Timer Count
This 13-bit field defines the refresh rate of the SDRAM device. It is expressed in number of memory clock cycles. It must be set at least to 41 SDRAM clock cycles (0x29).
Refresh rate = (COUNT + 1) x SDRAM frequency clock
COUNT = (SDRAM refresh period / Number of rows) - 20

Bit 0 **CRE:** Clear Refresh error flag
This bit is used to clear the Refresh Error Flag (RE) in the Status Register.
0: no effect
1: Refresh Error flag is cleared

Note: The programmed COUNT value must not be equal to the sum of the following timings: TWR+TRP+TRC+TRCD+4 memory clock cycles .

SDRAM Status register (FMC_SDSR)

Address offset: 0x158

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODES2		MODES1		RE
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:3 **MODES2:** Status Mode for Bank 2

These bits define the Status Mode of SDRAM Bank 2.

00: Normal Mode

01: Self-refresh mode

10: Power-down mode

Bits 2:1 **MODES1:** Status Mode for Bank 1

These bits define the Status Mode of SDRAM Bank 1.

00: Normal Mode

01: Self-refresh mode

10: Power-down mode

Bit 0 **RE:** Refresh error flag

0: No refresh error has been detected

1: A refresh error has been detected

An interrupt is generated if REIE = 1 and RE = 1

21.10 FMC register map

The following table summarizes the FMC registers.

Table 189. FMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**FMC_BCR1**	FMCEN	Res.	Res.	Res.	Res.	Res.	BMAP[1:0]		Res.	Res.	WFDIS	CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID		MTYP		MUXEN	MBKEN
0x08	**FMC_BCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value													0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x10	**FMC_BCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value													0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x18	**FMC_BCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value													0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x04	**FMC_BTR1**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x0C	**FMC_BTR2**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x14	**FMC_BTR3**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x1C	**FMC_BTR4**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x104	**FMC_BWTR1**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x10C	**FMC_BWTR2**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x114	**FMC_BWTR3**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x11C	**FMC_BWTR4**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]									ADDHLD[3:0]			ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0x80	**FMC_PCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS [2:0]				TAR[3:0]				TCLR[3:0]			Res.	Res.	ECCEN	PWID [1:0]		Res.	PBKEN	PWAITEN	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0			0	0	1	1	0	0	
0x84	**FMC_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
	Reset value																										1	0	0	0	0	0	0

Table 189. FMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x88	**FMC_PMEM**	MEMHIZx[7:0]								MEMHOLDx[7:0]								MEMWAITx[7:0]								MEMSETx[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x8C	**FMC_PATT**	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x94	**FMC_ECCR**	ECC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	**FMC_SDCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIPE[1:0]		RBURST	SDCLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC	
	Reset value																		0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
0x144	**FMC_SDCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIPE[1:0]		RBURST	SDCLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC	
	Reset value																		0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
0x148	**FMC_SDTR1**	Res.	Res.	Res.	Res.	TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]				TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14C	**FMC_SDTR2**	Res.	Res.	Res.	Res.	TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]				TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x150	**FMC_SDCMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD															NRFS[3:0]			CTB1	CTB2	MODE[2:0]		
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	**FMC_SDRTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REIE	COUNT[12:0]													CRE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158	**FMC_SDSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODES2[1:0]		MODES1[1:0]		RE
	Reset value																												0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

22 Quad-SPI interface (QUADSPI)

22.1 Introduction

The QUADSPI is a specialized communication interface targeting single, dual or quad SPI Flash memories. It can operate in any of the three following modes:

- indirect mode: all the operations are performed using the QUADSPI registers
- status polling mode: the external Flash memory status register is periodically read and an interrupt can be generated in case of flag setting
- memory-mapped mode: the external Flash memory is mapped to the microcontroller address space and is seen by the system as if it was an internal memory

Both throughput and capacity can be increased two-fold using dual-flash mode, where two Quad-SPI Flash memories are accessed simultaneously.

22.2 QUADSPI main features

- Three functional modes: indirect, status-polling, and memory-mapped
- Dual-flash mode, where 8 bits can be sent/received simultaneously by accessing two Flash memories in parallel.
- SDR and DDR support
- Fully programmable opcode for both indirect and memory mapped mode
- Fully programmable frame format for both indirect and memory mapped mode
- Integrated FIFO for reception and transmission
- 8, 16, and 32-bit data accesses are allowed
- MDMA trigger generation for FIFO threshold and transfer complete
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error

22.3 QUADSPI functional description

22.3.1 QUADSPI block diagram

Figure 122. QUADSPI block diagram when dual-flash mode is disabled

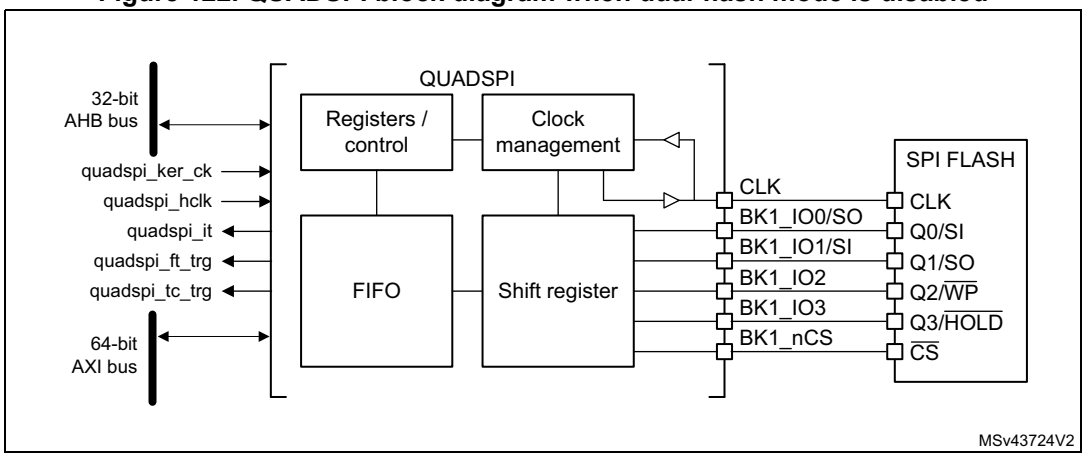
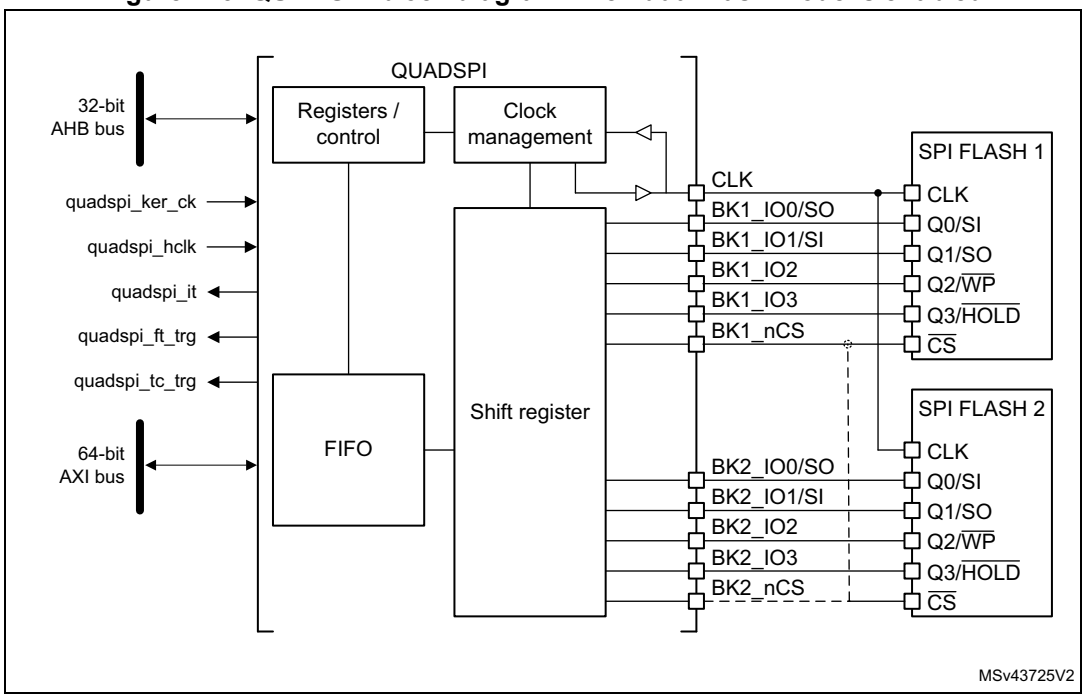


Figure 123. QUADSPI block diagram when dual-flash mode is enabled



22.3.2 QUADSPI pins and internal signals

Table 190 lists the QUADSPI internal signals.

Table 190. QUADSPI internal signals

Signal name	Signal type	Description
quadspi_ker_ck	Digital input	QUADSPI kernel clock
quadspi_hclk	Digital input	QUADSPI register interface clock
quadspi_it	Digital output	QUADSPI global interrupt
quadspi_ft_trg	Digital output	QUADSPI FIFO threshold trigger for MDMA
quadspi_tc_trg	Digital output	QUADSPI transfer complete trigger for MDMA

Table 191 lists the QUADSPI pins, six for interfacing with a single Flash memory, or 10 to 11 for interfacing with two Flash memories (FLASH 1 and FLASH 2) in dual-flash mode.

Table 191. QUADSPI pins

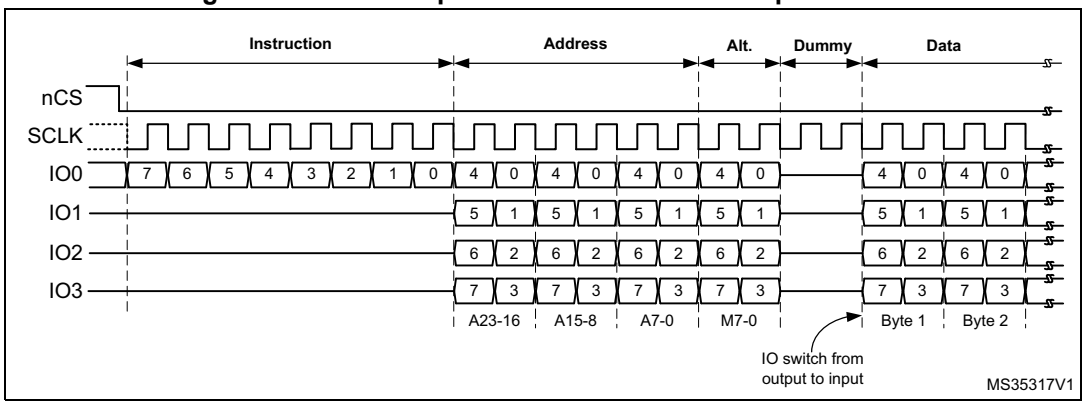
Signal name	Signal type	Description
CLK	Digital output	Clock to FLASH 1 and FLASH 2
BK1_IO0/SO	Digital input/output	Bidirectional IO in dual/quad modes or serial output in single mode, for FLASH 1
BK1_IO1/SI	Digital input/output	Bidirectional IO in dual/quad modes or serial input in single mode, for FLASH 1
BK1_IO2	Digital input/output	Bidirectional IO in quad mode, for FLASH 1
BK1_IO3	Digital input/output	Bidirectional IO in quad mode, for FLASH 1
BK2_IO0/SO	Digital input/output	Bidirectional IO in dual/quad modes or serial output in single mode, for FLASH 2
BK2_IO1/SI	Digital input/output	Bidirectional IO in dual/quad modes or serial input in single mode, for FLASH 2
BK2_IO2	Digital input/output	Bidirectional IO in quad mode, for FLASH 2
BK2_IO3	Digital input/output	Bidirectional IO in quad mode, for FLASH 2
BK1_nCS	Digital output	Chip select (active low) for FLASH 1. Can also be used for FLASH 2 if QUADSPI is always used in dual-flash mode.
BK2_nCS	Digital output	Chip select (active low) for FLASH 2. Can also be used for FLASH 1 if QUADSPI is always used in dual-flash mode.

22.3.3 QUADSPI command sequence

The QUADSPI communicates with the Flash memory using commands. Each command can include 5 phases: instruction, address, alternate byte, dummy, data. Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phase must be present.

nCS falls before the start of each command and rises again after each command finishes.

Figure 124. An example of a read command in quad mode



Instruction phase

During this phase, an 8-bit instruction, configured in INSTRUCTION field of QUADSPI_CCR[7:0] register, is sent to the Flash memory, specifying the type of operation to be performed.

Though most Flash memories can receive instructions only one bit at a time from the IO0/SO signal (single SPI mode), the instruction phase can optionally send 2 bits at a time (over IO0/IO1 in dual SPI mode) or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the IMODE[1:0] field of QUADSPI_CCR[9:8] register.

When IMODE = 00, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

Address phase

In the address phase, 1-4 bytes are sent to the Flash memory to indicate the address of the operation. The number of address bytes to be sent is configured in the ADSIZE[1:0] field of QUADSPI_CCR[13:12] register. In indirect and automatic-polling modes, the address bytes to be sent are specified in the ADDRESS[31:0] field of QUADSPI_AR register, while in memory-mapped mode the address is given directly via the AXI (from the Cortex® or from a DMA).

The address phase can send 1 bit at a time (over SO in single SPI mode), 2 bits at a time (over IO0/IO1 in dual SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the ADMODE[1:0] field of QUADSPI_CCR[11:10] register.

When ADMODE = 00, the address phase is skipped, and the command sequence proceeds directly to the next phase, if any.

Alternate-bytes phase

In the alternate-bytes phase, 1-4 bytes are sent to the Flash memory, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the ABSIZE[1:0] field of QUADSPI_CCR[17:16] register. The bytes to be sent are specified in the QUADSPI_ABR register.

The alternate-bytes phase can send 1 bit at a time (over SO in single SPI mode), 2 bits at a time (over IO0/IO1 in dual SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the ABMODE[1:0] field of QUADSPI_CCR[15:14] register.

When ABMODE = 00, the alternate-bytes phase is skipped, and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when dual-mode is used and only two cycles are used for the alternate bytes. In this case, firmware can use quad-mode (ABMODE = 11) and send a byte with bits 7 and 3 of ALTERNATE set to '1' (keeping the IO3 line high), and bits 6 and 2 set to '0' (keeping the IO2 line low). In this case the upper two bits of the nibble to be sent are placed in bits 4:3 of ALTERNATE while the lower two bits are placed in bits 1 and 0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE should be set to 0x8A (1000_1010).

Dummy-cycles phase

In the dummy-cycles phase, 1-31 cycles are given without any data being sent or received, in order to allow the Flash memory the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] field of QUADSPI_CCR[22:18] register. In both SDR and DDR modes, the duration is specified as a number of full CLK cycles.

When DCYC is zero, the dummy-cycles phase is skipped, and the command sequence proceeds directly to the data phase, if present.

The operating mode of the dummy-cycles phase is determined by DMODE.

In order to assure enough "turn-around" time for changing the data signals from output mode to input mode, there must be at least one dummy cycle when using dual or quad mode to receive data from the Flash memory.

Data phase

During the data phase, any number of bytes can be sent to, or received from the Flash memory.

In indirect and automatic-polling modes, the number of bytes to be sent/received is specified in the QUADSPI_DLR register.

In indirect write mode the data to be sent to the Flash memory must be written to the QUADSPI_DR register, while in indirect read mode the data received from the Flash memory is obtained by reading from the QUADSPI_DR register.

In memory-mapped mode, the data which is read is sent back directly over the AXI to the Cortex or to a DMA.

The data phase can send/receive 1 bit at a time (over SO/SI in single SPI mode), 2 bits at a time (over IO0/IO1 in dual SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad SPI mode). This can be configured using the ABMODE[1:0] field of QUADSPI_CCR[15:14] register.

When DMODE = 00, the data phase is skipped, and the command sequence finishes immediately by raising nCS. This configuration must only be used in only indirect write mode.

22.3.4 QUADSPI signal interface protocol modes

Single SPI mode

Legacy SPI mode allows just a single bit to be sent/received serially. In this mode, data is sent to the Flash memory over the SO signal (whose I/O shared with IO0). Data received from the Flash memory arrives via SI (whose I/O shared with IO1).

The different phases can each be configured separately to use this single bit mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in QUADSPI_CCR) to 01.

In each phase which is configured in single mode:

- IO0 (SO) is in output mode
- IO1 (SI) is in input mode (high impedance)
- IO2 is in output mode and forced to ‘0’ (to deactivate the “write protect” function)
- IO3 is in output mode and forced to ‘1’ (to deactivate the “hold” function)

This is the case even for the dummy phase if DMODE = 01.

Dual SPI mode

In dual SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 10.

In each phase which is configured in dual mode:

- IO0/IO1 are at high-impedance (input) during the data phase for read operations, and outputs in all other cases
- IO2 is in output mode and forced to ‘0’
- IO3 is in output mode and forced to ‘1’

In the dummy phase when DMODE = 01, IO0/IO1 are always high-impedance.

Quad SPI mode

In quad SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use quad SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 11.

In each phase which is configured in quad mode, IO0/IO1/IO2/IO3 are all are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE = 11, IO0/IO1/IO2/IO3 are all high-impedance.

IO2 and IO3 are used only in Quad SPI mode. If none of the phases are configured to use Quad SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while QUADSPI is active.

SDR mode

By default, the DDRM bit (QUADSPI_CCR[31]) is 0 and the QUADSPI operates in single data rate (SDR) mode.

In SDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the QUADSPI assumes that the Flash memories also send the data using CLK's falling edge. By default (when SSHIFT = 0), the signals are sampled using the following (rising) edge of CLK.

DDR mode

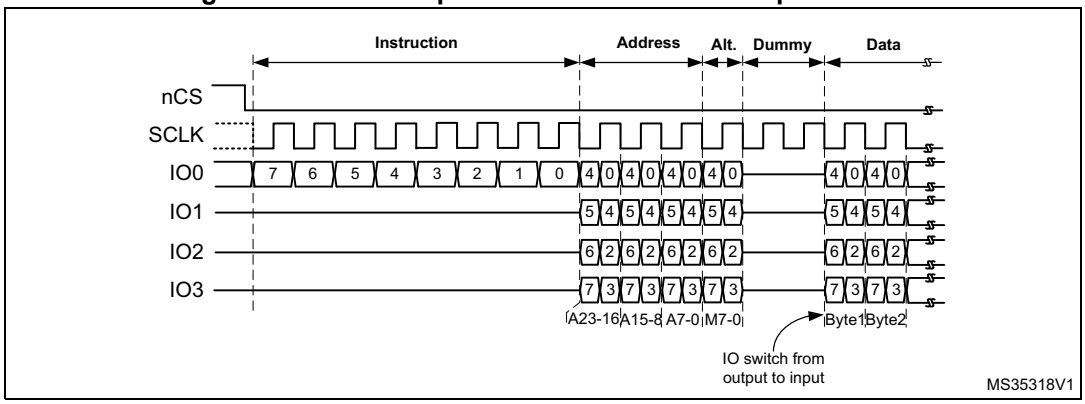
When the DDRM bit (QUADSPI_CCR[31]) is set to 1, the QUADSPI operates in double data rate (DDR) mode.

In DDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals in the address/alternate-byte/data phases, a bit is sent on each of the falling and rising edges of CLK.

The instruction phase is not affected by DDRM. The instruction is always sent using CLK's falling edge.

When receiving data in DDR mode, the QUADSPI assumes that the Flash memories also send the data using both rising and falling CLK edges. When DDRM = 1, firmware must clear SSHIFT bit (bit 4 of QUADSPI_CR). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

Figure 125. An example of a DDR command in quad mode



Dual-flash mode

When the DFM bit (bit 6 of QUADSPI_CR) is 1, the QUADSPI is in dual-flash mode, where two external quad SPI Flash memories (FLASH 1 and FLASH 2) are used in order to send/receive 8 bits (or 16 bits in DDR mode) every cycle, effectively doubling the throughput as well as the capacity.

Each of the Flash memories use the same CLK and optionally the same nCS signals, but each have separate IO0, IO1, IO2, and IO3 signals.

Dual-flash mode can be used in conjunction with single-bit, dual-bit, and quad-bit modes, as well as with either SDR or DDR mode.

The Flash memory size, as specified in FSIZE[4:0] (QUADSPI_DCR[20:16]), should reflect the total Flash memory capacity, which is double the size of one individual component.

If address X is even, then the byte which the QUADSPI gives for address X is the byte at the address X/2 of FLASH 1, and the byte which the QUADSPI gives for address X+1 is the byte at the address X/2 of FLASH 2. In other words, bytes at even addresses are all stored in FLASH 1 and bytes at odd addresses are all stored in FLASH 2.

When reading the Flash memories status registers in dual-flash mode, twice as many bytes should be read compared to doing the same read in single-flash mode. This means that if each Flash memory gives 8 valid bits after the instruction for fetching the status register, then the QUADSPI must be configured with a data length of 2 bytes (16 bits), and the QUADSPI will receive one byte from each Flash memory. If each Flash memory gives a status of 16 bits, then the QUADSPI must be configured to read 4 bytes to get all the status bits of both Flash memories in dual-flash mode. The least-significant byte of the result (in the data register) is the least-significant byte of FLASH 1 status register, while the next byte is the least-significant byte of FLASH 2 status register. Then, the third byte of the data register is FLASH 1 second byte, while the forth byte is FLASH 2 second byte (in the case that the Flash memories have 16-bit status registers).

An even number of bytes must always be accessed in dual-flash mode. For this reason, bit 0 of the data length field (QUADSPI_DLR[0]) is stuck at 1 when DRM = 1.

In dual-flash mode, the behavior of FLASH 1 interface signals are basically the same as in normal mode. FLASH 2 interface signals have exactly the same waveforms as FLASH 1 during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each Flash memory always receives the same instruction and the same address. Then, during the data phase, the BK1_IOx and BK2_IOx buses are both transferring data in parallel, but the data that are sent to (or received from) FLASH 1 are distinct from those of FLASH 2.

22.3.5 QUADSPI indirect mode

When in indirect mode, commands are started by writing to QUADSPI registers and data is transferred by writing or reading the data register, in the same way as for other communication peripherals.

When FMODE = 00 (QUADSPI_CCR[27:26]), the QUADSPI is in indirect write mode, where bytes are sent to the Flash memory during the data phase. Data are provided by writing to the data register (QUADSPI_DR).

When FMODE = 01, the QUADSPI is in indirect read mode, where bytes are received from the Flash memory during the data phase. Data are recovered by reading QUADSPI_DR.

The number of bytes to be read/written is specified in the data length register (QUADSPI_DLR). If QUADSPI_DLR = 0xFFFF_FFFF (all 1's), then the data length is considered undefined and the QUADSPI simply continues to transfer data until the end of Flash memory (as defined by FSIZE) is reached. If no bytes are to be transferred, DMODE (QUADSPI_CCR[25:24]) should be set to 00.

If QUADSPI_DLR = 0xFFFF_FFFF and FSIZE = 0x1F (max value indicating a 4GB Flash memory), then in this special case the transfers continue indefinitely, stopping only after an abort request or after the QUADSPI is disabled. After the last memory address is read (at address 0xFFFF_FFFF), reading continues with address = 0x0000_0000.

When the programmed number of bytes to be transmitted or received is reached, TCF is set and an interrupt is generated if TCIE = 1. In the case of undefined number of data, the TCF is set when the limit of the external SPI memory is reached according to the Flash memory size defined in the QUADSPI_CR.

Triggering the start of a command

Essentially, a command starts as soon as firmware gives the last information that is necessary for this command. Depending on the QUADSPI configuration, there are three different ways to trigger the start of a command in indirect mode. The commands starts immediately after:

1. a write is performed to INSTRUCTION[7:0] (QUADSPI_CCR), if no address is necessary (when ADMODE = 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
2. a write is performed to ADDRESS[31:0] (QUADSPI_AR), if an address is necessary (when ADMODE != 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
3. a write is performed to DATA[31:0] (QUADSPI_DR), if an address is necessary (when ADMODE != 00) and if data needs to be provided by the firmware (when FMODE = 00 and DMODE != 00)

Writes to the alternate byte register (QUADSPI_ABR) never trigger the communication start. If alternate bytes are required, they must be programmed before.

As soon as a command is started, the BUSY bit (bit 5 of QUADSPI_SR) is automatically set.

FIFO and data management

In indirect mode, data go through a 32-byte FIFO which is internal to the QUADSPI. FLEVEL[5:0] (QUADSPI_SR[13:8]) indicates how many bytes are currently being held in the FIFO.

In indirect write mode (FMODE = 00), firmware adds data to the FIFO when it writes QUADSPI_DR. Word writes add 4 bytes to the FIFO, halfword writes add 2 bytes, and byte writes add only 1 byte. If firmware adds too many bytes to the FIFO (more than is indicated by DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

Byte/halfword accesses to QUADSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES[3:0] is used to define a FIFO threshold. When the threshold is reached, the FTF (FIFO threshold flag) is set. In indirect read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there are data in the FIFO after the last byte is read from the Flash memory, regardless of the FTHRES setting. In indirect write mode, FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when FTF is set. FTF is cleared by HW as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

In indirect read mode when the FIFO becomes full, the QUADSPI temporarily stops reading bytes from the Flash memory to avoid an overrun. Note that the reading of the Flash memory does not restart until 4 bytes become vacant in the FIFO (when FLEVEL ≤ 11). Thus, when FTHRES ≥ 13, the application must take care to read enough bytes to assure that the QUADSPI starts retrieving data from the Flash memory again. Otherwise, the FTF flag stays at '0' as long as 11 < FLEVEL < FTHRES.

22.3.6 QUADSPI status flag polling mode

In automatic-polling mode, the QUADSPI periodically starts a command to read a defined number of status bytes (up to 4). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

The accesses to the Flash memory begin in the same way as in indirect read mode: if no address is required (AMODE = 00), accesses begin as soon as the QUADSPI_CCR is written. Otherwise, if an address is required, the first access begins when QUADSPI_AR is written. BUSY goes high at this point and stays high even between the periodic accesses.

The contents of MASK[31:0] (QUADSPI_PSMAR) are used to mask the data from the Flash memory in automatic-polling mode. If the MASK[n] = 0, then bit n of the result is masked and not considered. If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] (QUADSPI_PSMAR), then there is a match for bit n.

If the polling match mode bit (PMM, bit 23 of QUADSPI_CR) is 0, then “AND” match mode is activated. This means status match flag (SMF) is set only when there is a match on all of the unmasked bits.

If PMM = 1, then “OR” match mode is activated. This means SMF is set if there is a match on any of the unmasked bits.

An interrupt is called when SMF is set if SMIE = 1.

If the automatic-polling-mode-stop (APMS) bit is set, operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at ‘1’ and the periodic accesses continue until there is an abort or the QUADSPI is disabled (EN = 0).

The data register (QUADSPI_DR) contains the latest received status bytes (the FIFO is deactivated). The content of the data register is not affected by the masking used in the matching logic. The FTF status bit is set as soon as a new reading of the status is complete, and FTF is cleared as soon as the data is read.

22.3.7 QUADSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

It is forbidden to access QUADSPI Flash bank area before having properly configured and enabled the QUADSPI peripheral.

No more than 256MB can addressed even if the Flash memory capacity is larger.

If an access is made to an address outside of the range defined by FSIZE but still within the 256MB range, then a bus error is given. The effect of this error depends on the bus master that attempted the access:

- If it is the Cortex® CPU, bus fault exception is generated when enabled (or a hard fault exception when bus fault is disabled)
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, halfword, and word access types are all supported.

Support for execute in place (XIP) operation is implemented, where the QUADSPI anticipates the next microcontroller access and load in advance the byte at the following address. If the subsequent access is indeed made at a continuous address, the access will be completed faster since the value is already prefetched.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with nCS maintained low, even if no access to the Flash memory occurs for a long time. Since Flash memories tend to consume more when nCS is held low, the application might want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that nCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without any access since when the FIFO becomes full with prefetch data.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is a timeout, there is an abort, or the peripheral is disabled.

22.3.8 QUADSPI Free running clock mode

When configured in Free running clock mode, the QUADSPI peripheral continuously outputs the clock for test and calibration purposes.

Free running clock mode is entered as soon as the Free running clock mode bit (FRCM) is set in the QUADSPI communication configuration register (QUADSPI_CCR). It is exited by setting the ABORT bit of the QUADSPI control register (QUADSPI_CR).

When the QUADSPI operates in Free running clock mode:

- the clock is running continuously,
- nCS stays High (external device deselected),
- data lines are released (High-Z),
- the BUSY flag of the QUADSPI status register (QUADSPI_SR) is set.

22.3.9 QUADSPI Flash memory configuration

The device configuration register (QUADSPI_DCR) can be used to specify the characteristics of the external SPI Flash memory.

The FSIZE[4:0] field defines the size of external memory using the following formula:

Number of bytes in Flash memory = $2^{[FSIZE+1]}$

FSIZE+1 is effectively the number of address bits required to address the Flash memory. The Flash memory capacity can be up to 4GB (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256MB.

If DFM = 1, FSIZE indicates the total capacity of the two Flash memories together.

When the QUADSPI executes two commands, one immediately after the other, it raises the chip select signal (nCS) high between the two commands for only one CLK cycle by default. If the Flash memory requires more time between commands, the chip select high time (CSHT) field can be used to specify the minimum number of CLK cycles (up to 8) that nCS must remain high.

The clock mode (CKMODE) bit indicates the CLK signal logic level in between commands (when nCS = 1).

22.3.10 QUADSPI delayed data sampling

By default, the QUADSPI samples the data driven by the Flash memory one half of a CLK cycle after the Flash memory drives the signal.

In case of external signal delays, it may be beneficial to sample the data later. Using the SSHIFT bit (bit 4 of QUADSPI_CR), the sampling of the data can be shifted by half of a CLK cycle.

Clock shifting is not supported in DDR mode: the SSHIFT bit must be clear when DDRM bit is set.

22.3.11 QUADSPI configuration

The QUADSPI configuration is done in two phases:

- QUADSPI IP configuration
- QUADSPI Flash memory configuration

Once configured and enabled, the QUADSPI can be used in one of its three operating modes: indirect mode, status-polling mode, or memory-mapped mode.

QUADSPI IP configuration

The QUADSPI IP is configured using the QUADSPI_CR. The user shall configure the clock prescaler division factor and the sample shifting settings for the incoming data.

DDR mode can be set through the DDRM bit. Once enabled, the address and the alternate bytes are sent on both clock edges and the data are sent/received on both clock edges. Regardless of the DDRM bit setting, instructions are always sent in SDR mode.

FIFO level for either MDMA trigger generation or interrupt generation is programmed in the FTHRES bits.

If timeout counter is needed, the TCEN bit can be set and the timeout value programmed in the QUADSPI_LPTR register.

Dual-flash mode can be activated by setting DFM to 1.

QUADSPI Flash memory configuration

The parameters related to the targeted external Flash memory are configured through the QUADSPI_DCR register.The user shall program the Flash memory size in the FSIZE bits, the Chip Select minimum high time in the CSHT bits, and the functional mode (Mode 0 or Mode 3) in the MODE bit.

22.3.12 QUADSPI usage

The operating mode is selected using FMODE[1:0] (QUADSPI_CCR[27:26]).

Indirect mode procedure

When FMODE is programmed to 00, indirect write mode is selected and data can be sent to the Flash memory. With FMODE = 01, indirect read mode is selected where data can be read from the Flash memory.

When the QUADSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in the QUADSPI_DLR.
2. Specify the frame format, mode and instruction code in the QUADSPI_CCR.
3. Specify optional alternate byte to be sent right after the address phase in the QUADSPI_ABR.
4. Specify the operating mode in the QUADSPI_CR.
5. Specify the targeted address in the QUADSPI_AR.
6. Read/Write the data from/to the FIFO through the QUADSPI_DR.

When writing the control register (QUADSPI_CR) the user specifies the following settings:

- The enable bit (EN) set to '1'
- Timeout counter enable bit (TCEN)
- Sample shift setting (SSHIFT)
- FIFO threshold level (FTRHES) to indicate when the FTF flag should be set
- Interrupt enables
- Automatic polling mode parameters: match mode and stop mode (valid when FMODE = 11)
- Clock prescaler

When writing the communication configuration register (QUADSPI_CCR) the user specifies the following parameters:

- The instruction byte through the INSTRUCTION bits
- The way the instruction has to be sent through the IMODE bits (1/2/4 lines)
- The way the address has to be sent through the ADMODE bits (None/1/2/4 lines)
- The address size (8/16/24/32-bit) through the ADSIZE bits
- The way the alternate bytes have to be sent through the ABMODE (None/1/2/4 lines)
- The alternate bytes number (1/2/3/4) through the ABSIZE bits
- The presence or not of dummy bytes through the DBMODE bit
- The number of dummy bytes through the DCYC bits
- The way the data have to be sent/received (None/1/2/4 lines) through the DMODE bits

If neither the address register (QUADSPI_AR) nor the data register (QUADSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as QUADSPI_CCR is written. This is the case when both ADMODE and DMODE are 00, or if just ADMODE = 00 when in indirect read mode (FMODE = 01).

When an address is required (ADMODE is not 00) and the data register does not need to be written (when FMODE = 01 or DMODE = 00), the command sequence starts as soon as the address is updated with a write to QUADSPI_AR.

In case of data transmission (FMODE = 00 and DMODE! = 00), the communication start is triggered by a write in the FIFO through QUADSPI_DR.

Status flag polling mode

The status flag polling mode is enabled setting the FMODE field (QUADSPI_CCR[27:26]) to 10. In this mode, the programmed frame will be sent and the data retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in QUADSPI_DLR, it will be ignored and only 4 bytes will be read.

The periodicity is specified in the QUADSPI_PISR register.

Once the status data has been retrieved, it can internally be processed i order to:

- set the status match flag and generate an interrupt if enabled
- stop automatically the periodic retrieving of the status bytes

The received value can be masked with the value stored in the QUADSPI_PSMKR and ORed or ANDed with the value stored in the QUADSPI_PSMAR.

In case of match, the status match flag is set and an interrupt is generated if enabled, and the QUADSPI can be automatically stopped if the AMPS bit is set.

In any case, the latest retrieved value is available in the QUADSPI_DR.

Memory-mapped mode

In memory-mapped mode, the external Flash memory is seen as internal memory but with some latency during accesses. Only read operations are allowed to the external Flash memory in this mode.

Memory-mapped mode is entered by setting the FMODE to 11 in the QUADSPI_CCR register.

The programmed instruction and frame is sent when a master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate linear reads. Any access to QUADSPI_DR in this mode returns zero.

The data length register (QUADSPI_DLR) has no meaning in memory-mapped mode.

22.3.13 Sending the instruction only once

Some Flash memories (e.g. Winbound) might provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. One can take advantage of such a feature using the SIOO bit (QUADSPI_CCR[28]).

SIOO is valid for all functional modes (indirect, automatic polling, and memory-mapped). If the SIOO bit is set, the instruction is sent only for the first command following a write to QUADSPI_CCR. Subsequent command sequences skip the instruction phase, until there is a write to QUADSPI_CCR.

SIOO has no effect when IMODE = 00 (no instruction).

22.3.14 QUADSPI error management

An error can be generated in the following case:

- In indirect mode or status flag polling mode when a wrong address has been programmed in the QUADSPI_AR (according to the Flash memory size defined by FSIZE[4:0] in the QUADSPI_DCR): this will set the TEF and an interrupt is generated if enabled.
- Also in indirect mode, if the address plus the data length exceeds the Flash memory size, TEF will be set as soon as the access is triggered.
- In memory-mapped mode, when an out of range access is done by a master or when the QUADSPI is disabled: this will generate a bus error as a response to the faulty bus master request.
- When a master is accessing the memory mapped space while the memory mapped mode is disabled: this will generate a bus error as a response to the faulty bus master request.

22.3.15 QUADSPI busy bit and abort functionality

Once the QUADSPI starts an operation with the Flash memory, the BUSY bit is automatically set in the QUADSPI_SR.

In indirect mode, the BUSY bit is reset once the QUADSPI has completed the requested command sequence and the FIFO is empty.

In automatic-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1, or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on a timeout event or on an abort.

Any operation can be aborted by setting the ABORT bit in the QUADSPI_CR. Once the abort is completed, the BUSY bit and the ABORT bit are automatically reset, and the FIFO is flushed.

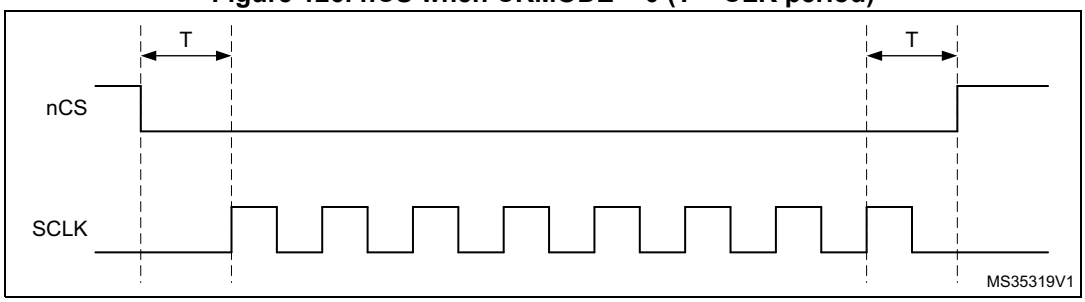
Note: Some Flash memories might misbehave if a write operation to a status registers is aborted.

22.3.16 nCS behavior

By default, nCS is high, deselecting the external Flash memory. nCS falls before an operation begins and rises as soon as it finishes.

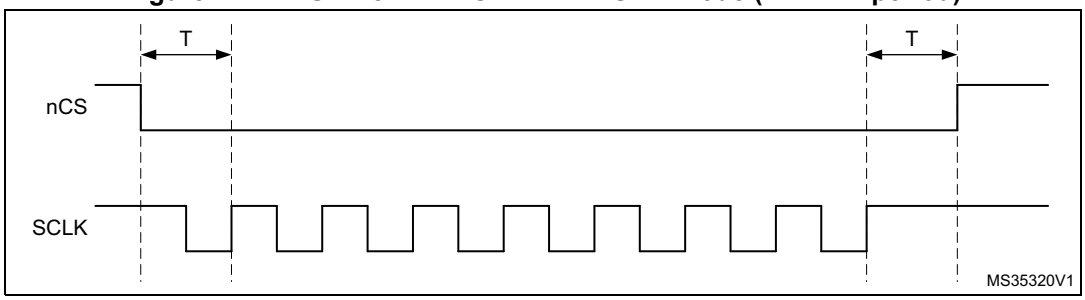
When CKMODE = 0 (“mode0”, where CLK stays low when no operation is in progress) nCS falls one CLK cycle before an operation first rising CLK edge, and nCS rises one CLK cycle after the operation final rising CLK edge, as shown in *Figure 126*.

Figure 126. nCS when CKMODE = 0 (T = CLK period)



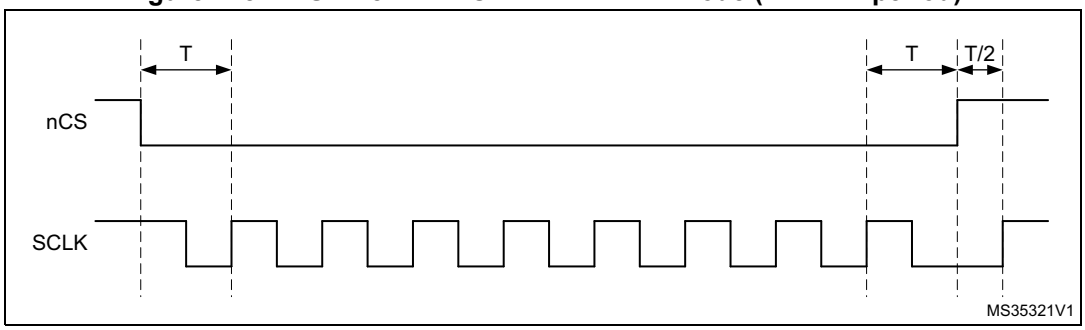
When CKMODE=1 (“mode3”, where CLK goes high when no operation is in progress) and DDRM=0 (SDR mode), nCS still falls one CLK cycle before an operation first rising CLK edge, and nCS rises one CLK cycle after the operation final rising CLK edge, as shown in *Figure 127*.

Figure 127. nCS when CKMODE = 1 in SDR mode (T = CLK period)



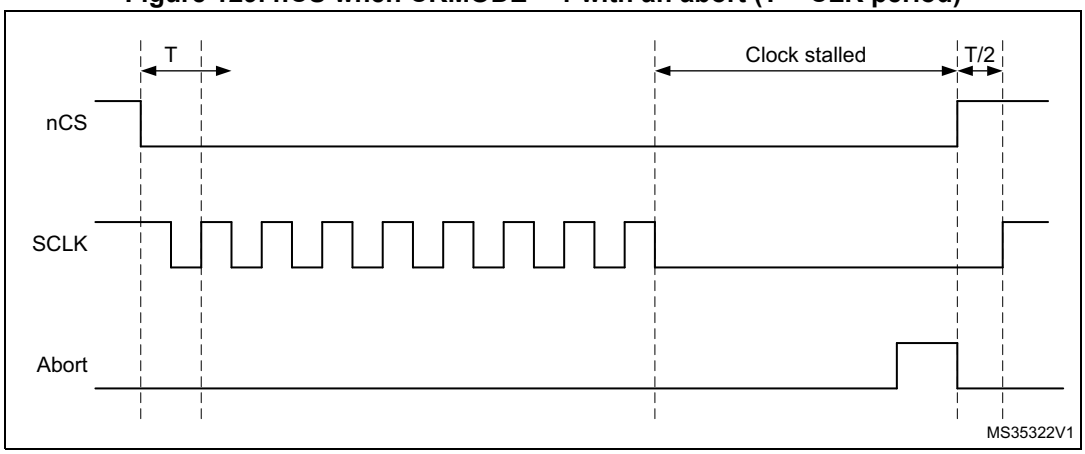
When CKMODE = 1 (“mode3”) and DDRM = 1 (DDR mode), nCS falls one CLK cycle before an operation first rising CLK edge, and nCS rises one CLK cycle after the operation final active rising CLK edge, as shown in *Figure 128*. Because DDR operations must finish with a falling edge, CLK is low when nCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 128. nCS when CKMODE = 1 in DDR mode (T = CLK period)



When the FIFO stays full in a read operation or if the FIFO stays empty in a write operation, the operation stalls and CLK stays low until firmware services the FIFO. If an abort occurs when an operation is stalled, nCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later, as shown in *Figure 129*.

Figure 129. nCS when CKMODE = 1 with an abort (T = CLK period)



When not in dual-flash mode (DFM = 0), only FLASH 1 is accessed and thus the BK2_nCS stays high. In dual-flash mode, BK2_nCS behaves exactly the same as BK1_nCS. Thus, if there is a FLASH 2 and if the application always stays in dual-flash mode, then FLASH 2 may use BK1_nCS and the pin outputting BK2_nCS can be used for other functions.

22.4 QUADSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Table 192. QUADSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

22.5 QUADSPI registers

22.5.1 QUADSPI control register (QUADSPI_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESCALER[7:0]								PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTHRES[4:0]					FSEL	DFM	Res.	SSHIFT	TCEN	Res.	ABORT	EN
			rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw

Bits 31:24 **PRESCALER[7:0]**: Clock prescaler

This field defines the scaler factor for generating CLK based on the quadspi_ker_ck clock (value+1).

0: $F_{CLK} = F_{quadspi_ker_ck}$, quadspi_ker_ck clock used directly as QUADSPI CLK (prescaler bypassed)

1: $F_{CLK} = F_{quadspi_ker_ck}/2$

2: $F_{CLK} = F_{quadspi_ker_ck}/3$

...

255: $F_{CLK} = F_{quadspi_ker_ck}/256$

For odd clock division factors, CLK's duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.

This field can be modified only when BUSY = 0.

Bit 23 **PMM**: Polling match mode

This bit indicates which method should be used for determining a "match" during automatic polling mode.

0: AND match mode. SMF is set if all the unmasked bits received from the Flash memory match the corresponding bits in the match register.

1: OR match mode. SMF is set if any one of the unmasked bits received from the Flash memory matches its corresponding bit in the match register.

This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic poll mode stop

This bit determines if automatic polling is stopped after a match.

0: Automatic polling mode is stopped only by abort or by disabling the QUADSPI.

1: Automatic polling mode stops as soon as there is a match.

This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: TimeOut interrupt enable

This bit enables the TimeOut interrupt.

0: Interrupt disable

1: Interrupt enabled

Bit 19 **SMIE**: Status match interrupt enable
This bit enables the status match interrupt.
0: Interrupt disable
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disable
1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]** FIFO threshold level
Defines, in indirect mode, the threshold number of bytes in the FIFO that will cause the FIFO threshold flag (FTF, QUADSPI_SR[2]) to be set.
In indirect write mode (FMODE = 00):
0: FTF is set if there are 1 or more free bytes available to be written to in the FIFO
1: FTF is set if there are 2 or more free bytes available to be written to in the FIFO
...
31: FTF is set if there are 32 free bytes available to be written to in the FIFO
In indirect read mode (FMODE = 01):
0: FTF is set if there are 1 or more valid bytes that can be read from the FIFO
1: FTF is set if there are 2 or more valid bytes that can be read from the FIFO
...
31: FTF is set if there are 32 valid bytes that can be read from the FIFO

Bit 7 **FSEL**: Flash memory selection
This bit selects the Flash memory to be addressed in single flash mode (when DFM = 0).
0: FLASH 1 selected
1: FLASH 2 selected
This bit can be modified only when BUSY = 0.
This bit is ignored when DFM = 1.

Bit 6 **DFM**: Dual-flash mode
This bit activates dual-flash mode, where two external Flash memories are used simultaneously to double throughput and capacity.
0: Dual-flash mode disabled
1: Dual-flash mode enabled
This bit can be modified only when BUSY = 0.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **SSHIFT**: Sample shift

By default, the QUADSPI samples data 1/2 of a CLK cycle after the data is driven by the Flash memory. This bit allows the data is to be sampled later in order to account for external signal delays.

0: No shift

1: 1/2 cycle shift

Firmware must assure that SSHIFT = 0 when in DDR mode (when DDRM = 1).

This field can be modified only when BUSY = 0.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when memory-mapped mode (FMODE = 11) is selected. Activating this bit causes the chip select (nCS) to be released (and thus reduces consumption) if there has not been an access after a certain amount of time, where this time is defined by TIMEOUT[15:0] (QUADSPI_LPTR).

Enable the timeout counter.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with nCS maintained low, even if no access to the Flash memory occurs for a long time. Since Flash memories tend to consume more when nCS is held low, the application might want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that nCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without an access since when the FIFO becomes full with prefetch data.

0: Timeout counter is disabled, and thus the chip select (nCS) remains active indefinitely after an access in memory-mapped mode.

1: Timeout counter is enabled, and thus the chip select is released in memory-mapped mode after TIMEOUT[15:0] cycles of Flash memory inactivity.

This bit can be modified only when BUSY = 0.

Bit 2 Reserved

Bit 1 **ABORT**: Abort request

This bit aborts the on-going command sequence. It is automatically reset once the abort is complete.

This bit stops the current transfer.

In polling mode or memory-mapped mode, this bit also reset the APM bit or the DM bit.

0: No abort requested

1: Abort requested

Bit 0 **EN**: Enable

Enable the QUADSPI.

0: QUADSPI is disabled

1: QUADSPI is enabled

22.5.2 QUADSPI device configuration register (QUADSPI_DCR)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSIZE[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CSHT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	CK MODE
					rw	rw	rw								rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **FSIZE[4:0]**: Flash memory size

This field defines the size of external memory using the following formula:
Number of bytes in Flash memory = $2^{[FSIZE+1]}$
FSIZE+1 is effectively the number of address bits required to address the Flash memory. The Flash memory capacity can be up to 4GB (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256MB.
If DFM = 1, FSIZE indicates the total capacity of the two Flash memories together.
This field can be modified only when BUSY = 0.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **CSHT[2:0]**: Chip select high time

CSHT+1 defines the minimum number of CLK cycles which the chip select (nCS) must remain high between commands issued to the Flash memory.
0: nCS stays high for at least 1 cycle between Flash memory commands
1: nCS stays high for at least 2 cycles between Flash memory commands
...
7: nCS stays high for at least 8 cycles between Flash memory commands
This field can be modified only when BUSY = 0.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **CKMODE**: Mode 0 / mode 3

This bit indicates the level that CLK takes between commands (when nCS = 1).
0: CLK must stay low while nCS is high (chip select released). This is referred to as mode 0.
1: CLK must stay high while nCS is high (chip select released). This is referred to as mode 3.
This field can be modified only when BUSY = 0.

22.5.3 QUADSPI status register (QUADSPI_SR)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
		r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This field gives the number of valid bytes which are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full. In memory-mapped mode and in automatic status polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **BUSY**: Busy

This bit is set when an operation is on going. This bit clears automatically when the operation with the Flash memory is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic polling mode when the unmasked received data matches the corresponding bits in the match register (QUADSPI_PSMAR). It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after reads from the Flash memory are complete. It is cleared automatically as soon as threshold condition is no longer true.

In automatic polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

22.5.4 QUADSPI flag clear register (QUADSPI_FCR)

Address offset: 0x000C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
											w	w		w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in the QUADSPI_SR register

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in the QUADSPI_SR register

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in the QUADSPI_SR register

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the QUADSPI_SR register

22.5.5 QUADSPI data length register (QUADSPI_DLR)

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and status-polling modes. A value no greater than 3 (indicating 4 bytes) should be used for status-polling mode.
All 1s in indirect mode means undefined length, where QUADSPI will continue until the end of memory, as defined by FSIZE.
0x0000_0000: 1 byte is to be transferred
0x0000_0001: 2 bytes are to be transferred
0x0000_0002: 3 bytes are to be transferred
0x0000_0003: 4 bytes are to be transferred
...
0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred
0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred
0xFFFF_FFFF: undefined length -- all bytes until the end of Flash memory (as defined by FSIZE) are to be transferred. Continue reading indefinitely if FSIZE = 0x1F.
DL[0] is stuck at '1' in dual-flash mode (DFM = 1) even when '0' is written to this bit, thus assuring that each access transfers an even number of bytes.
This field has no effect when in memory-mapped mode (FMODE = 10).
This field can be written only when BUSY = 0.

22.5.6 QUADSPI communication configuration register (QUADSPI_CCR)

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DDRM	DHHC	FRCM	SIOO	FMODE[1:0]		DMODE[1:0]		Res.	DCYC[4:0]					ABSIZE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]		INSTRUCTION[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DDRM**: Double data rate mode

This bit sets the DDR mode for the address, alternate byte and data phase:
0: DDR Mode disabled
1: DDR Mode enabled
This field can be written only when BUSY = 0.

Bit 30 **DHHC**: DDR hold

Delay the data output by 1/4 of the QUADSPI output clock cycle in DDR mode:
0: Delay the data output using analog delay
1: Delay the data output by 1/4 of a QUADSPI output clock cycle.
This feature is only active in DDR mode.
This field can be written only when BUSY = 0.

Bit 29 **FRCM:** Free Running Clock Mode

When this bit is set, the QUADSPI peripheral enters Free running clock mode regardless of the FMODE bits.
0: Normal mode
1: Free running clock mode
This bit can be written only when BUSY = 0.

Bit 28 **SIOO**: Send instruction only once mode
See *Section 22.3.13: Sending the instruction only once on page 872*. This bit has no effect when IMODE = 00.
0: Send instruction on every transaction
1: Send instruction only for the first command
This field can be written only when BUSY = 0.

Bits 27:26 **FMODE[1:0]**: Functional mode
This field defines the QUADSPI functional mode of operation.
00: Indirect write mode
01: Indirect read mode
10: Automatic polling mode
11: Memory-mapped mode
This field can be written only when BUSY = 0.

Bits 25:24 **DMODE[1:0]**: Data mode
This field defines the data phase's mode of operation:
00: No data
01: Data on a single line
10: Data on two lines
11: Data on four lines
This field also determines the dummy phase mode of operation.
This field can be written only when BUSY = 0.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **DCYC[4:0]**: Number of dummy cycles
This field defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of CLK cycles (0-31).
This field can be written only when BUSY = 0.

Bits 17:16 **ABSIZE[1:0]**: Alternate bytes size
This bit defines alternate bytes size:
00: 8-bit alternate byte
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes
This field can be written only when BUSY = 0.

Bits 15:14 **ABMODE[1:0]**: Alternate bytes mode
This field defines the alternate-bytes phase mode of operation:
00: No alternate bytes
01: Alternate bytes on a single line
10: Alternate bytes on two lines
11: Alternate bytes on four lines
This field can be written only when BUSY = 0.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bit defines address size:
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address
This field can be written only when BUSY = 0.

Bits 11:10 **ADMODE[1:0]**: Address mode
This field defines the address phase mode of operation:
00: No address
01: Address on a single line
10: Address on two lines
11: Address on four lines
This field can be written only when BUSY = 0.

Bits 9:8 **IMODE[1:0]**: Instruction mode
This field defines the instruction phase mode of operation:
00: No instruction
01: Instruction on a single line
10: Instruction on two lines
11: Instruction on four lines
This field can be written only when BUSY = 0.

Bits 7:0 **INSTRUCTION[7:0]**: Instruction
Instruction to be send to the external SPI device.
This field can be written only when BUSY = 0.

22.5.7 QUADSPI address register (QUADSPI_AR)

Address offset: 0x0018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address
Address to be send to the external Flash memory
Writes to this field are ignored when BUSY = 0 or when FMODE = 11 (memory-mapped mode).
In dual flash mode, ADDRESS[0] is automatically stuck to ‘0’ as the address should always be even

22.5.8 QUADSPI alternate bytes registers (QUADSPI_ABR)

Address offset: 0x001C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate Bytes

Optional data to be send to the external SPI device right after the address.
This field can be written only when BUSY = 0.

22.5.9 QUADSPI data register (QUADSPI_DR)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device.

In indirect write mode, data written to this register is stored on the FIFO before it is sent to the Flash memory during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect read mode, reading this register gives (via the FIFO) the data which was received from the Flash memory. If the FIFO does not have as many bytes as requested by the read operation and if BUSY=1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic polling mode, this register contains the last data read from the Flash memory (without masking).

Word, halfword, and byte accesses to this register are supported. In indirect write mode, a byte write adds 1 byte to the FIFO, a halfword write 2, and a word write 4. Similarly, in indirect read mode, a byte read removes 1 byte from the FIFO, a halfword read 2, and a word read 4. Accesses in indirect mode must be aligned to the bottom of this register: a byte read must read DATA[7:0] and a halfword read must read DATA[15:0].

22.5.10 QUADSPI polling status mask register (QUADSPI _PSMKR)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in polling mode.
For bit n:
0: Bit n of the data received in automatic polling mode is masked and its value is not considered in the matching logic
1: Bit n of the data received in automatic polling mode is unmasked and its value is considered in the matching logic
This field can be written only when BUSY = 0.

22.5.11 QUADSPI polling status match register (QUADSPI _PSMAR)

Address offset: 0x0028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MATCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MATCH[31:0]**: Status match

Value to be compared with the masked status register to get a match.
This field can be written only when BUSY = 0.

22.5.12 QUADSPI polling interval register (QUADSPI _PIR)

Address offset: 0x002C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval

Number of CLK cycles between to read during automatic polling phases.
This field can be written only when BUSY = 0.

22.5.13 QUADSPI low-power timeout register (QUADSPI_LPTR)

Address offset: 0x0030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the QUADSPI prefetches the subsequent bytes and holds these bytes in the FIFO. This field indicates how many CLK cycles the QUADSPI waits after the FIFO becomes full until it raises nCS, putting the Flash memory in a lower-consumption state.
This field can be written only when BUSY = 0.

22.5.14 QUADSPI register map

Table 193. QUADSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**QUADSPI_CR**	PRESCALER[7:0]								PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES [4:0]					FSEL	DFM	Res.	SSHIFT	TCEN	Res.	ABORT	EN
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0	0	0	0	0		0	0		0	0
0x0004	**QUADSPI_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	CSHT			Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKMODE
	Reset value												0	0	0	0	0						0	0	0								0
0x0008	**QUADSPI_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[6:0]							Res.	Res.	BUS	TOF	SMF	FTF	TCF	TEF
	Reset value																		0	0	0	0	0	0	0			0	0	0	0	0	0
0x000C	**QUADSPI_FCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																												0	0		0	0
0x0010	**QUADSPI_DLR**	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	**QUADSPI_CCR**	DDRM	DHHC	FRCM	SIOO	FMODE[1:0]		DMODE[1:0]		Res.	DCYC[4:0]					ABSIZE[1:0]		ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]		INSTRUCTION[7:0]							
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	**QUADSPI_AR**	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	**QUADSPI_ABR**	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	**QUADSPI_DR**	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	**QUADSPI_ PSMKR**	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	**QUADSPI_ PSMAR**	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	**QUADSPI_PIR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	**QUADSPI_ LPTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

23 Delay block (DLYB)

23.1 Introduction

The delay block (DLYB) is used to generate an output clock which is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or QUADSPI interface.

The delay is voltage- and temperature-dependent, which may require the application to re-configure and recenter the output clock phase with the receive data.

23.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 to 208 MHz
- Up to 12 oversampling phases.

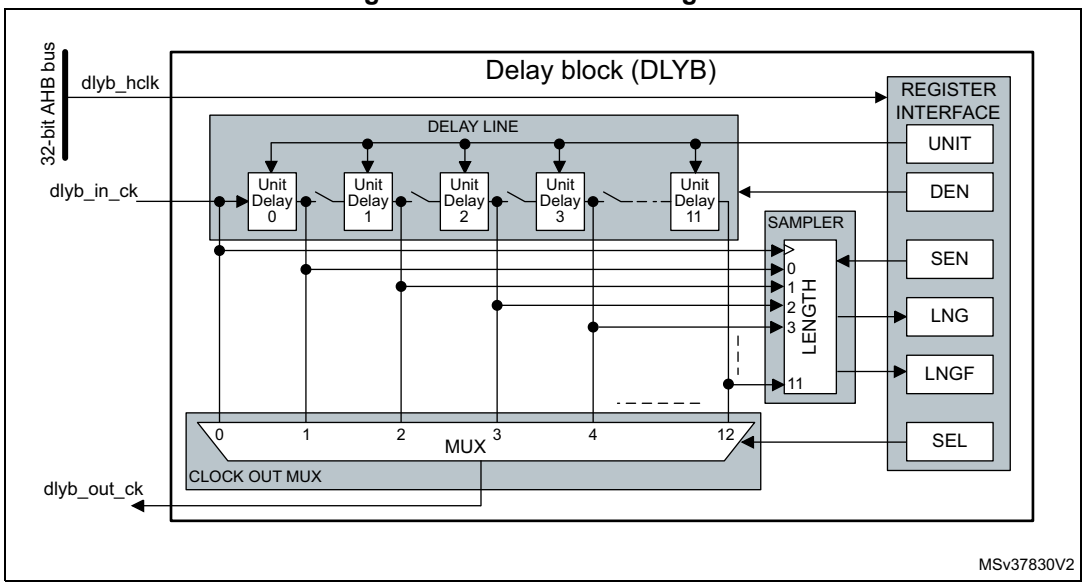
23.3 DLYB functional description

23.3.1 DLYB diagram

The delay block includes of the following sub-blocks:

- Register interface block providing AHB access to the Delay Block registers.
- Delay line supporting the unit delays.
- Delay line length sampling
- Output clock selection multiplexer

Figure 130. DLYB block diagram



23.3.2 DLYB pins and internal signals

Table 194 lists the DLYB internal signals.

Table 194. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

23.3.3 General description

The delay block is enabled by setting the DEN bit in the DLYB_CR register (see *Section 23.4.1: DLYB control register (DLYB_CR)*). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT bits in DLYB_CFGR register (see *Section 23.4.2: DLYB configuration register (DLYB_CFGR)*). Note that the UNIT bits can be programmed only when the output clock is disabled (SEN = ‘1’).

When the delay block is enabled, the output clock phase is selected through the SEL bit in DLYB_CFGR register. Note that SEL can be programmed only when the output clock is disabled (SEN = ‘1’).

Before dephasing the output clock, the delay line length shall be configured to one input clock period. The delay line length can be configured by enabling the length sampler through the SEN bit, which gives access to the delay line length (LNG bits) and Length valid flag (LNGF) in DLYB_CFGR.

Once the delay line length has been configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL bits. The output clock is only available on the selected phase when SEN is set to ‘0’.

Table 195. gives a summary of the delay block control.

Table 195. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don’t care	Don’t care	Don’t care	Don’t care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don’t care	Don’t care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = ‘1’.
2. The output clock phase can only be changed when SEN = ‘1’.

23.3.4 Delay line length configuration procedure

LNG bits are used to determine the delay line length with respect to the input clock period. The length shall be configured so that one full input clock period is covered by the delay line length.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to ‘1’.
2. Enable the length sampling by setting SEN bit to ‘1’.
3. Enable all delay cells by setting SEL bits to 12.
4. For UNIT = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT value and wait till the length flag LNGF is set to ‘1’.
   b) Read LNG bits.

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period.
   – For N = 10 to 0:

     If LNG[N] = ‘1’, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to ‘0’.

23.3.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL bits by setting SEN bit to ‘1’.
2. Program SEL bits with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to ‘0’.

23.4 DLYB registers

All registers can be accessed in word, half-word and byte access.

23.4.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SEN:** Sampler length enable bit
0: Sampler length and register access to UNIT and SEL disabled, output clock enabled.
1: Sampler length and register access to UNIT and SEL enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit
0: Delay block disabled.
1: Delay block enabled.

23.4.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LNGF	Res.	Res.	Res.	LNG[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LNGF:** Length valid flag
This flag indicates when the delay line length value contained in LNG bits is valid after UNIT bits changed.
0: Length value in LNG is not valid.
1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value
These bits reflect the 12 unit delay values sampled at the rising edge of the input clock. The value is only valid when LNGF = ‘1’.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay Defines the delay of a Unit delay cell.
These bits can only be written when SEN = '1'.
Unit delay = Initial delay + UNIT x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Select the phase for the Output clock.
These bits can only be written when SEN = '1'.
Output clock phase = Input clock + SEL x Unit delay

23.4.3 DLYB register map

Table 196. DLYB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DLYB_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	**DLYB_CFGR**	LNGF	Res	Res	Res	LNG												Res	UNIT							Res	Res	Res	Res	SEL			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

24 Analog-to-digital converters (ADC)

24.1 Introduction

This section describes the ADC implementation:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).
- ADC3 is instantiated separately.

Each ADC consists of a 16-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

24.2 ADC main features

- High-performance features
  - Up to 2x ADCs which can operate in dual mode
  - 16, 14, 12, 10 or 8-bit configurable resolution
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Can manage Single-ended or differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Self-calibration (both offset and linearity)
  - Channel-wise programmable sampling time
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by GP-DMA for regular channel conversions with FIFO
  - Data can be routed to DFSDM for post processing
  - 4 dedicated data registers for the injected channels
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024x
  - Programmable data right and left shift
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- Each ADC features an external analog input channel
  - Up to 6 fast channels from dedicated GPIO pads
  - Up to 14 slow channels from dedicated GPIO pads
- In addition, there are 5 internal dedicated channels
  - The internal reference voltage (V_{REFINT}), connected to ADC3
  - The internal temperature sensor (V_{SENSE}), connected to ADC3
  - The V_{BAT} monitoring channel ($V_{BAT}/4$), connected to ADC3
  - The internal DAC channel 1 and channel 2, connected to ADC2
- Start-of-conversion can be initiated:
  - by software for both regular and injected conversions
  - by hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels

  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Dual ADC mode for ADC1 and 2
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

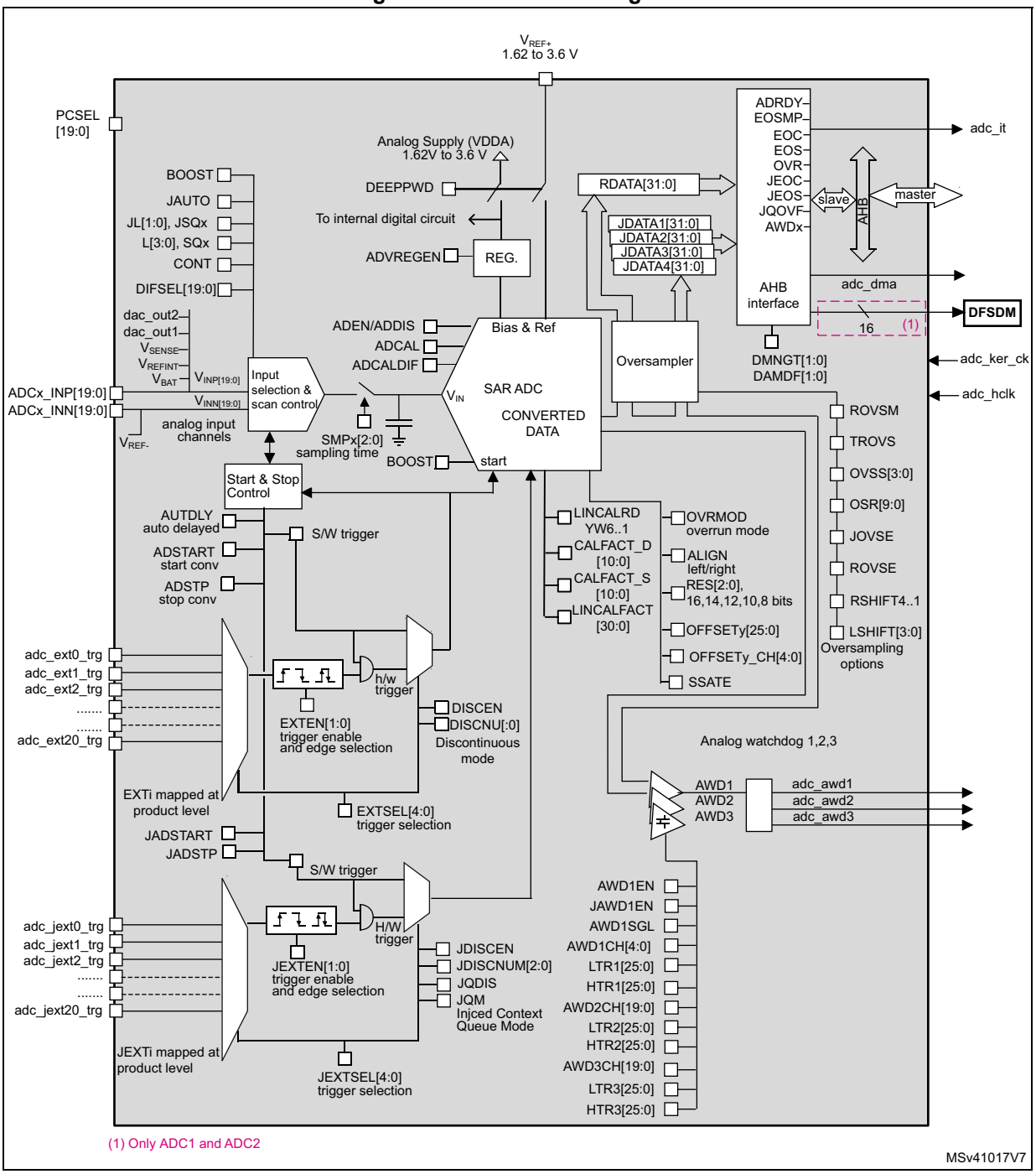
Figure 131 shows the block diagram of one ADC.

24.3 ADC functional description

24.3.1 ADC block diagram

Figure 131 shows the ADC block diagram and *Table 198* gives the ADC pin description.

Figure 131. ADC block diagram



24.3.2 ADC pins and internal signals

Table 197. ADC internal input/output signals

Internal signal name	Signal type	Description
adc_ext_trg[20:0]	Inputs	Up to 21 external trigger inputs for the regular conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
adc_jext_trg[20:0]	Inputs	Up to 21 external trigger inputs for the injected conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
adc_awd1 adc_awd2 adc_awd3	Outputs	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
V_{SENSE}	Analog input	Output voltage from internal temperature sensor
V_{REFINT}	Analog input	Output voltage from internal reference voltage
V_{BAT}	Analog input	External battery voltage supply voltage
adc_it	Output	ADC interrupt
adc_hclk	Input	AHB clock
adc_ker_ck	Input	ADC kernel clock
adc_dma	Output	ADC DMA requests
dac_out1/dac_out2	Analog inputs	DAC channel 1 and channel 2 inputs

Table 198. ADC input/output pins

Name	Signal type	Comments
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, 1.62 V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}	Input, analog supply	Analog power supply equal V_{DDA}: 1.62 V ≤ V_{DDA} ≤ 3.6 V
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, V_{REF-} = V_{SSA}
V_{SSA}	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
V_{INP}[19:0]	Positive input analog channels for each ADC	Connected either to external channels: ADCx_INP*i* or internal channels.
V_{INN}[19:0]	Negative input analog channels for each ADC	Connected to V_{REF-} or external channels: ADCx_INN*i*

Table 198. ADC input/output pins (continued)

Name	Signal type	Comments
ADCx_INP[19:0]	External analog input signals	Up to 20 analog input channels (x = ADC number= 1 to 3): – ADCx_INP[0:5] fast channels – ADCx_INP[6:19] slow channels
ADCx_INN[19:0]		Up to 20 analog input channels (x = ADC number= 1 to 3): – ADCx_INN[0:5] fast channels – ADCx_INN[6:19] slow channels
PCSEL[19:0]	Output, channel preselection control signal	Connected to GPIO to select the channel in advance

24.3.3 Clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADCs clock is independent from the AHB bus clock.

The input clock is the same for the three ADCs and can be selected between two different clock sources (see *Figure 132: ADC clock scheme*):

1. The ADC clock can be a specific clock source, named adc_ker_ck which is independent and asynchronous with the AHB clock.

   It can be configured in the RCC (refer to RCC Section for more information on how to generate the ADC clock (adc_ker_ck) dedicated clock).

   To select this scheme, CKMODE[1:0] bits of the ADCx_CCR register must be reset.

2. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).

   To select this scheme, CKMODE[1:0] bits of the ADCx_CCR register must be different from “00”.

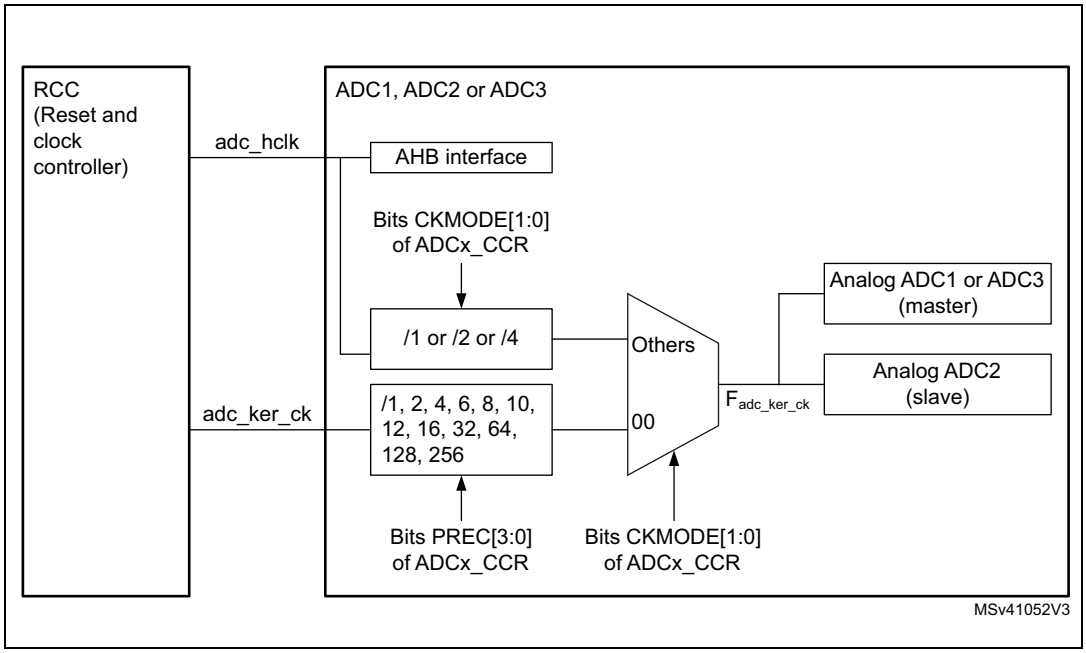
Note: For option 2), a prescaling factor of 1 (CKMODE[1:0]=01) can be used only if the AHB prescaler is set to 1 (HPRE[3:0] = 0xxx in RCC_CFGR register).

Option 1) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits PRESC[3:0] in the ADCx_CCR register.

Option 2) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock configured through CKMODE[1:0] bits must be compliant with the operating frequency specified in the product datasheet.

Figure 132. ADC clock scheme



1. Refer to the RCC section to see how adc_hclk and adc_ker_ck can be generated.

Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{adc_hclk} \geq F_{adc_ker_ck} / 4$ if the resolution of all channels are 16-bit, 14-bit,12-bit or 10-bit
- $F_{adc_hclk} \geq F_{adc_ker_ck} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)

BOOST bit control

There is ADC boost control bit BOOST in the ADC_CR register.

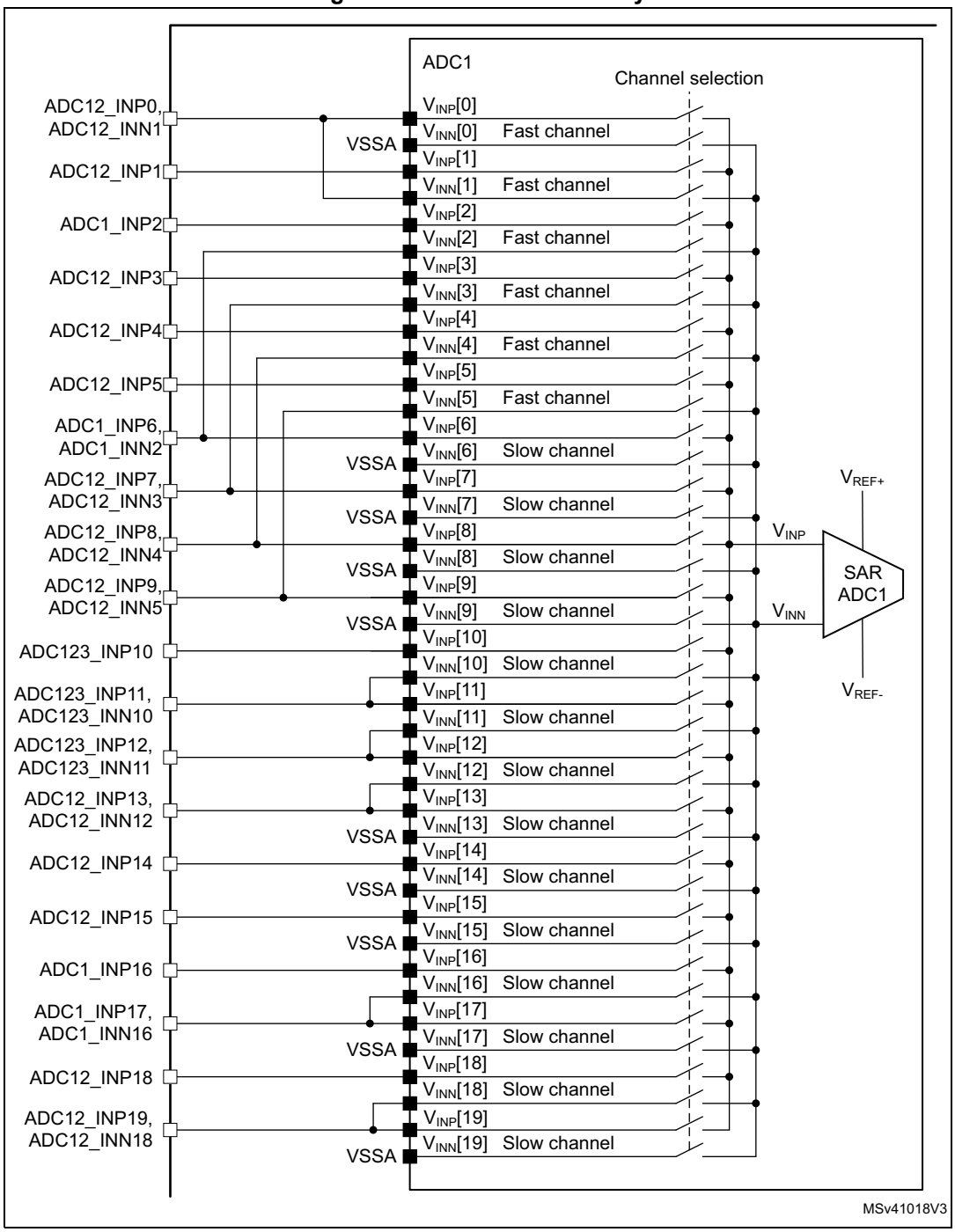
This bit must be set when the ADC clock is more than 20 MHz. When ADC clock is less than 20 MHz, this bit can be cleared to save power.

24.3.4 ADC1/2/3 connectivity

ADC1 and ADC2 are tightly coupled and share some external channels as described in the following figures.

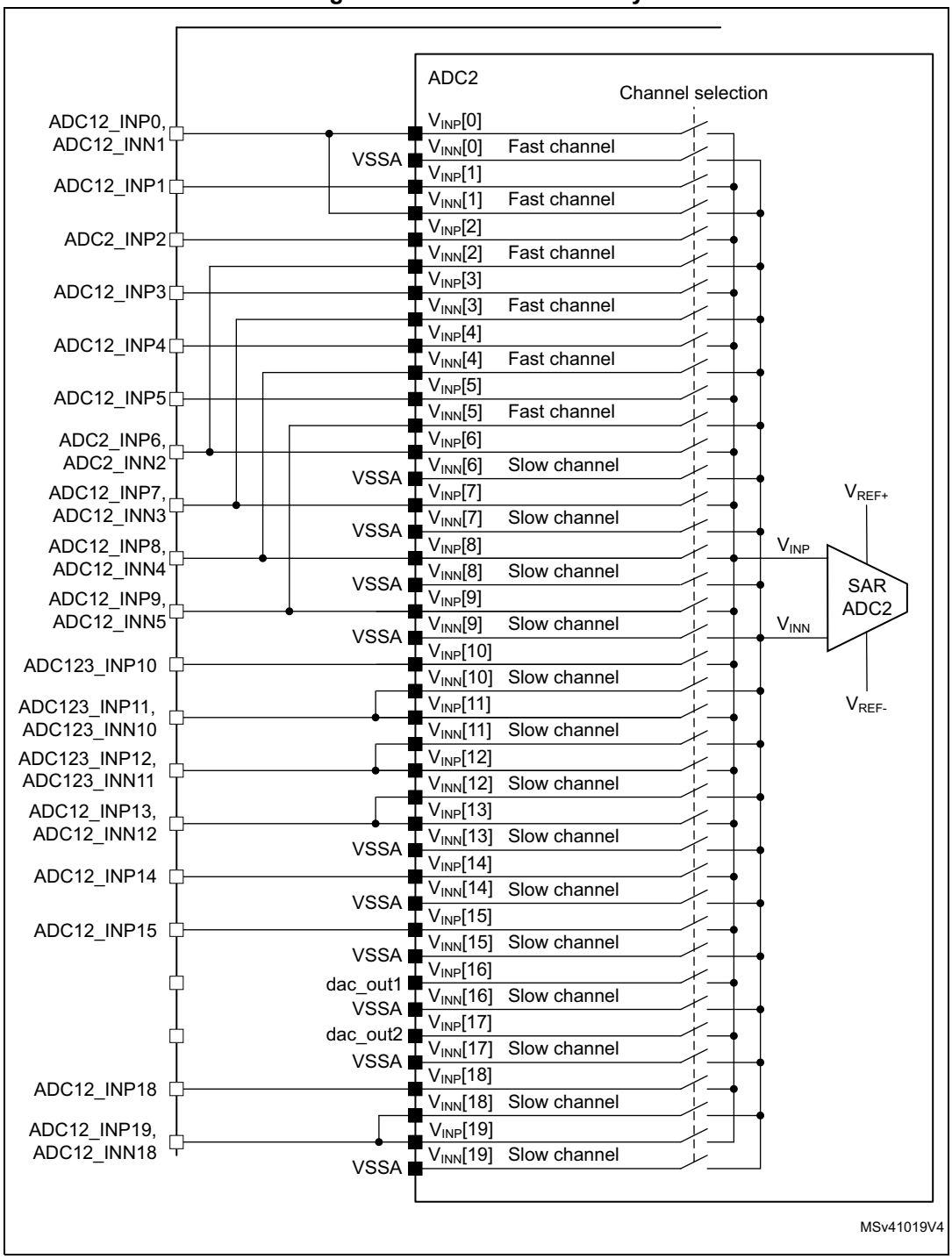
ADC3 is instantiated separately, but some inputs are shared with ADC1 and ADC2.

Figure 133. ADC1 connectivity



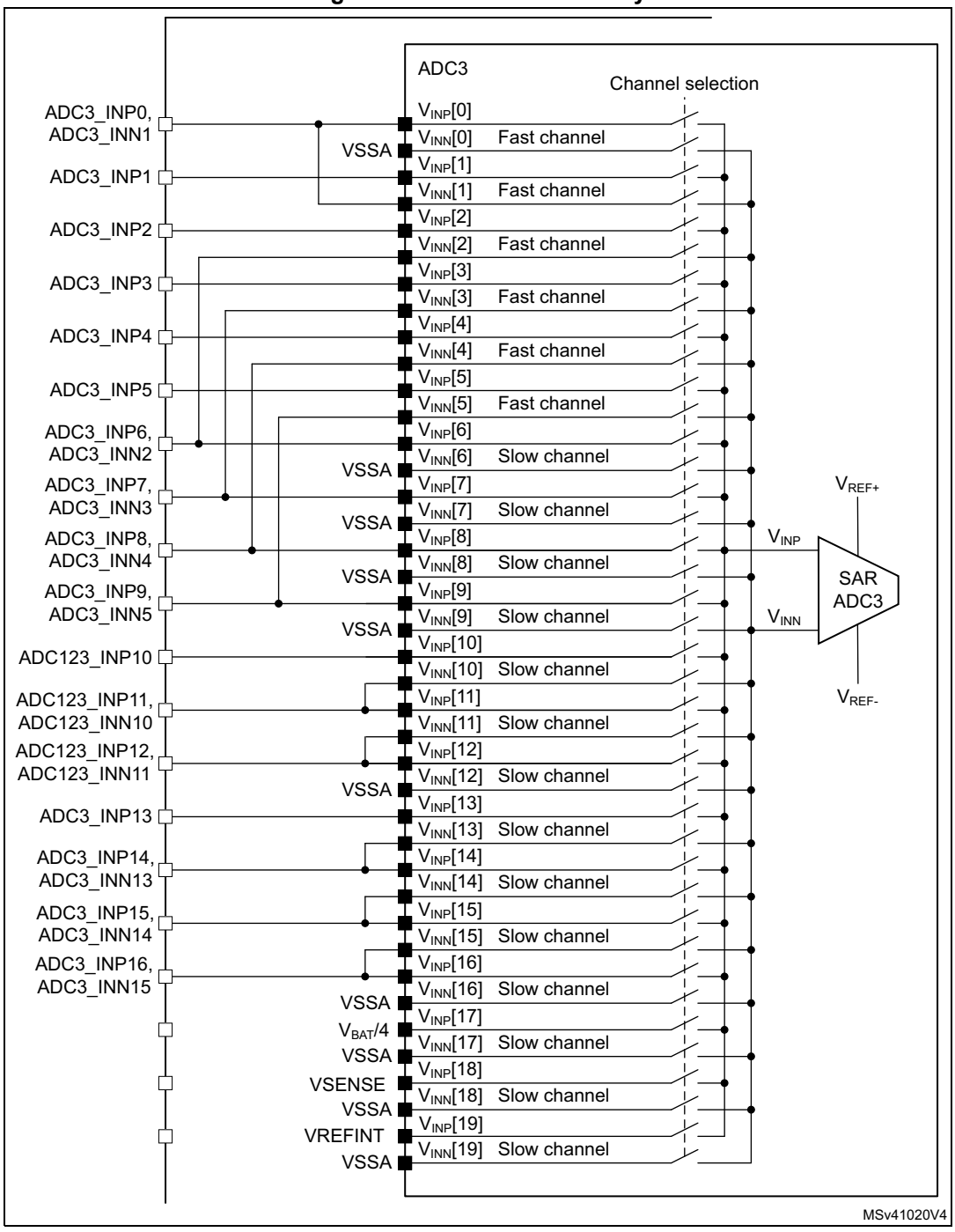
1. ADCx_INNy signal can only be used when the corresponding ADC input channel is configured as differential mode.

Figure 134. ADC2 connectivity



1. ADCx_INNy signal can only be used when the corresponding ADC input channel is configured as differential mode.

Figure 135. ADC3 connectivity



1. ADCx_INNy signal can only be used when the corresponding ADC input channel is configured as differential mode.

24.3.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

24.3.6 ADC deep-power-down mode (DEEPPWD) & ADC voltage regulator (ADVREGEN)

By default, the ADC is in deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, it is first needed to exit deep-power-down mode by clearing bit DEEPPWD=0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN=1 into ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, please refer to device datasheet for $T_{ADCVREG_STUP}$ parameter.

After ADC operations are complete, the ADC can be disabled (ADEN=0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN=0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC deep-power-down mode by setting bit DEEPPWD=1 into ADC_CR register. This is particularly interesting before entering STOP mode.

Note: Writing DEEPPWD=1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.

Note: When the internal voltage regulator is disabled (ADVREGEN=0), the internal analog calibration is kept.

In ADC deep-power-down mode (DEEPPWD=1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to *Section 24.3.8: Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT)*).

24.3.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by writing into bits DIFSEL[19:0] in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN=0).

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and V_{REF-} (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

The output data for the differential mode is an unsigned data. When $V_{INP[i]}$ is VREF-, $V_{INN[i]}$ is VREF+, the output data is 0x0000 (16-bit resolution mode), when $V_{INP[i]}$ is VREF+, $V_{INN[i]}$ is VREF-, the output data is 0xFFFF.

$$\text{Converted value} = \frac{\text{ADC_Full_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When ADC is configured as differential mode, both input should be biased at V_{REF+} / 2 voltage.

The input signal are supposed to be differential (common mode voltage should be fixed).

For a complete description of how the input channels are connected for each ADC, refer to *Figure 133: ADC1 connectivity* to *Figure 135: ADC3 connectivity*.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage is connected to $V_{INN[i]}$. As a consequence, channel “*i+n*”, which is connected to $V_{INN[i]}$, should not be converted at same time by different ADCs. Some channels are shared between ADC1/ADC2: this can make the channel on the other ADC unusable.

24.3.8 Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 11-bits of offset or 160-bits of linearity and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the systematic errors which may vary from chip to chip and allows to compensate offset and linearity deviation.

The calibration factor for the offset to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF=0 before launching a calibration which will be applied for single-ended input conversions.
- Write ADCALDIF=1 before launching a calibration which will be applied for differential input conversions.

The linearity correction must be done once only, regardless of single / differential configuration.

- Write ADCALLIN=1 before launching a calibration which will run the linearity calibration same time as the offset calibration.
- Write ADCALLIN=0 before launching a calibration which will not run the linearity calibration but only the offset calibration.

The calibration is then initiated by software by setting bit ADCAL=1. Calibration can only be initiated when the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[10:0] or CALFACT_D[10:0] of ADC_CALFACT register (depending on single-ended or differential input calibration). The 160-bit linearity calibration factor can be accessed using the ADC_CALFACT2 register with ADEN set to 1.

The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in STANDBY or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT and ADC_CALFACT2 register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

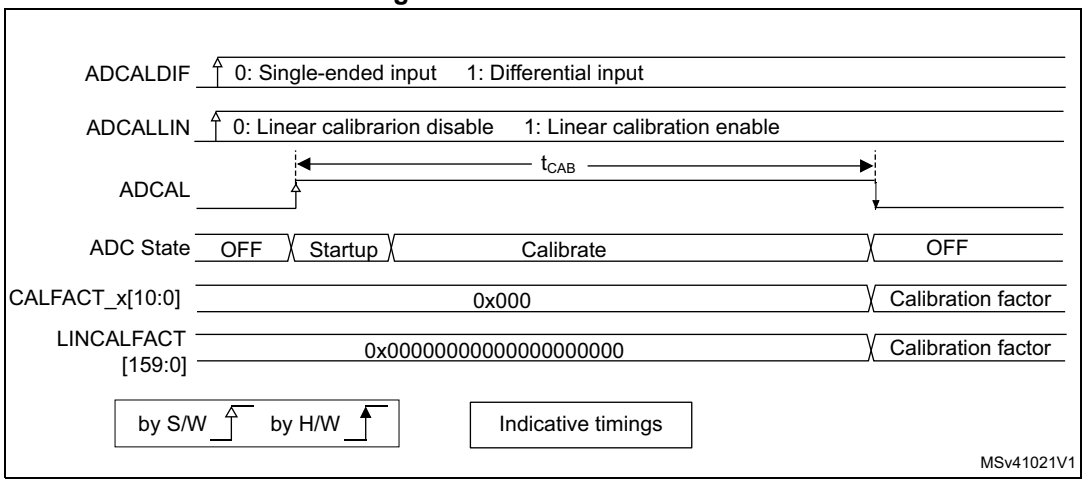
The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0 and JADSTART=0). Then, at the next start of conversion, the calibration factor will automatically be injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when V_{REF+} voltage changed more than 10%.

Refer to the datasheets for the clock cycle requirement for both linear and offset calibration.

Software procedure to calibrate the ADC

1. Ensure DEEPPWD=0, ADVREGEN=1 and check that the ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN=0.
3. Select the input mode for this calibration by setting ADCALDIF=0 (Single-ended input) or ADCALDIF=1 (Differential input). Select if Linearity calibration enable or not by ADCALLIN=1(enabled) or ADCALLIN=0(disabled).
4. Set ADCAL=1.
5. Wait until ADCAL=0.
6. The offset calibration factor can be read from ADC_CALFACT register.
7. The linearity calibration factor can be read from ADC_CALFACT2 register, following the procedure described in *Section : Linearity calibration reading procedure* (ADEN must be set to 1 prior to accessing ADC_CALFACT2 register).

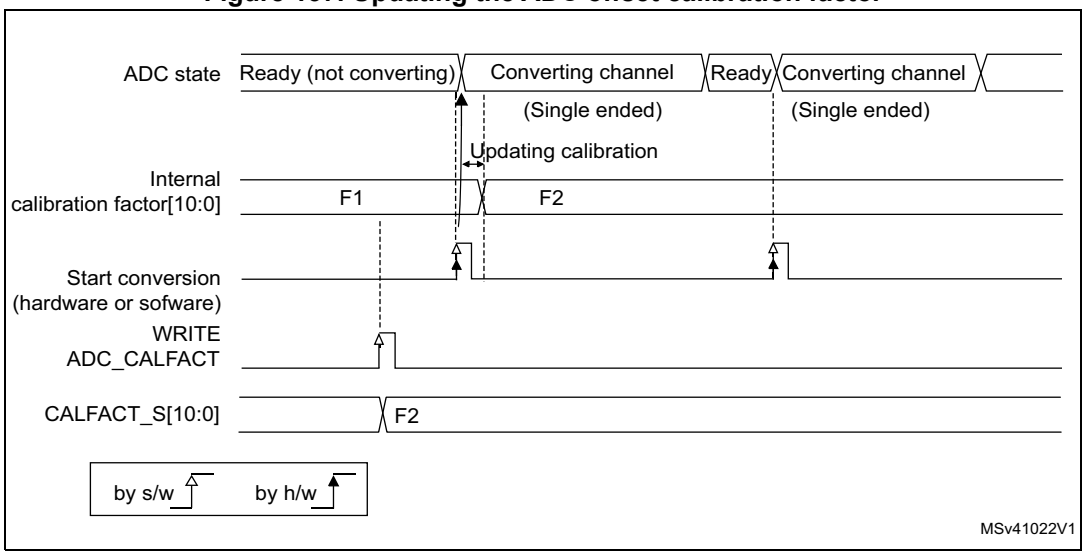
Figure 136. ADC calibration



Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new offset calibration factors.
3. Write LINCALFACT bits with the new linearity calibration factors, following the procedure described in *Section : Linearity calibration writing procedure*.
4. When a conversion is launched, the calibration factor will be injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 137. Updating the ADC offset calibration factor

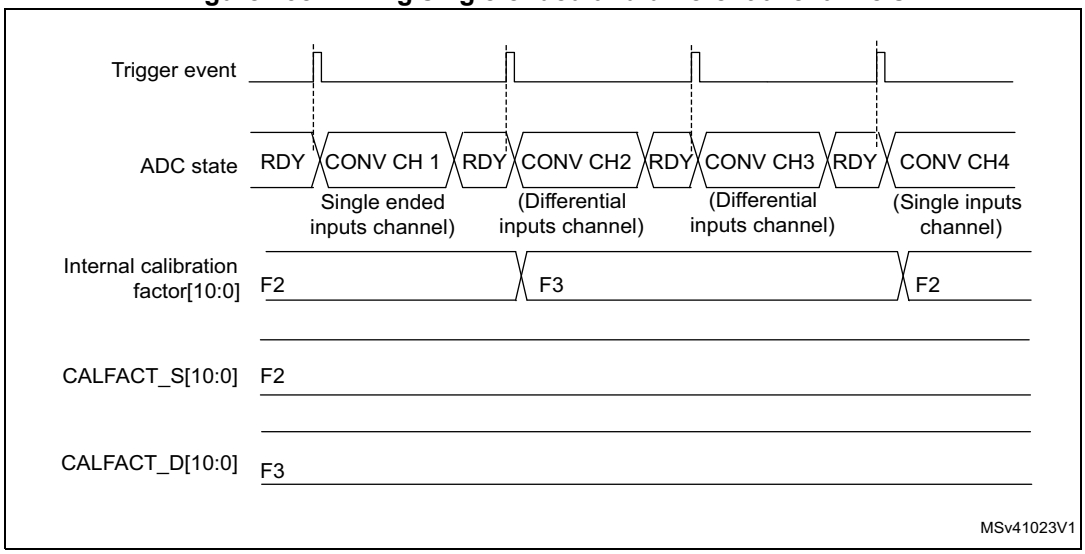


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF=0 and one with ADCALDIF=1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF=0) and Linearity calibration enable (with ADCALLIN=1). This updates the registers CALFACT_S[10:0] and LINCALFACT[159:0].
3. Calibrate the ADC in Differential input modes (with ADCALDIF=1) and Linearity calibration disable (with ADCALLIN=0). This updates the register CALFACT_D[10:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration will automatically be injected into the analog ADC.

Figure 138. Mixing single-ended and differential channels



Linearity calibration reading procedure

Once the calibration is done (ADCAL bit cleared by hardware) with ADCALLIN=1, the 160-bit linearity correction factor can be read using the ADC_CALFACT2 30-bit registers (6 read accesses are necessary).

The six LINCALRDYW1..6 control/status bits in ADC_CR are set when the calibration is complete. When ADEN is set to 1, clearing one of these bits launches the transfer of part of the linearity factor into the LINCALFACT[29:0] of the ADC_CALFACT2 register. The bit will be reset by hardware when the ADC_CALFACT2 register can be read (software must poll the bit until it is cleared). The complete procedure is as following:

1. Ensure DEEPPWD=0, ADVREGEN=1 and that the ADC voltage regulator startup time has elapsed.
2. Set ADEN = 1 and wait until ADRDY=1.
3. Clear LINCALRDYW6 bit (Linearity calibration ready Word 6).
4. Poll LINCALRDYW6 bit until returned value is zero, indicating linearity correction bits[159:150] are available in ADC_CALFACT2[29:0].
5. Read ADC_CALFACT2[29:0].
6. Clear LINCALRDYW5 bit.
7. Poll LINCALRDYW5 bit until returned value is zero, indicating linearity correction bits[149:120] are available in ADC_CALFACT2[29:0].
8. Read ADC_CALFACT2[29:0].
9. Clear LINCALRDYW4 bit.
10. Poll LINCALRDYW4 bit until returned value is zero, indicating linearity correction bits[119:90] are available in ADC_CALFACT2[29:0].
11. Read ADC_CALFACT2[29:0].
12. Clear LINCALRDYW3 bit.
13. Poll LINCALRDYW3 bit until returned value is zero, indicating linearity correction bits[89:60] are available in ADC_CALFACT2[29:0].
14. Read ADC_CALFACT2[29:0].
15. Clear LINCALRDYW2 bit.
16. Poll LINCALRDYW2 bit until returned value is zero, indicating linearity correction bits[59:30] are available in ADC_CALFACT2[29:0].
17. Read ADC_CALFACT2[29:0].
18. Clear LINCALRDYW1 bit.
19. Poll LINCALRDYW1 bit until returned value is zero, indicating linearity correction bits[29:0] are available in ADC_CALFACT2[29:0].
20. Read ADC_CALFACT2[29:0].

Note: The software is allowed to toggle a single LINCALRDYWx bit at once (other bits left unchanged), otherwise causing unexpected behavior.

The software can access the linearity calibration factor by writing LINCALRDYW1..6 bits only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).

Linearity calibration writing procedure

The six LINCALRDYW1..6 control/status bits in ADC_CR are reset when the calibration has not yet been done or a new linearity calibration factor have been rewritten. It is possible to force directly a linearity calibration factor or re-inject it using the following procedure:

1. Ensure DEEPPWD=0, ADVREGEN=1 and that ADC voltage regulator startup time has elapsed.
2. Set ADEN = 1 and wait until ADRDY=1.
3. Write ADC_CALFACT2[9:0] with previously saved linearity correction factor bits[159:150].
4. Set LINCALRDYW6 bit.
5. Poll LINCALRDYW6 bit until returned value is one, indicating linearity correction bits[159:150] have been effectively written.
6. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[149:120].
7. Set LINCALRDYW5 bit.
8. Poll LINCALRDYW5 bit until returned value is one, indicating linearity correction bits[149:120] have been effectively written.
9. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[119:90].
10. Set LINCALRDYW4 bit.
11. Poll LINCALRDYW4 bit until returned value is one, indicating linearity correction bits[119:90] have been effectively written.
12. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[89:60].
13. Set LINCALRDYW3 bit.
14. Poll LINCALRDYW3 bit until returned value is one, indicating linearity correction bits[89:60] have been effectively written.
15. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[59:30].
16. Set LINCALRDYW2 bit.
17. Poll LINCALRDYW2 bit until returned value is one, indicating linearity correction bits[59:30] have been effectively written.
18. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[29:0].
19. Set LINCALRDYW1 bit.
20. Poll LINCALRDYW1 bit until returned value is one, indicating linearity correction bits[29:0] have been effectively written.

Note: The software is allowed to toggle a single LINCALRDYWx bit at once (other bits left unchanged), otherwise causing unexpected behavior.

The software is allowed to update the linearity calibration factor by writing LINCALRDYW1..6 bits only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).

24.3.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 24.3.6: ADC deep-power-down mode (DEEPPWD) & ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD=0 and ADVREGEN=1, the ADC can be enabled and the ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in *Figure 139*. Two control bits enable or disable the ADC:

- ADEN=1 enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
- ADDIS=1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART=1 (refer to *Section 24.3.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART=1 or when an external injected trigger event occurs, if injected triggers are enabled.

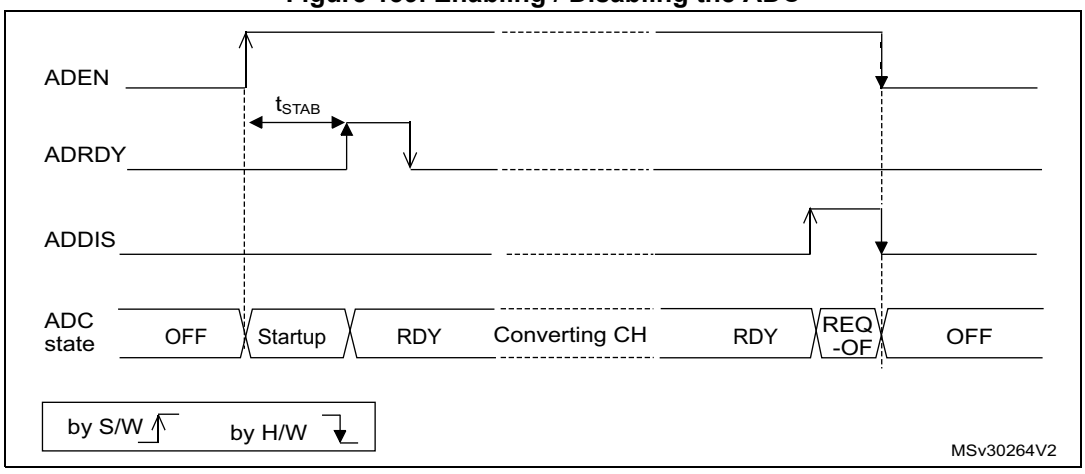
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN=1.
3. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

Software procedure to disable the ADC

1. Check that both ADSTART=0 and JADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP=1 and JADSTP=1 and then wait until ADSTP=0 and JADSTP=0.
2. Set ADDIS=1.
3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS will automatically be reset once ADEN=0).

Figure 139. Enabling / Disabling the ADC



24.3.10 Constraints when writing the ADC control bits

The software can write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the control bits DIFSEL in the ADC_DIFSEL register, ADCx_CCR register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRy, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN=1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software can write ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and eventually converting and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN=1).

The software is allowed to write the ADC_JSQR register only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN=0 as well as all the bits of ADC_CR register).

24.3.11 Channel selection (SQRx, JSQRx)

There are up to 20 multiplexed channels per ADC:

- 6 fast analog inputs coming from Analog PADs and GPIO pads (ADCx_INP/INN[0..5])
- Up to 14 slow analog inputs coming from GPIO pads (ADCx_INP/INN[6..19]).
- The ADCs are connected to 5 internal analog inputs:
  - the internal temperature sensor (V_{SENSE}) is connected to ADC3_INP/INN18
  - the internal reference voltage (V_{REFINT}) is connected to ADC3_INP/INN19
  - the V_{BAT} monitoring channel ($V_{BAT}/4$) is connected to ADC3_INP/INN17
  - DAC internal channel 1, connected to ADC2_INP/INN16

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INP/INN3, ADCx_INP/INN8, ADCx_INP/INN2, ADCx_INP/INN2, ADCx_INP/INN0, ADCx_INP/INN2, ADCx_INP/INN2, ADCx_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The

total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.

- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP=1 (refer to *Section 24.3.18: Stopping an ongoing conversion (ADSTP, JADSTP)*).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 only when the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Temperature sensor, V_{REFINT} and V_{BAT} internal channels

The temperature sensor V_{SENSE} is connected to channel ADC3 V_{INP}[18].

The internal reference voltage V_{REFINT} is connected to ADC3 V_{INP}[19].

The V_{BAT} channel is connected to channel ADC3 V_{INP}[17].

Note: To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, VSENSEEN or VBATEN in the ADCx_CCR registers.

24.3.12 Channel preselection register (ADC_PCSEL)

For each channel selected through SQRx or JSQRx, the corresponding ADC_PCSEL bit must be previously configured.

This ADC_PCSEL bit controls the analog switch integrated in the IO level. The ADC input MUX selects the ADC input according to the SQRx and JSQRx with very high speed, the analog switch integrated in the IO cannot react as fast as ADC mux do. To avoid the delay on analog switch control on IO, it is necessary to pre select the input channels which will be selected in the SQRx, JSQRx.

The selection is based on the $V_{INP[i]}$ of the each ADC input. If ADC1 will convert the ADC123_INP2($V_{INP[2]}$) as differential mode, ADC123_INP6($V_{INP[6]}$) also need to be selected in ADC_PCSEL.

Some I/Os are connected to several $V_{INP[i]}$ of the ADCx. The control inputs of the analog switch are ORed with the corresponding ADC_PCSEL register bits.

24.3.13 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 8.5 ADC clock cycles
- SMP = 011: 16.5 ADC clock cycles
- SMP = 100: 32.5 ADC clock cycles
- SMP = 101: 64.5 ADC clock cycles
- SMP = 110: 387.5 ADC clock cycles
- SMP = 111: 810.5 ADC clock cycles

The total conversion time is calculated as follows:

T_{CONV} = Sampling time + 7.5 ADC clock cycles

Example:

With $F_{adc_ker_ck}$ = 24 MHz and a sampling time of 1.5 ADC clock cycles (14-bit mode):

T_{CONV} = (1.5 + 7.5) ADC clock cycles = 9 ADC clock cycles = 0.375 µs (14 bit mode for fast channels)

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time for fast and slow channels

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

I/O analog switches voltage booster

The I/O analog switches resistance increases when the V_{DDA} voltage is too low. This requires to have the sampling time adapted accordingly (cf datasheet for electrical characteristics). This resistance can be minimized at low V_{DDA} by enabling an internal voltage booster with BOOSTE bit in the SYSCFG_PMCR register.

24.3.14 Single conversion mode (CONT=0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel, with software trigger selected)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel, with software trigger selected)
- External hardware trigger event (for a regular or injected channel)

  ADSTART bit or JADSTART bit must be set before triggering an external event.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

24.3.15 Continuous conversion mode (CONT=1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically re-starts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section).

24.3.16 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART=1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN /= 0x0

Software starts ADC injected conversions by setting JADSTART=1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN /= 0x0

Note: In auto-injection mode (JAUTO=1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0 and JADSTART=0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software trigger (CONT=0, EXTEN=0x0)
  - at any end of conversion sequence (EOS =1)
- In discontinuous mode with software trigger (CONT=0, DISCEN=1, EXTEN=0x0)
  - at end of conversion (EOC=1)
- In all other cases (CONT=x, EXTEN=x)
  - after execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT=1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT=0 and EXTEN /=0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- in single mode with software injected trigger (JEXTEN=0x0)
  - at any end of injected conversion sequence (JEOS assertion) or at any end of sub-group processing if JDISCEN=1
- in all cases (JEXTEN=x)
  - after execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.

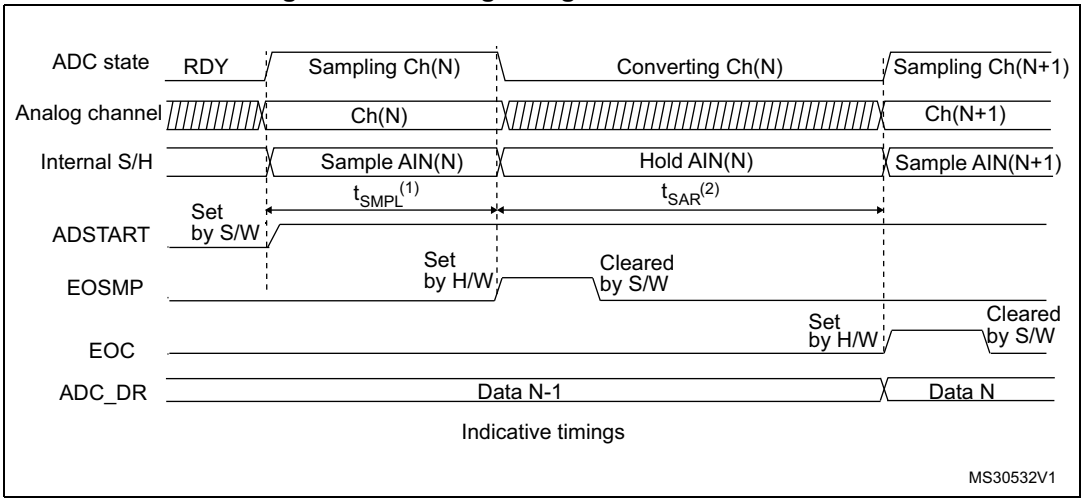
24.3.17 Timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$T_{CONV} = T_{SMPL} + T_{SAR} = [1.5\ |_{min} + 7.5\ |_{14bit}] \times T_{adc_ker_ck}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} = 62.5\ ns\ |_{min} + 312.5\ ns\ |_{14bit} = 375.0\ ns\ (for\ F_{adc_ker_ck} = 24\ MHz)$$

Figure 140. Analog to digital conversion time



1. T_{SMPL} depends on SMP[2:0]
2. T_{SAR} depends on RES[2:0]

24.3.18 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP=1 and injected conversions ongoing by setting JADSTP=1.

Stopping conversions will reset the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would re-start a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 141. Stopping ongoing regular conversions

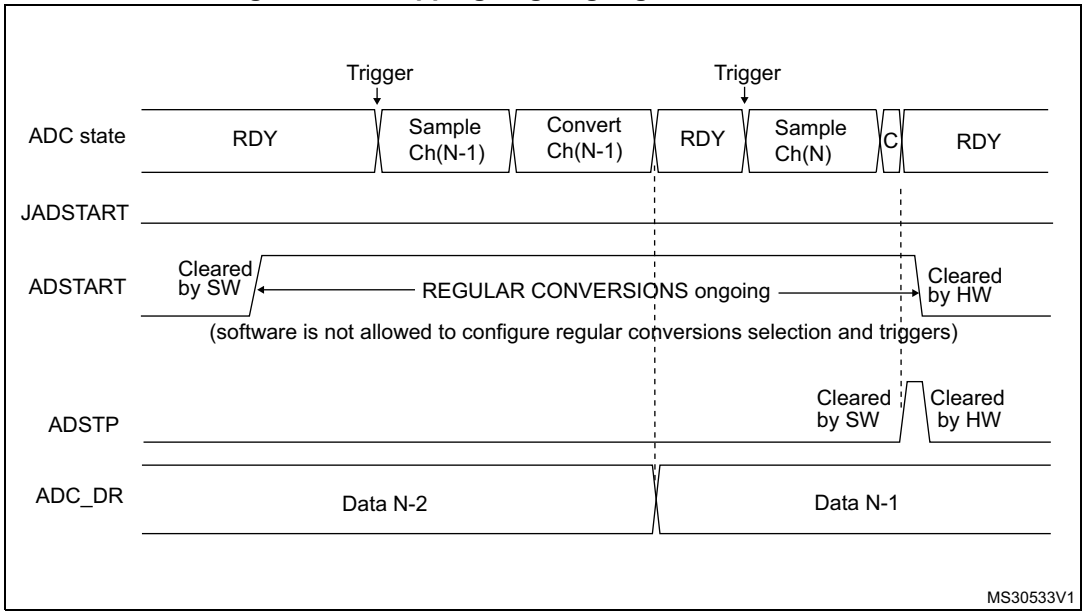
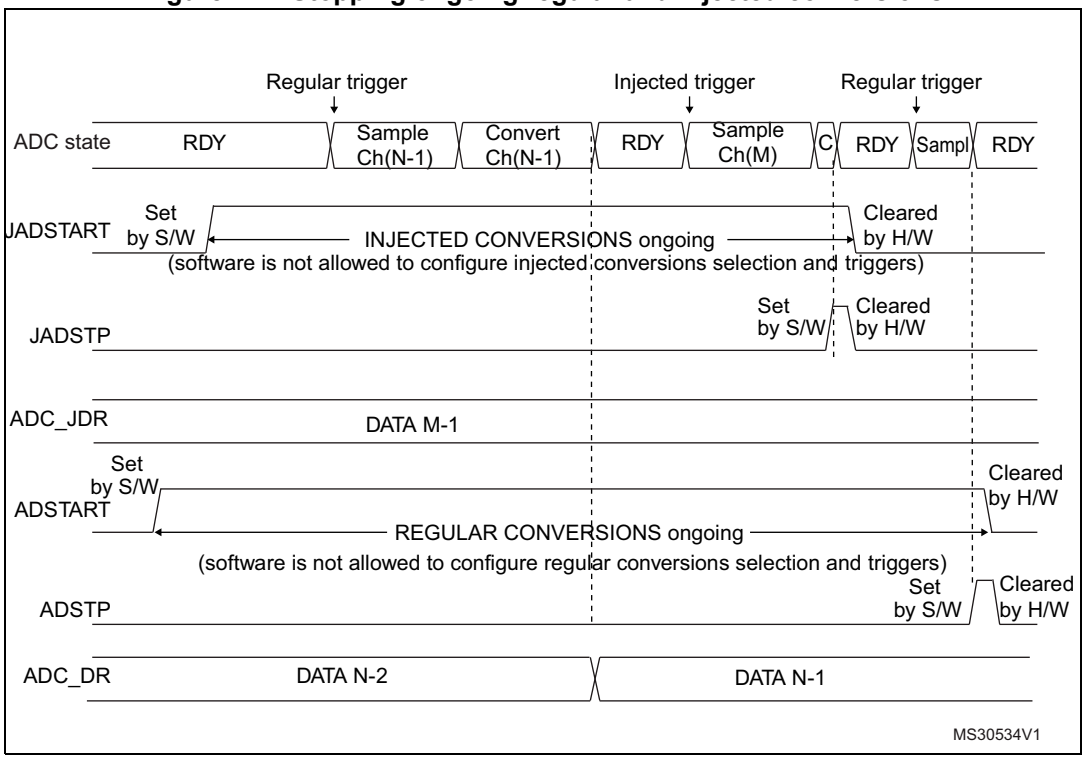


Figure 142. Stopping ongoing regular and injected conversions



24.3.19 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS=0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART=1 and the injected trigger selection is effective once software has set bit JADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART=0, any regular hardware triggers which occur are ignored.
- If bit JADSTART=0, any injected hardware triggers which occur are ignored.

Table 199 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 199. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 200. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	– If JQDIS=1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled – If JQDIS=0 (Queue enabled), Hardware and software trigger detection disabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS=0). Refer to Section 24.3.22: Queue of context for injected conversions.

The EXTSEL[4:0] and JEXTSEL[4:0] control bits select which out of 21 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly.
The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 24.3.22: Queue of context for injected conversions on page 925

Each ADC master shares the same input triggers with its ADC slave as described in *Figure 143*.

Figure 143. Triggers are shared between ADC master and ADC slave

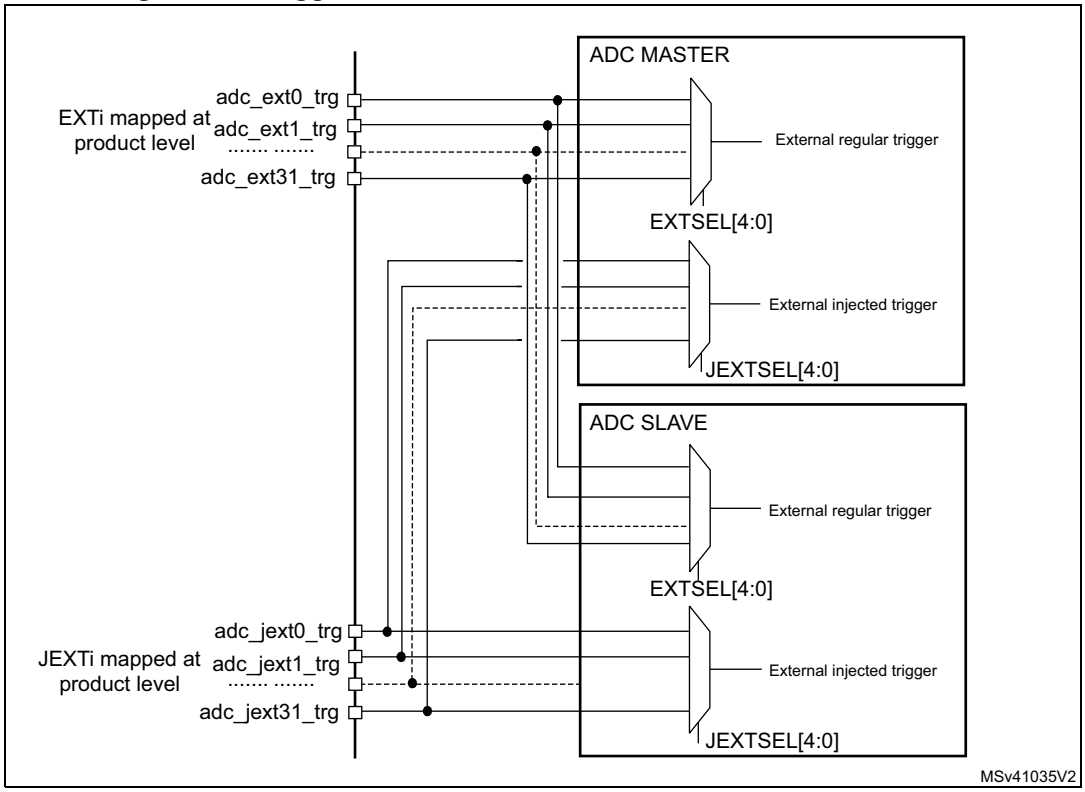


Table 201 and *Table 202* give all the possible external triggers of the three ADCs for regular and injected conversion.

Table 201. ADC1, ADC2 and ADC3 - External triggers for regular channels

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg0	TIM1_CC1 event	Internal signal from on-chip timers	00000
adc_ext_trg1	TIM1_CC2 event	Internal signal from on-chip timers	00001
adc_ext_trg2	TIM1_CC3 event	Internal signal from on-chip timers	00010
adc_ext_trg3	TIM2_CC2 event	Internal signal from on-chip timers	00011
adc_ext_trg4	TIM3_TRGO event	Internal signal from on-chip timers	00100
adc_ext_trg5	TIM4_CC4 event	Internal signal from on-chip timers	00101
adc_ext_trg6	EXTI line 11	External pin	00110
adc_ext_trg7	TIM8_TRGO event	Internal signal from on-chip timers	00111
adc_ext_trg8	TIM8_TRGO2 event	Internal signal from on-chip timers	01000

Table 201. ADC1, ADC2 and ADC3 - External triggers for regular channels (continued)

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg9	TIM1_TRGO event	Internal signal from on-chip timers	01001
adc_ext_trg10	TIM1_TRGO2 event	Internal signal from on-chip timers	01010
adc_ext_trg11	TIM2_TRGO event	Internal signal from on-chip timers	01011
adc_ext_trg12	TIM4_TRGO event	Internal signal from on-chip timers	01100
adc_ext_trg13	TIM6_TRGO event	Internal signal from on-chip timers	01101
adc_ext_trg14	TIM15_TRGO event	Internal signal from on-chip timers	01110
adc_ext_trg15	TIM3_CC4 event	Internal signal from on-chip timers	01111
adc_ext_trg16	HRTIM1_ADCTRG1 event	Internal signal from on-chip timers	10000
adc_ext_trg17	HRTIM1_ADCTRG3 event	Internal signal from on-chip timers	10001
adc_ext_trg18	LPTIM1_OUT event	Internal signal from on-chip timers	10010
adc_ext_trg19	LPTIM2_OUT event	Internal signal from on-chip timers	10011
adc_ext_trg20	LPTIM3_OUT event	Internal signal from on-chip timers	10100
adc_ext_trg21	Reserved	-	10101
adc_ext_trg22	Reserved	-	10110
adc_ext_trg23	Reserved	-	10111
adc_ext_trg24	Reserved	-	11000
adc_ext_trg25	Reserved	-	11001
adc_ext_trg26	Reserved	-	11010
adc_ext_trg27	Reserved	-	11011
adc_ext_trg28	Reserved	-	11100
adc_ext_trg29	Reserved	-	11101
adc_ext_trg30	Reserved	-	11110
adc_ext_trg31	Reserved	-	11111

Table 202. ADC1, ADC2 and ADC3 - External triggers for injected channels

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg0	TIM1_TRGO event	Internal signal from on-chip timers	00000
adc_jext_trg1	TIM1_CC4 event	Internal signal from on-chip timers	00001
adc_jext_trg2	TIM2_TRGO event	Internal signal from on-chip timers	00010
adc_jext_trg3	TIM2_CC1 event	Internal signal from on-chip timers	00011
adc_jext_trg4	TIM3_CC4 event	Internal signal from on-chip timers	00100
adc_jext_trg5	TIM4_TRGO event	Internal signal from on-chip timers	00101
adc_jext_trg6	EXTI line 15	External pin	00110
adc_jext_trg7	TIM8_CC4 event	Internal signal from on-chip timers	00111
adc_jext_trg8	TIM1_TRGO2 event	Internal signal from on-chip timers	01000

Table 202. ADC1, ADC2 and ADC3 - External triggers for injected channels (continued)

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg9	TIM8_TRGO event	Internal signal from on-chip timers	01001
adc_jext_trg10	TIM8_TRGO2 event	Internal signal from on-chip timers	01010
adc_jext_trg11	TIM3_CC3 event	Internal signal from on-chip timers	01011
adc_jext_trg12	TIM3_TRGO event	Internal signal from on-chip timers	01100
adc_jext_trg13	TIM3_CC1 event	Internal signal from on-chip timers	01101
adc_jext_trg14	TIM6_TRGO event	Internal signal from on-chip timers	01110
adc_jext_trg15	TIM15_TRGO event	Internal signal from on-chip timers	01111
adc_jext_trg16	HRTIM1_ADCTRG2 event	Internal signal from on-chip timers	10000
adc_jext_trg17	HRTIM1_ADCTRG4 event	Internal signal from on-chip timers	10001
adc_jext_trg18	LPTIM1_OUT event	Internal signal from on-chip timers	10010
adc_jext_trg19	LPTIM2_OUT event	Internal signal from on-chip timers	10011
adc_jext_trg20	LPTIM3_OUT event	Internal signal from on-chip timers	10100
adc_jext_trg21	Reserved	-	10101
adc_jext_trg22	Reserved	-	10110
adc_jext_trg23	Reserved	-	10111
adc_jext_trg24	Reserved	-	11000
adc_jext_trg25	Reserved	-	11001
adc_jext_trg26	Reserved	-	11010
adc_jext_trg27	Reserved	-	11011
adc_jext_trg28	Reserved	-	11100
adc_jext_trg29	Reserved	-	11101
adc_jext_trg30	Reserved	-	11110
adc_jext_trg31	Reserved	-	11111

24.3.20 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is

reset and the injected channel sequence switches are launched (all the injected channels are converted once).

3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 144* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 20 ADC clock cycles (that is two conversions with a sampling time of 1.5 clock periods), the minimum interval between triggers must be 21 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

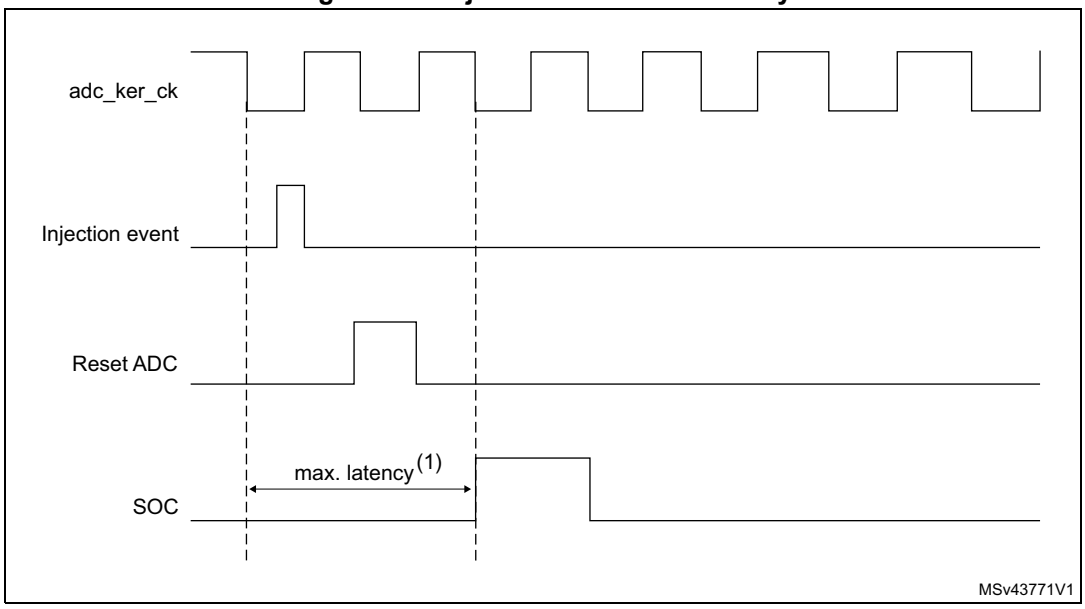
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence will be stopped upon DMA Transfer Complete event.

Figure 144. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

24.3.21 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (sub-group) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQR registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN=1, n=3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN=0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - all the next trigger events will relaunch the complete sequence.

Note: When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN=1, CONT=1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN=1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

24.3.22 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL[4:0] in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)

All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence will be served according to the last active context.
  - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP=1 or when disabling the ADC by setting ADDIS=1:
  - If JQM=0, the Queue is maintained with the last active context.
  - If JQM=1, the Queue becomes empty and triggers are ignored.

Note: When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

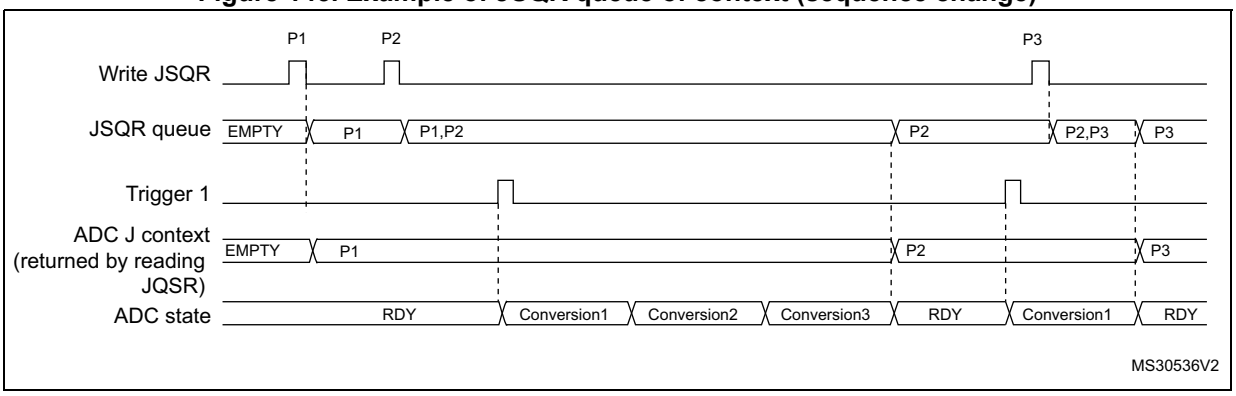
- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

Note: When queue of context enabled (bit JQDIS=0), only hardware trigger can be used.

Behavior when changing the trigger or sequence context

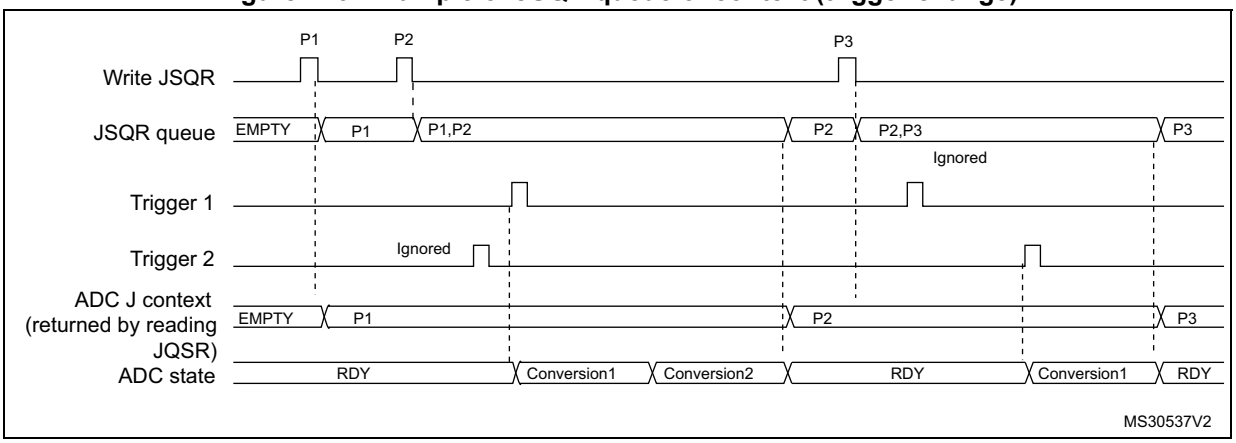
The *Figure 145* and *Figure 146* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 145. Example of JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 146. Example of JSQR queue of context (trigger change)

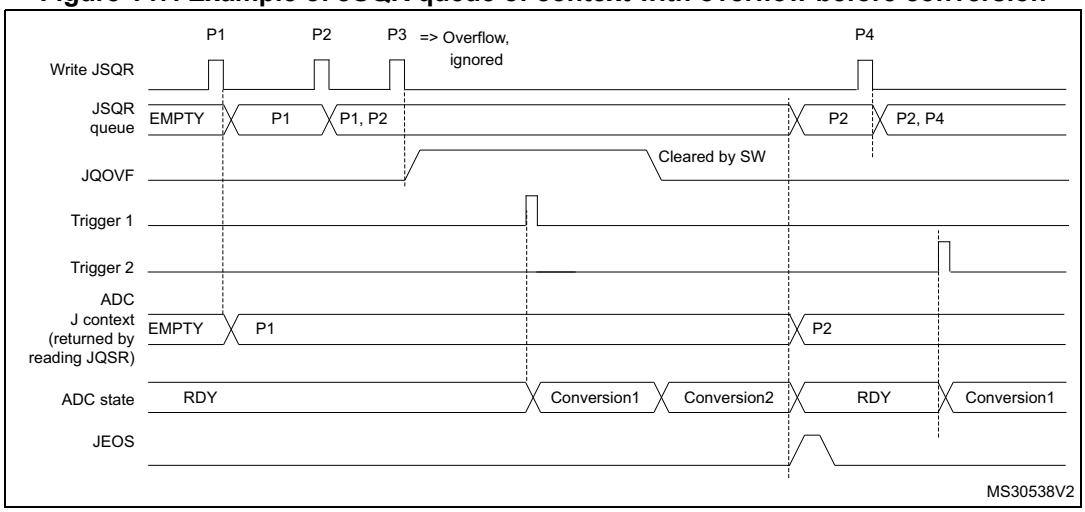


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

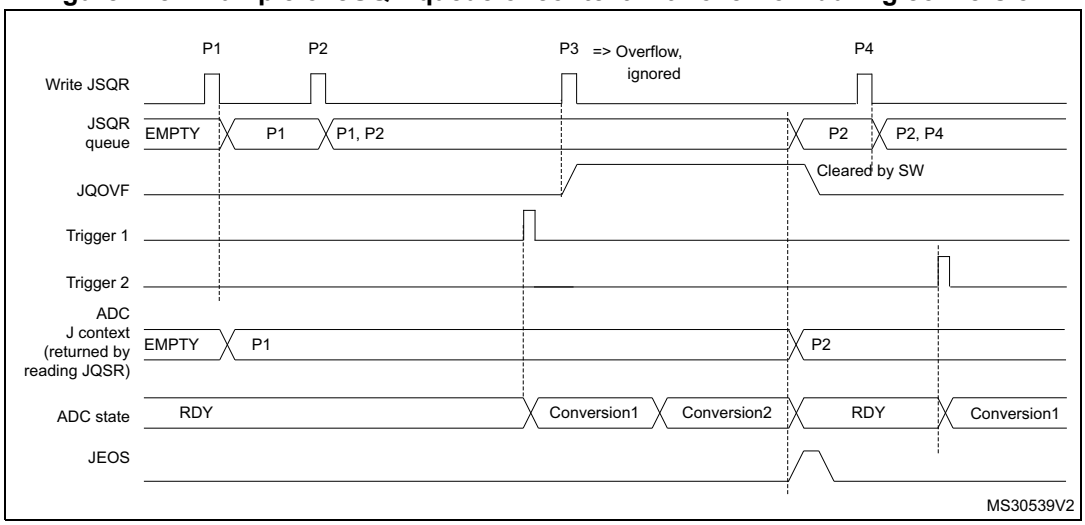
The *Figure 147* and *Figure 148* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 147. Example of JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 148. Example of JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

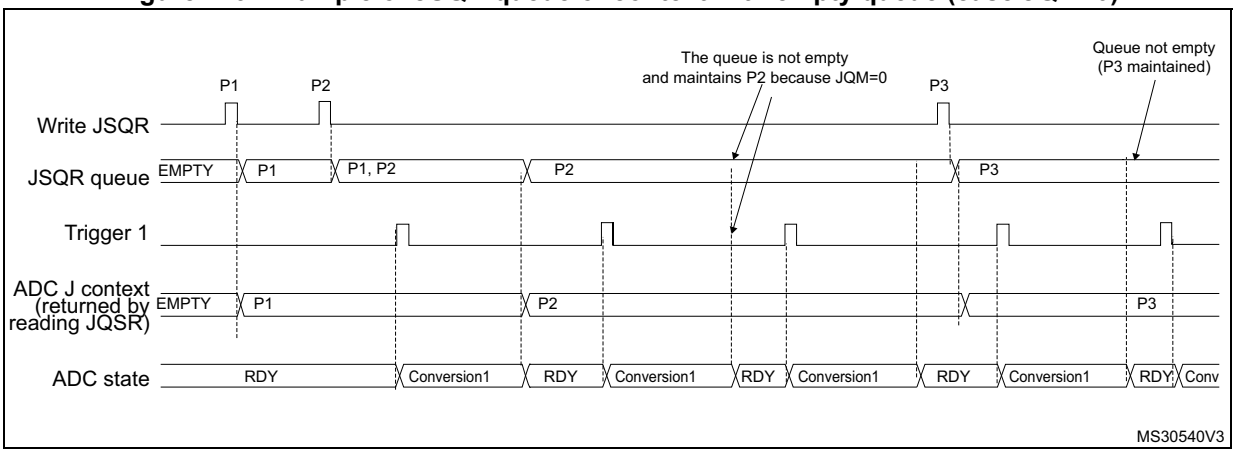
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 149 and *Figure 150* show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

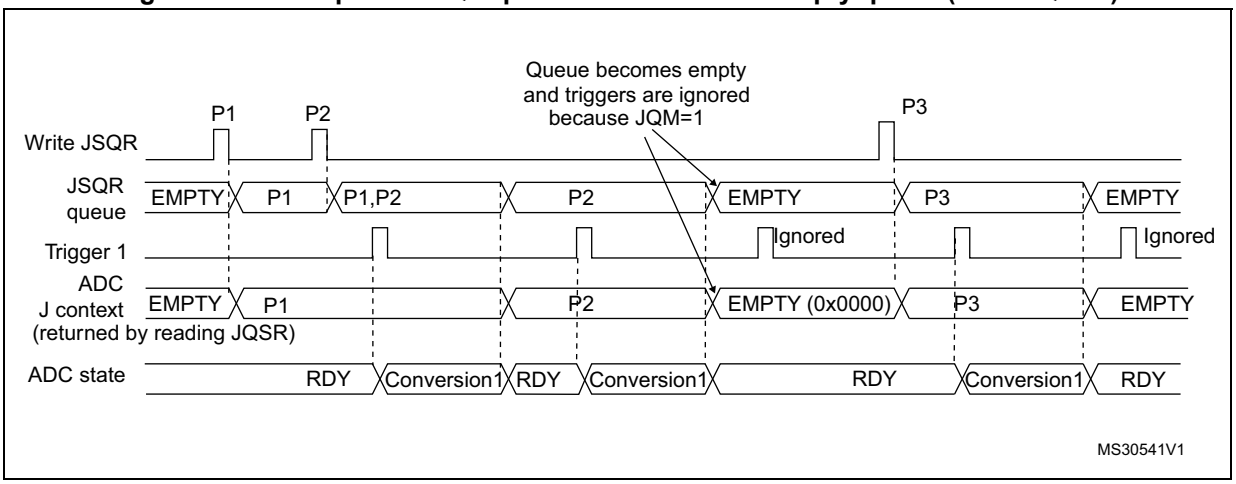
Figure 149. Example of JSQR queue of context with empty queue (case JQM=0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 150. Example of JSQR queue of context with empty queue (case JQM=1)

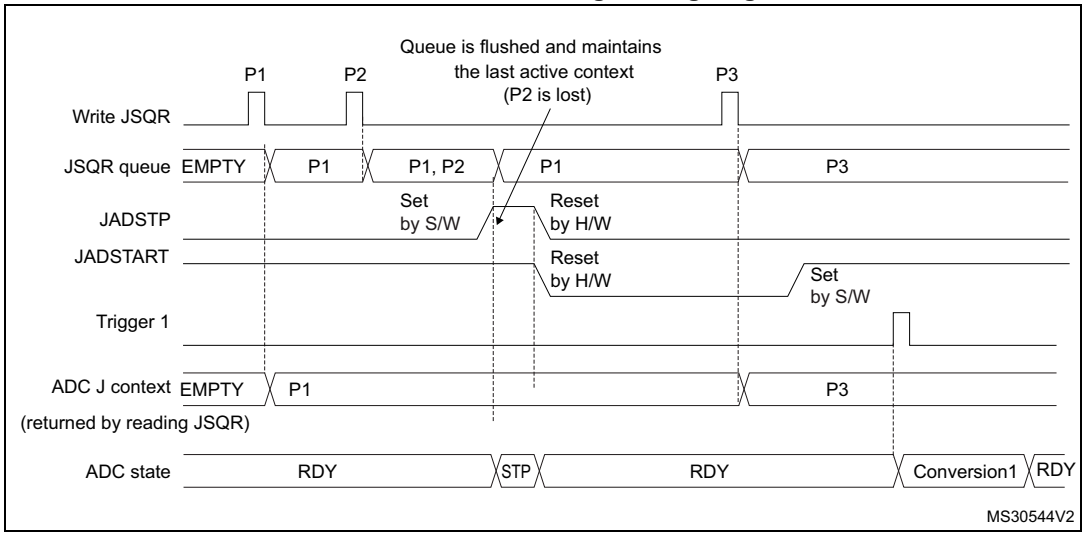


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

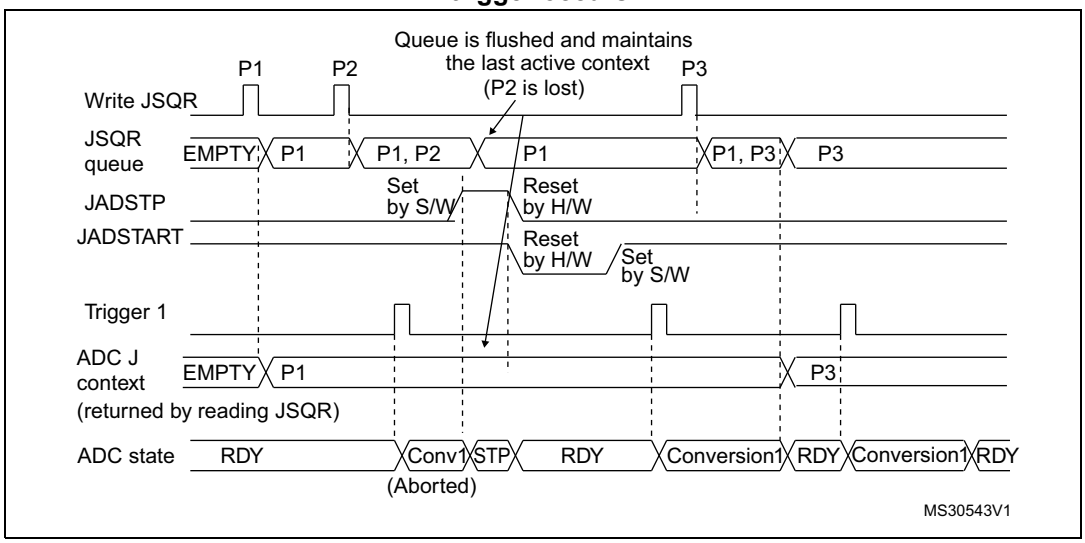
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 151. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.



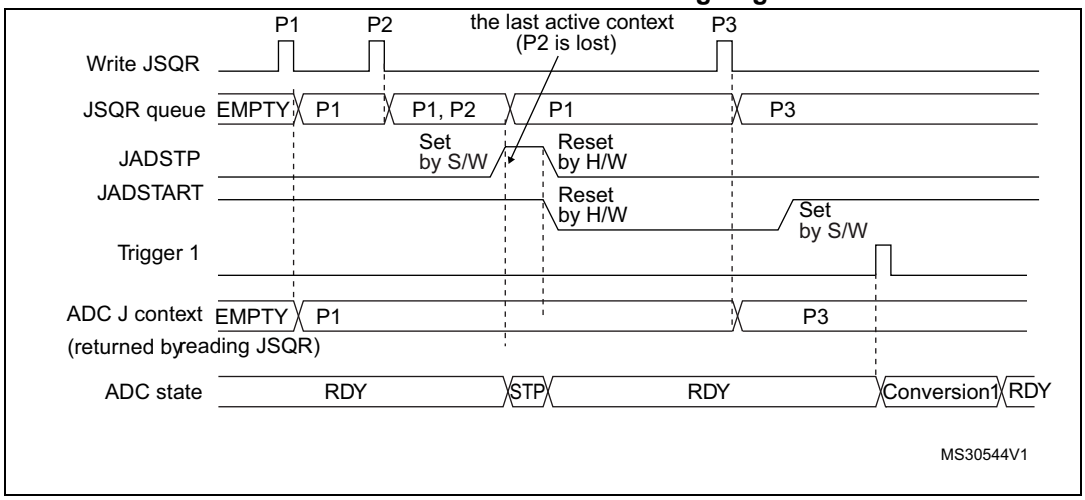
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 152. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.



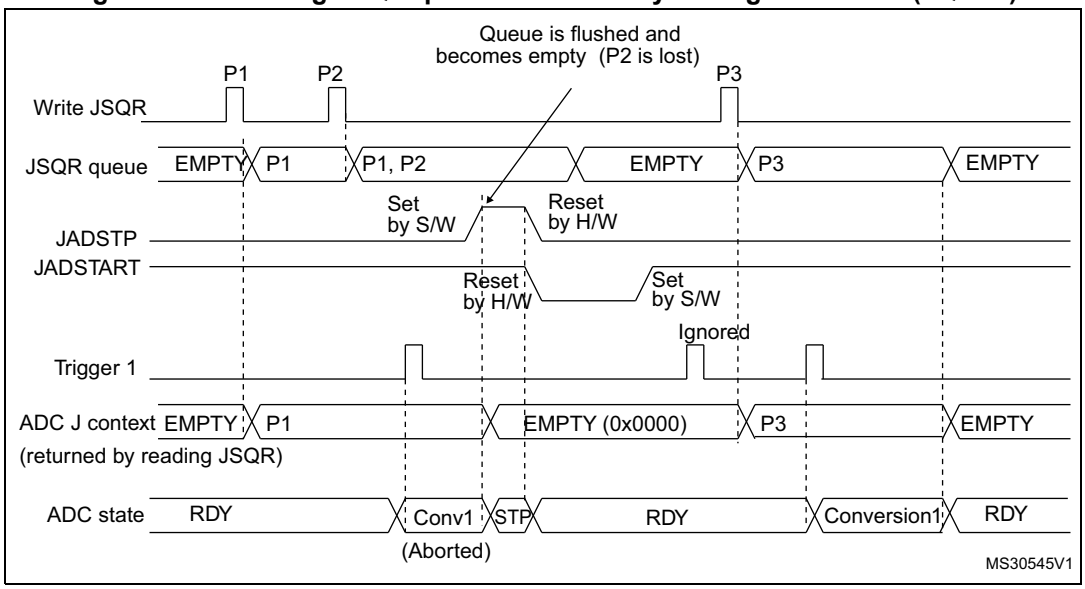
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 153. Flushing JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion



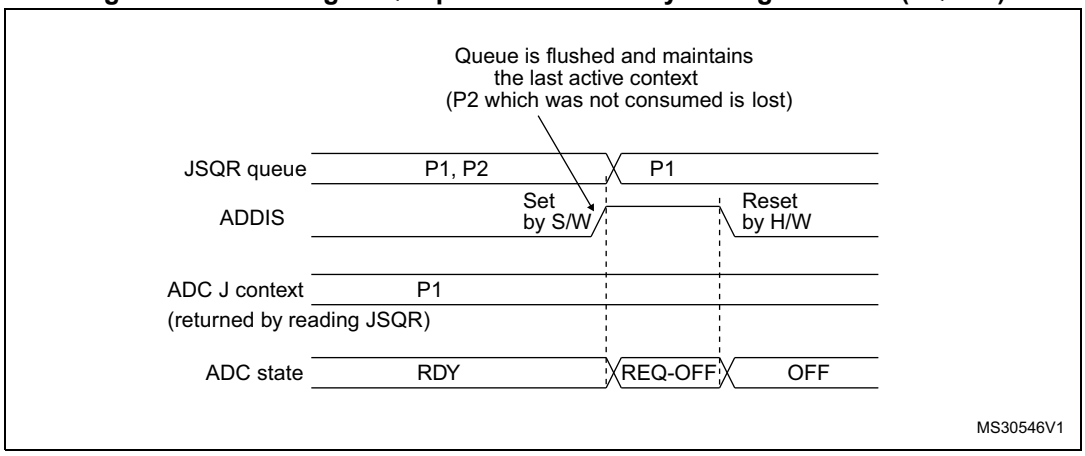
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 154. Flushing JSQR queue of context by setting JADSTP=1 (JQM=1)



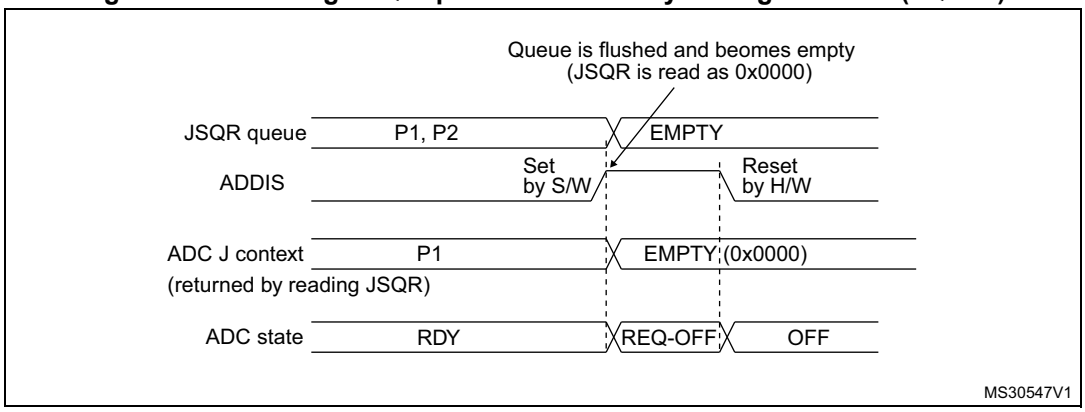
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 155. Flushing JSQR queue of context by setting ADDIS=1 (JQM=0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 156. Flushing JSQR queue of context by setting ADDIS=1 (JQM=1)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS=1 into the ADC_CFGR register.

Queue of context: Programming of the register ADC_JSQR

When the injected conversion queue of context is enabled (JQDIS=0), the ADC_JSQR must be programmed at one register write access. As JL[1:0] register define the number of the injected sequence, corresponding JSQ1 to JSQ4 must be written at same time. If ADC_JSQR is reprogrammed before the injected conversion start, reprogrammed data is put on the queue. When queue of context is empty, ADC_JSQR read back as 0x0000. Register access should not use the ‘read modify write’ sequence.

When ADC_JSQR is programmed when already 2 contexts are queued, it will raise JQOVF flag and generate the interrupt.

24.3.23 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 16, 14, 12, 10, 8 bits by programming the control bits RES[1:0]. *Figure 161*, *Figure 162*, *Figure 163* and *Figure 164* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 203*.

Table 203. T_{SAR} timings depending on resolution

RES	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$=24 MHz	$T_{adc_ker_ck}$ (ADC clock cycles) (with Sampling Time= 1.5 ADC clock cycles)	$T_{adc_ker_ck}$ (ns) at $F_{adc_ker_ck}$=24 MHz
16	8.5 ADC clock cycles	354.2	10 ADC clock cycles	416.7
14	7.5 ADC clock cycles	312.5	9 ADC clock cycles	375
12	6.5 ADC clock cycles	270.8	8 ADC clock cycles	333.3
10	5.5 ADC clock cycles	229.2	7 ADC clock cycles	291.7
8	4.5 ADC clock cycles	187.5	6 ADC clock cycles	250.0

24.3.24 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

24.3.25 End of conversion sequence (EOS, JEOS)

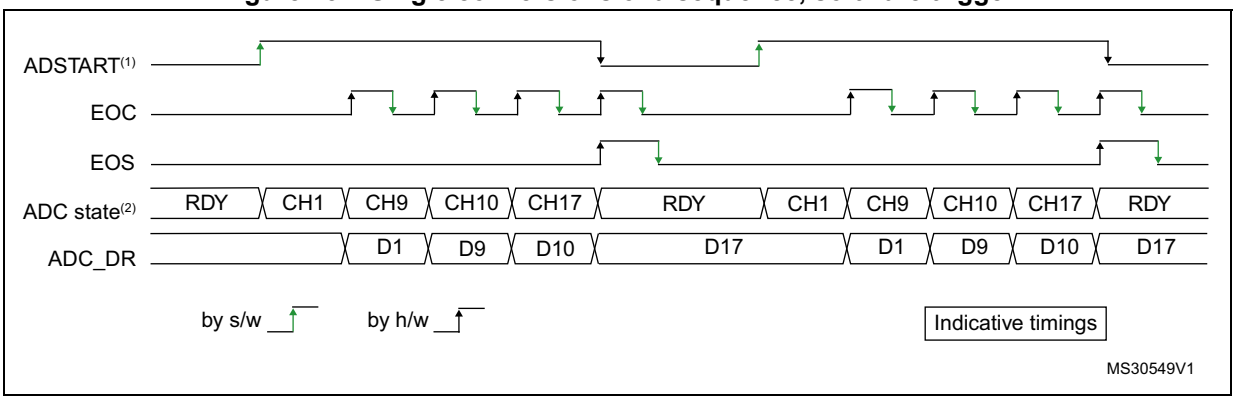
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

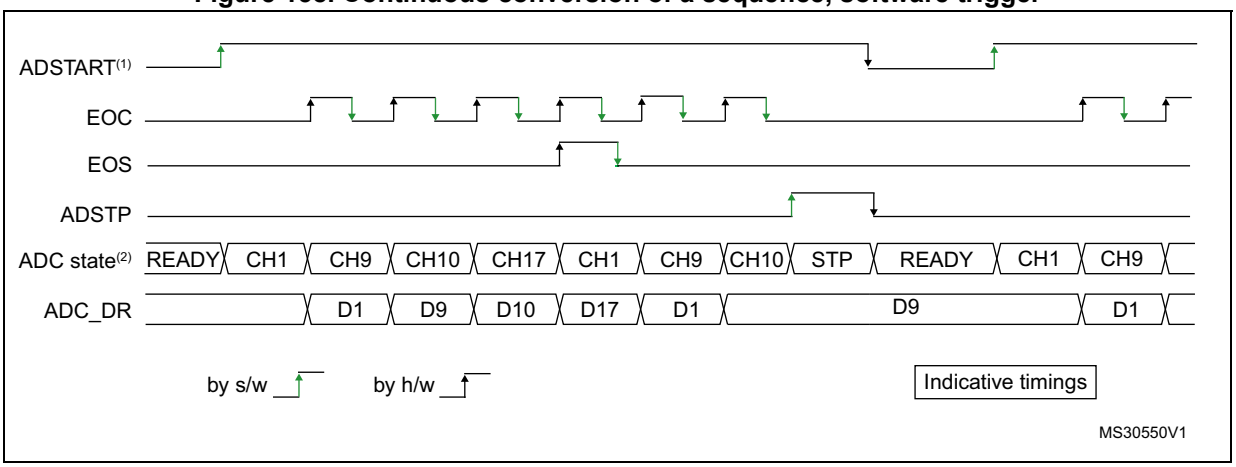
24.3.26 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 157. Single conversions of a sequence, software trigger



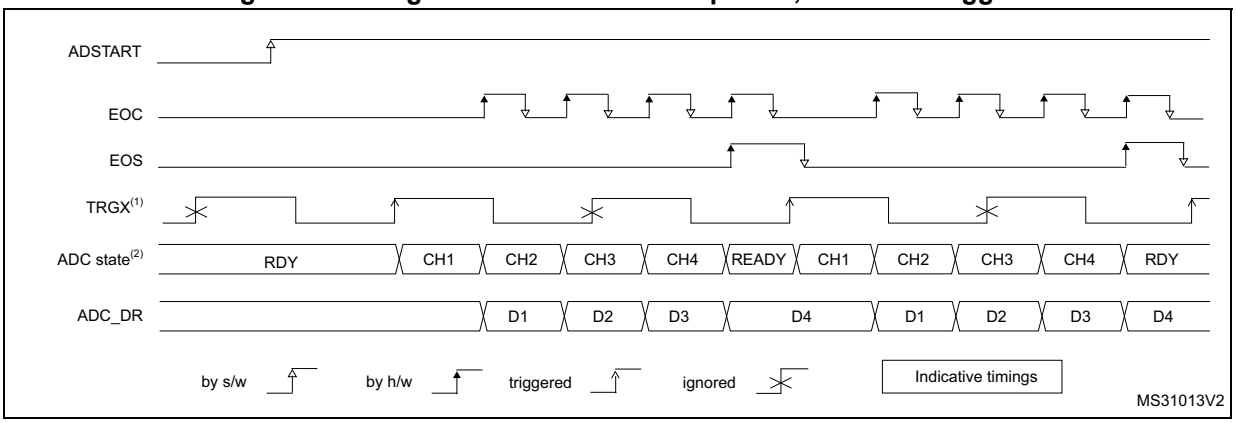
1. EXTEN=0x0, CONT=0
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 158. Continuous conversion of a sequence, software trigger



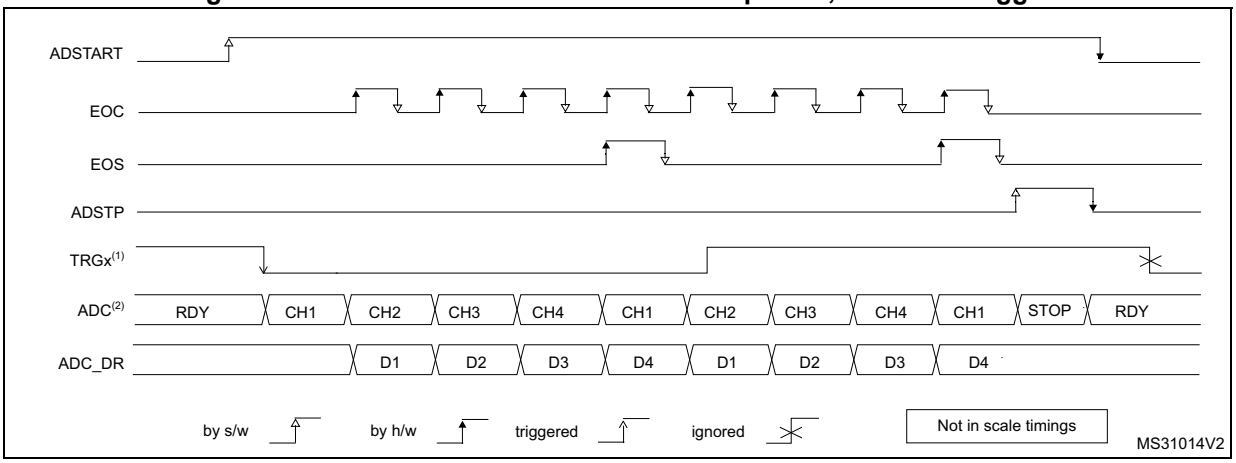
1. EXTEN=0x0, CONT=1
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 159. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

Figure 160. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

24.3.27 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 32 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 32 bits wide.

The OVSS[3:0] and LSHIFT[3:0] bitfields in the ADC_CFGR2 register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 161*, *Figure 162*, *Figure 163* and *Figure 164*.

Note: The data can be re-aligned in normal and in oversampling mode.

Offset

An offset y (y=1,2,3,4) can be applied to a channel by programming a value different from 0 in OFFSETy[25:0] bit field into ADC_OFRy register. The channel to which the offset will be applied is programmed into the bits OFFSETy_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[25:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

The offset value should be lower than the max conversion value (ex. 16bit mode, offset value max is 0xFFFF).

The offset correction is also supported in oversampling mode. For the oversampling mode, offset is subtracted before OVSS right shift applied.

Table 204 describes how the comparison is performed for all the possible resolutions for analog watchdog 1, 2, 3.

Table 204. Offset computation versus data resolution

Resolution (bits RES[2:0])	Subtraction between raw converted data and offset:		Result	Comments
	Raw converted Data, left aligned	Offset		
000: 16-bit	DATA[15:0]	OFFSET[25:0]	signed 27-bit data	-
001: 14-bit	DATA[15:2],00	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[1:0] to 00
010: 12-bit	DATA[15:4],00 00	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[3:0] to 0000
011: 10-bit	DATA[15:6],00 0000	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[5:0] to 000000
100: 8-bit	DATA[15:8],00 0000	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[7:0] to 00000000

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y=1,2,3,4), the offset compensation is disabled when ADC_OFRy [25:0] bitfield is reset. Otherwise, the offset for ADC_OFRy[30:26] channel is enabled.

Figure 161, *Figure 162*, *Figure 163* and *Figure 164* show alignments for signed and unsigned data together with corresponding OVSS and LSHIFT values.

Figure 161. Right alignment (offset disabled, unsigned value)

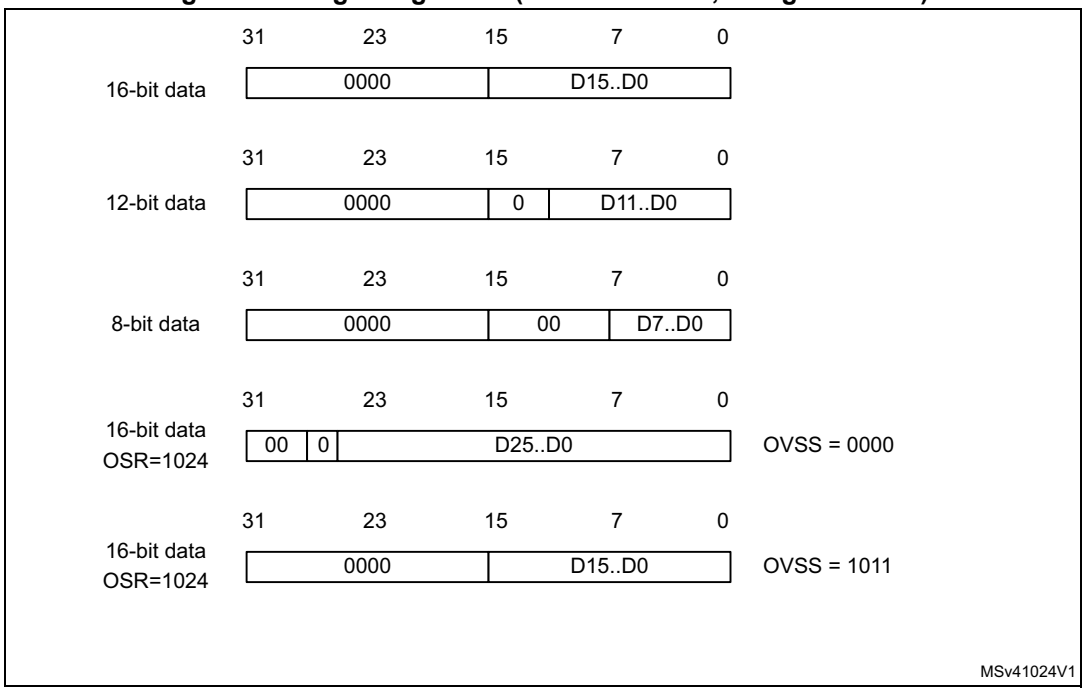


Figure 162. Right alignment (offset enabled, signed value)

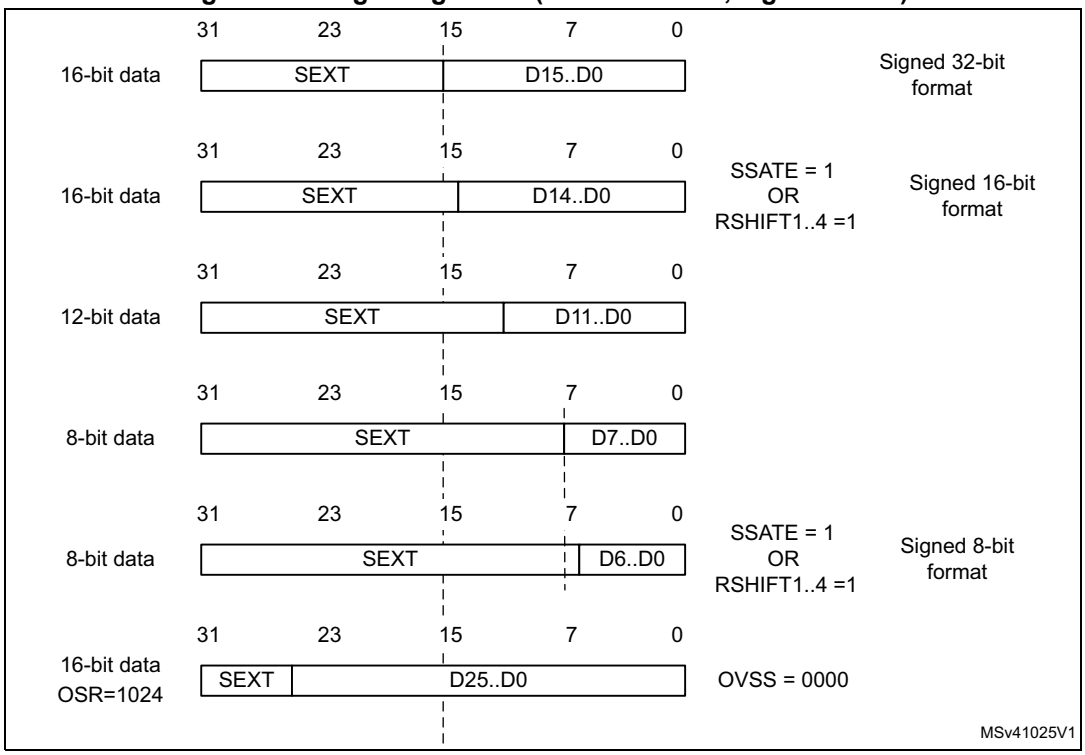


Figure 163. Left alignment (offset disabled, unsigned value)

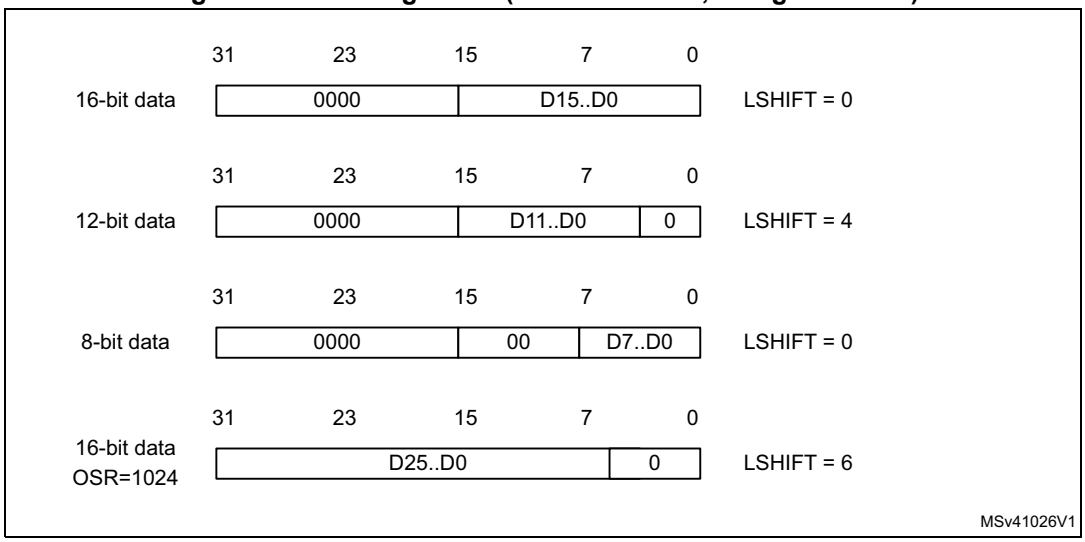
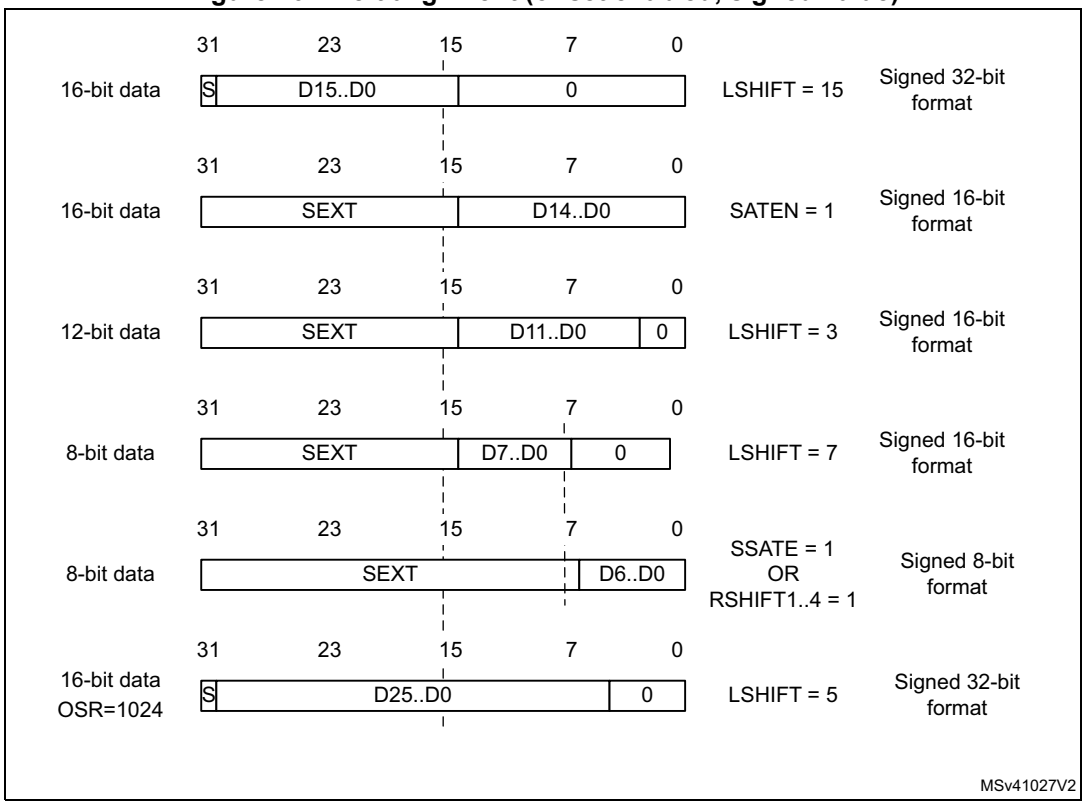


Figure 164. Left alignment (offset enabled, signed value)



16-bit and 8-bit signed format management: RSHIFTx,SSATE

The offset correction sign-extends the data format, resulting in an unsigned 16-bit conversion being extended to 17-bit signed format, for instance.

Three options are offered for formatting 8-bit and 16-bit conversion results.

For each offset correction channel 1 to 4, a RSHIFT1..4 bit in the ADC_CFGR2 register allows to have the result right-shifted 1-bit and have it fitting a standard 8 or 16-bit format.

Another option is to have the result saturated to the 16-bit and 8-bit signed formats, for the following cases only: RES[2:0] = 000 (16-bit format) and RES[2:0] = 100 (8-bit format).

This mode is enabled with the SSATE bit in the ADC_OFRy register.

The table below summarizes the 3 available use case for 16-bit format.

Table 205. 16-bit data formats

SSATE	RSHIFTx	Format	Data range (offset = 0x8000)
0	0	Sign-extended 17-bit significant data SEXT[31:16] DATA[15:0]	0x00007FFF - 0x FFFF8000
0	1	Sign-extended right-shifted 16-bit significant data SEXT[31:15] DATA[14:0]	0x3FFF - 0xC000
1	0	Sign-extended saturated 16-bit significant data SEXT[31:15] DATA[14:0]	7FFF - 0x8000
1	1	Reserved	-

Numerical examples are given in *Table 206* with 3 different offset values.

Table 206. Numerical examples for 16-bit format (bold indicates saturation)

Raw conversion result	Offset value	Result SSATE = 0 RSHIFT = 0	Result SSATE = 0 RSHIFT = 1	Result SSATE = 1 RSHIFT = 0
0xFFFF	0x8000	0x0000 7FFF	3FFF	7FFF
0x8000		0x0000 0000	0	0
0x0000		0xFFFF 8000	C000	8000
0xFFFF	0x8020	0x0000 7FDF	3FEF	7FDF
0x8000		0xFFFF FFE0	FFF0	FFE0
0x0000		0xFFFF 7FE0	BFF0	8000
0xFFFF	0x7FF0	0x0000 800F	4007	7FFF
0x8000		0x0000 0010	8	0010
0x0000		0xFFFF 8010	C008	8010

When oversampling mode is active, the SSATE and RSHIFT1..4 bits are not supported.

ADC overrun (OVSR, OVRMOD)

The overrun flag (OVSR) notifies of a buffer overrun event, when the regular converted data was not read (by the CPU or the DMA) before new converted data became available.

The OVSR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE=1.

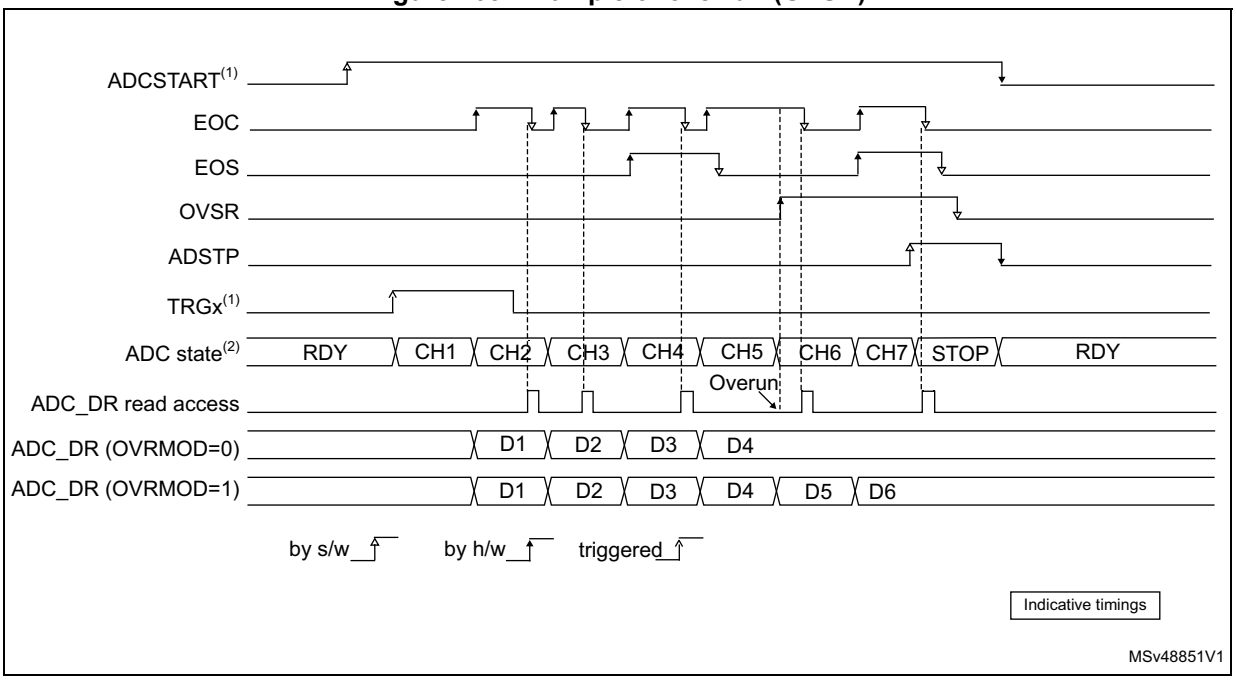
When an overrun condition occurs, the ADC is still operating and can continue to convert unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

OVSR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD=0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVSR remains at 1, any further conversions will occur but the result data will be also discarded.
- OVRMOD=1: The data register is overwritten with the last conversion result and the previous unread data is lost. If OVSR remains at 1, any further conversions will operate normally and the ADC_DR register will always contain the latest converted data.

Figure 165. Example of overrun (OVSR)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversion without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVSR flag should be ignored by the software. An overrun event will not prevent the ADC from continuing to convert and the ADC_DR register will always contain the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMNGT bit = 01 or 11 in the ADC_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVSR=1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *Section : ADC overrun (OVSR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVSR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMNGT of the ADC_CFGR register in single ADC mode, or with bit DAMDF of the ADCx_CCR register in dual ADC mode:

- DMA one shot mode (DMNGT bit = 01).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT bit = 11)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMNGT=01)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when DMA_EOT interrupt occurs - refer to DMA paragraph) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

DMA circular mode (DMNGT=11)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

DMA with FIFO

The output data register has 8 stage FIFO. Two different DMA requests are generated parallel. When a data is available, “SREQ single request” generated, when 4 data are available, “BREQ burst request” generated. DMA2 can be programmed either single transfer mode or incremental burst mode(4 beats), according to this mode, correct request line is selected by the DMA2. Please refer to the DMA2 chapter for further information.

24.3.28 Managing conversions using the DFSDM

The ADC conversion results can be transferred directly to the Digital Filter for Sigma Delta Modulators (DFSDM).

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers 16 least significant bits of the regular data register data to the DFSDM, which in turns will reset the EOC flag once the transfer is effective.

The data format must be 16-bit signed:

ADC_DR[31:16] = don’t care

ADC_DR[15] = sign

ADC_DR[14:0] = data

Any value above 16-bit signed format will be truncated.

24.3.29 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY=1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see *Figure 166*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 167*).

This is a way to automatically adapt the speed of the ADC to the speed of the system which will read the data.

The delay is inserted after each regular conversion (whatever DISCEN=0 or 1) and after each sequence of injected conversions (whatever JDISCEN=0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to re-start a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 167*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 169*).

The behavior is slightly different in auto-injected mode (JAUTO=1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 170*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

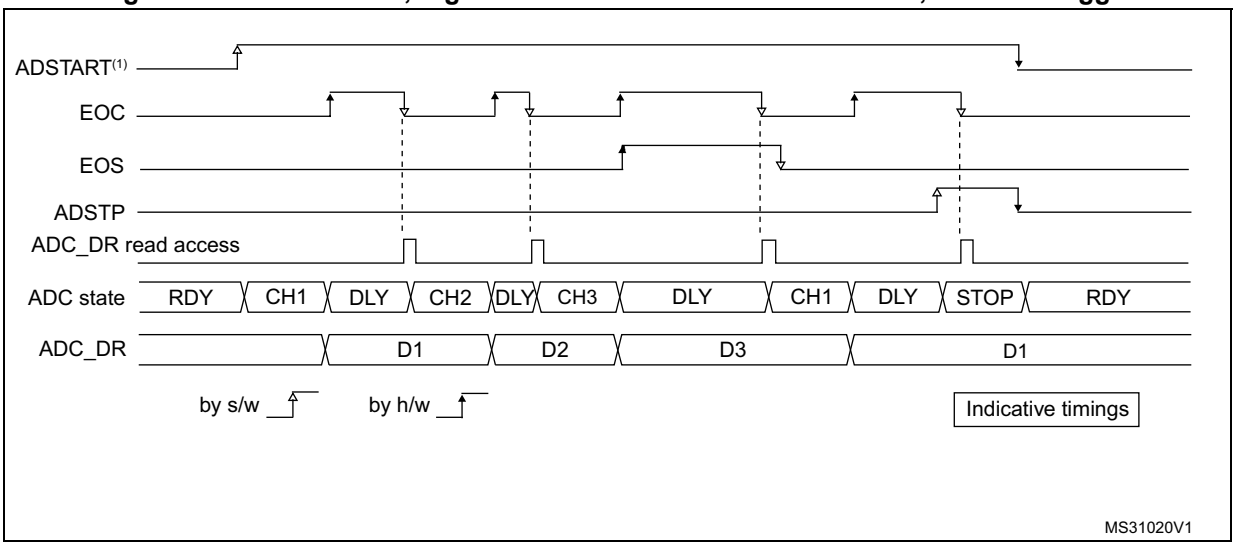
1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can re-start if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

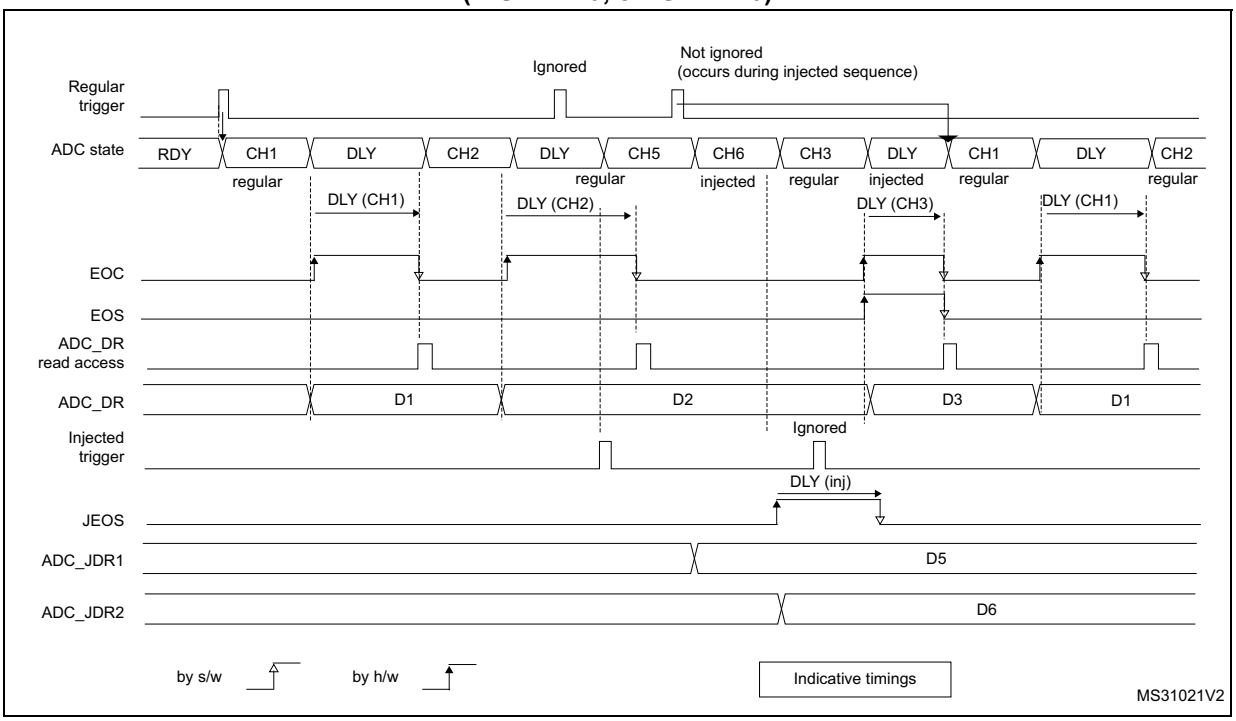
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 166. AUTODLY=1, regular conversion in continuous mode, software trigger



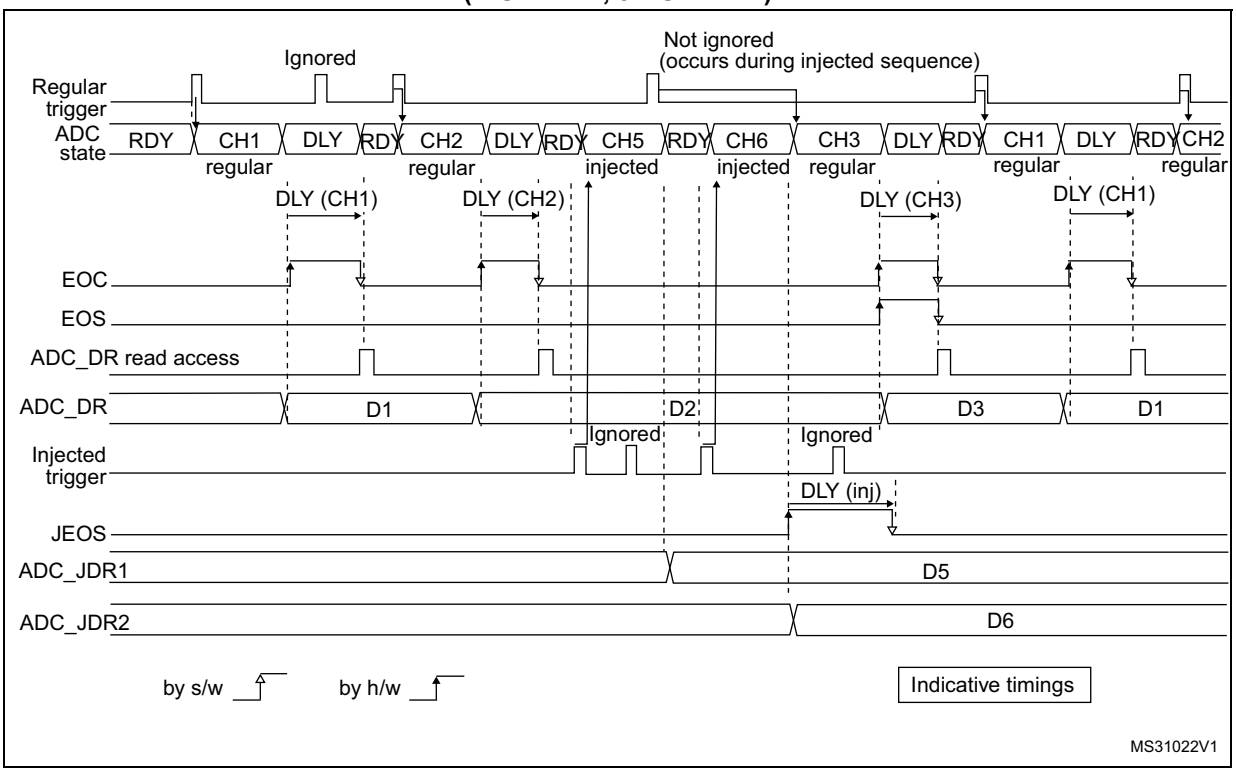
1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 167. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)



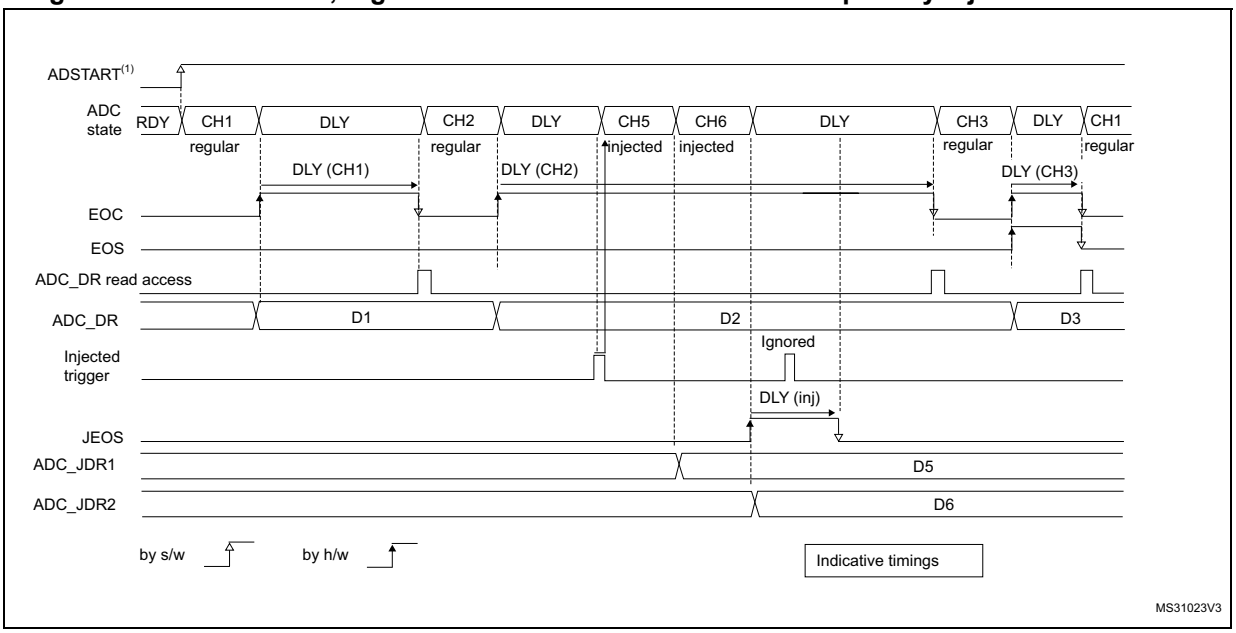
1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 168. AUTODLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)



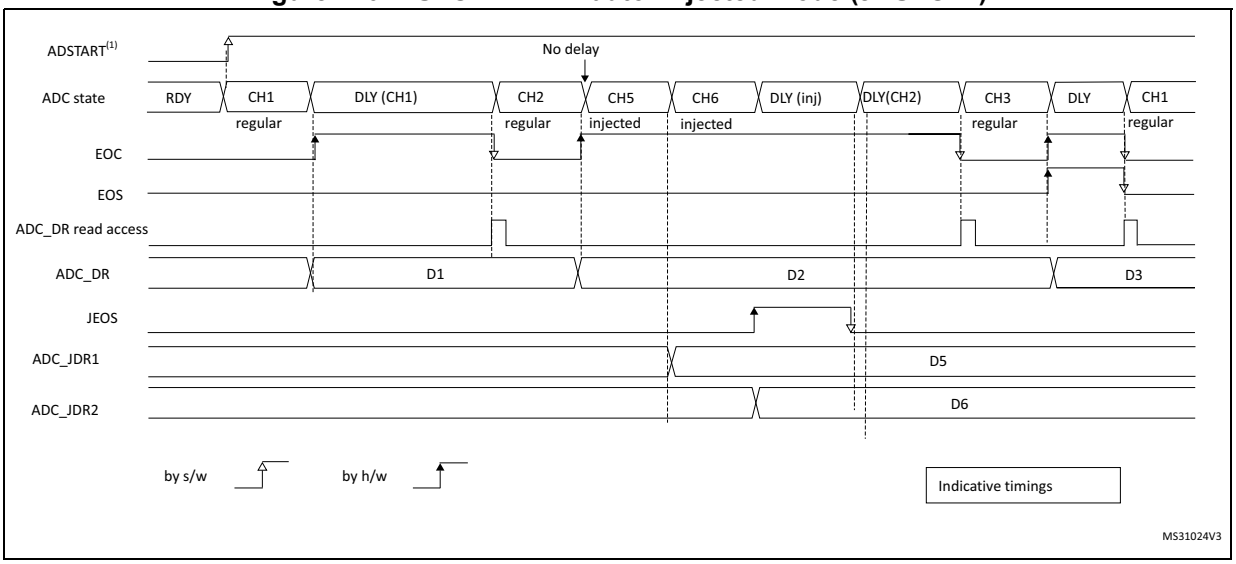
1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=1, CHANNELS = 5,6

Figure 169. AUTODLY=1, regular continuous conversions interrupted by injected conversions



1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 170. AUTODLY=1 in auto- injected mode (JAUTO=1)

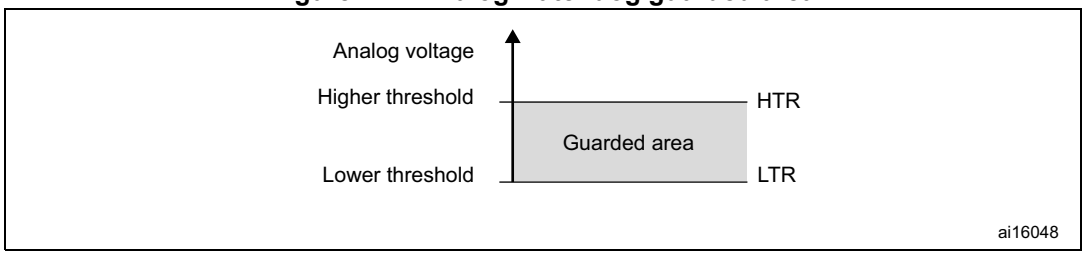


1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5,6

24.3.30 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 171. Analog watchdog guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDyIE in the ADC_IER register (x=1,2,3).

AWDy (y=1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels(1) remain within a configured voltage range (window).

Table 207 shows how the ADC_CFGRy registers should be configured to enable the analog watchdog on one or more channels.

Table 207. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWDyCH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HTR1[25:0] of the ADC_HTR1 register and LTR1[25:0] of the ADC_LTR1 register for the analog watchdog 1.

The threshold can be up to 26-bits (16-bit resolution with oversampling, OSR=1024).

When converting data with a resolution of less than 16 bits (according to bits RES[2:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 16-bit converted data (left aligned to the half-word boundary).

Table 208 describes how the comparison is performed for all the possible resolutions for analog watchdog 1,2,3.

Table 208. Analog watchdog 1,2,3 comparison

Resolution (bit RES[2:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
000: 16-bit	DATA[15:0]	LTR1[25:0] and HTR1[25:0]	-
001: 14-bit	DATA[15:2],00	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[1:0] and HTR1[1:0] to 00
010: 12-bit	DATA[15:4],0000	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[3:0] and HTR1[3:0] to 0000
011: 10-bit	DATA[15:6],000000	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[5:0] and HTR1[5:0] to 000000
100: 8-bit	DATA[15:8],00000000	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[7:0] and HTR1[7:0] to 00000000

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCHy[19:0] (y=2,3).

The corresponding watchdog is enabled when any bit of AWDCHy[19:0] (y=2,3) is set.

The threshold can be up to 26-bits (16-bit resolution with oversampling, OSR=1024) and are programmed with the ADC_HTR2, ADC_LTR2, ADC_LTR3, and ADC_HTR3 registers.

When converting data with a resolution of less than 16 bits (according to bits RES[2:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 16-bit converted data (left aligned to the half-word boundary).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCx_AWDy_OUT (x=ADC number, y=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCx_AWDy_OUT signal as ETR.

ADCx_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADCx_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCx_AWDy_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCx_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS=1). Note that stopping regular or injected conversions (setting ADSTP=1 or JADSTP=1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDy flag has no influence on the generation of ADCx_AWDy_OUT (ex: ADCy_AWDy_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 172. ADCy_AWDx_OUT signal generation (on all regular channels)

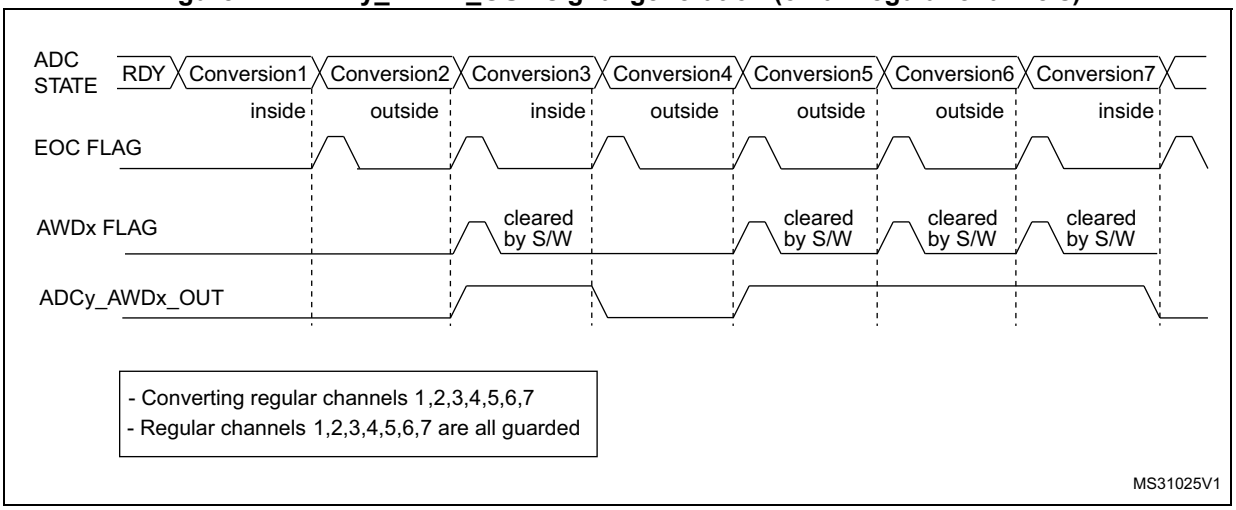


Figure 173. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW)

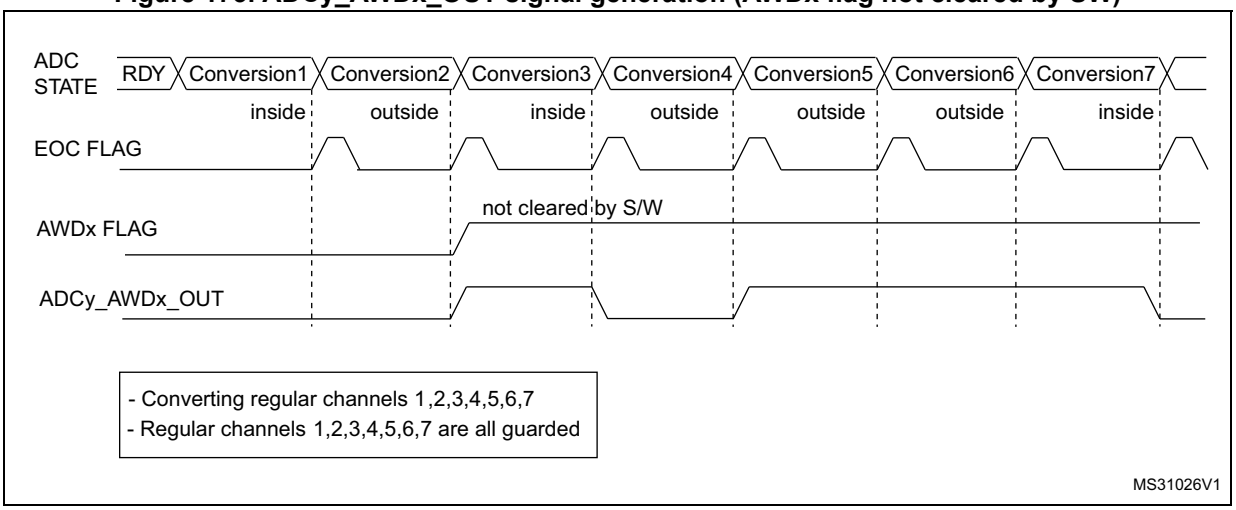


Figure 174. ADCy_AWDx_OUT signal generation (on a single regular channel)

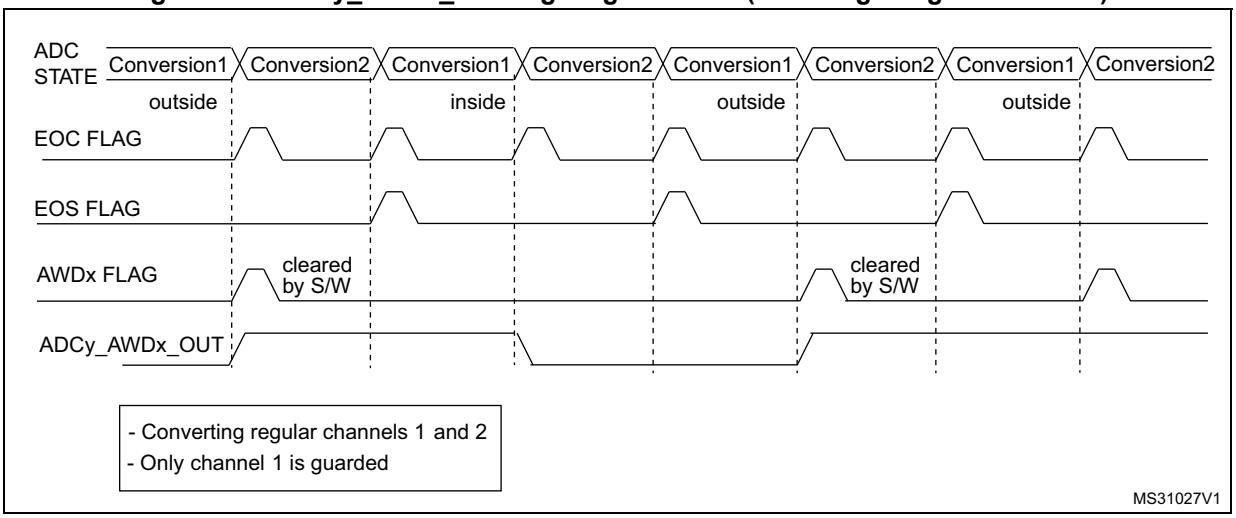
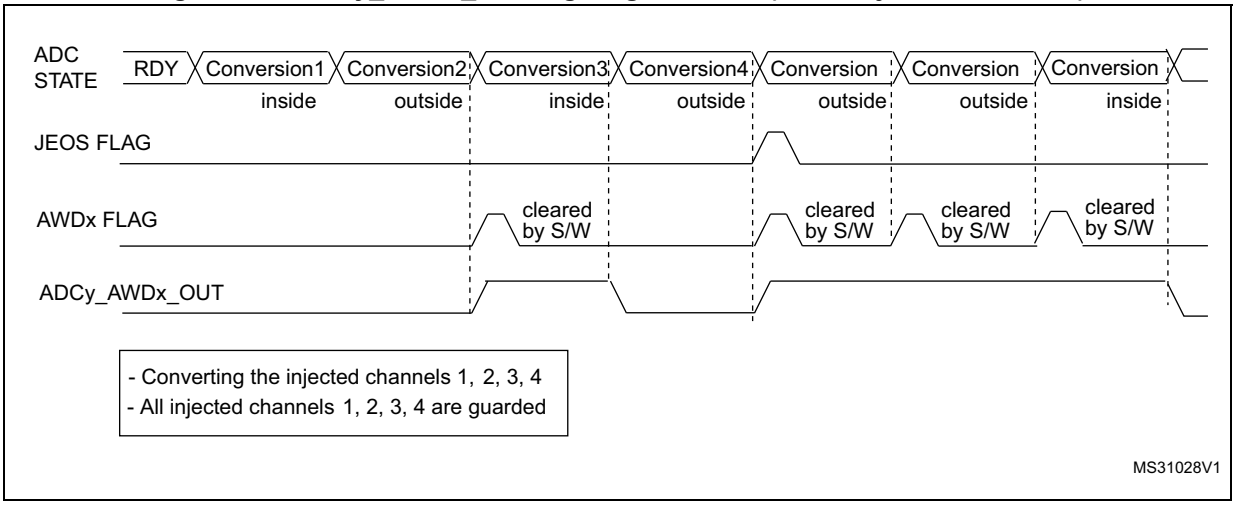


Figure 175. ADCy_AWDx_OUT signal generation (on all injected channels)



24.3.31 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 26-bit (16-bit values and OSR = 1024).

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

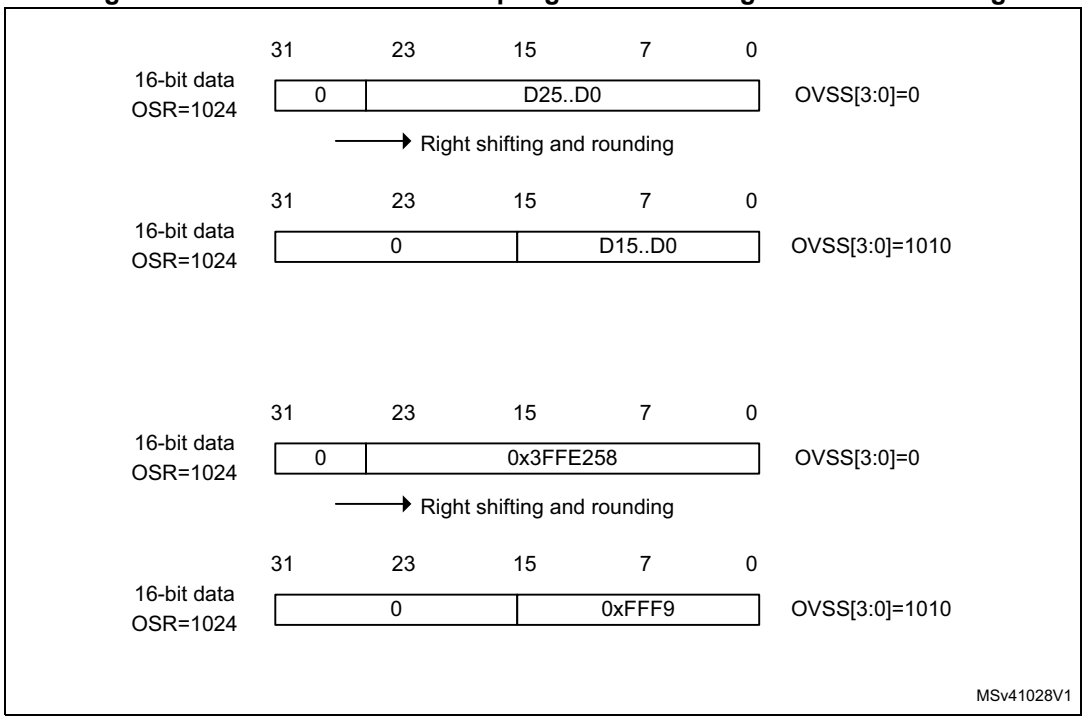
It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OSR[9:0] bits in the ADC_CFGR2 register, and can range from 2x to 1024x. The division coefficient M consists of a right bit shift up to 10 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 26 bits (1024 x 16-bit results), which can be left or right shifted. When right shifting is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being transferred into the ADC_DR data register.

The *Table 176* gives a numerical example of the processing, from a raw 26-bit accumulated data to the final 16-bit result.

Figure 176. 16-bit result oversampling with 10-bits right shift and rouding



There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversions, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follow:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- the end of sequence (EOS) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

Single ADC operating modes support when oversampling

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[2:0] bits in ADC_CFGR register) are accumulated, truncated, rounded and shifted in the same way as 16-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR is ignored and the data are always provided right-aligned.

Offset correction is not supported in oversampling mode.

Analog watchdog

The analog watchdog functionality is maintained (AWDSGL and AWDEN bits), with the following difference:

- the RES[2:0] bits are ignored, comparison is always done on using the full 26-bit values HTRx[25:0] and LTRx[25:0]
- the comparison is performed on the oversampled accumulated value before shifting

Note: Care must be taken when using high shifting values, this will reduce the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HT[0:7] / LT[[0:7], and HT[11:8] / LT[11:8] must be kept reset.

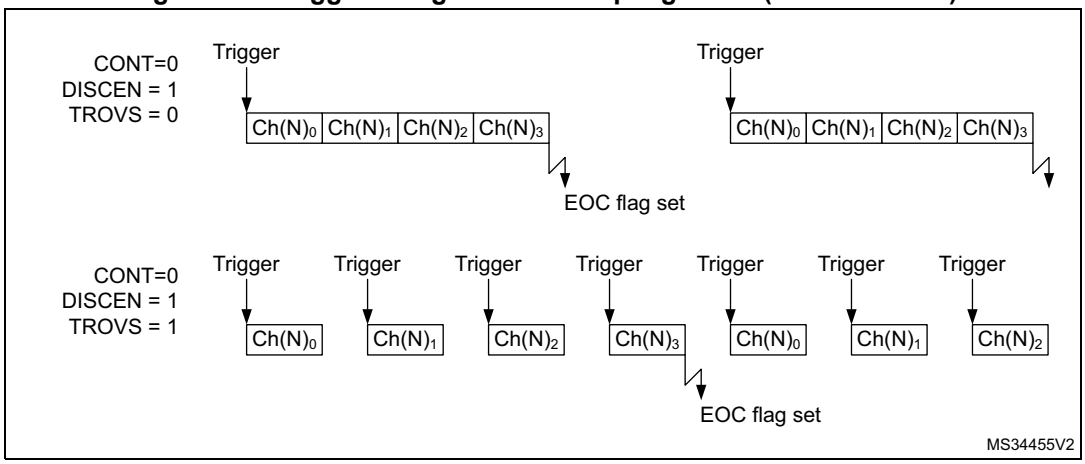
Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The *Figure 177* below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 177. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

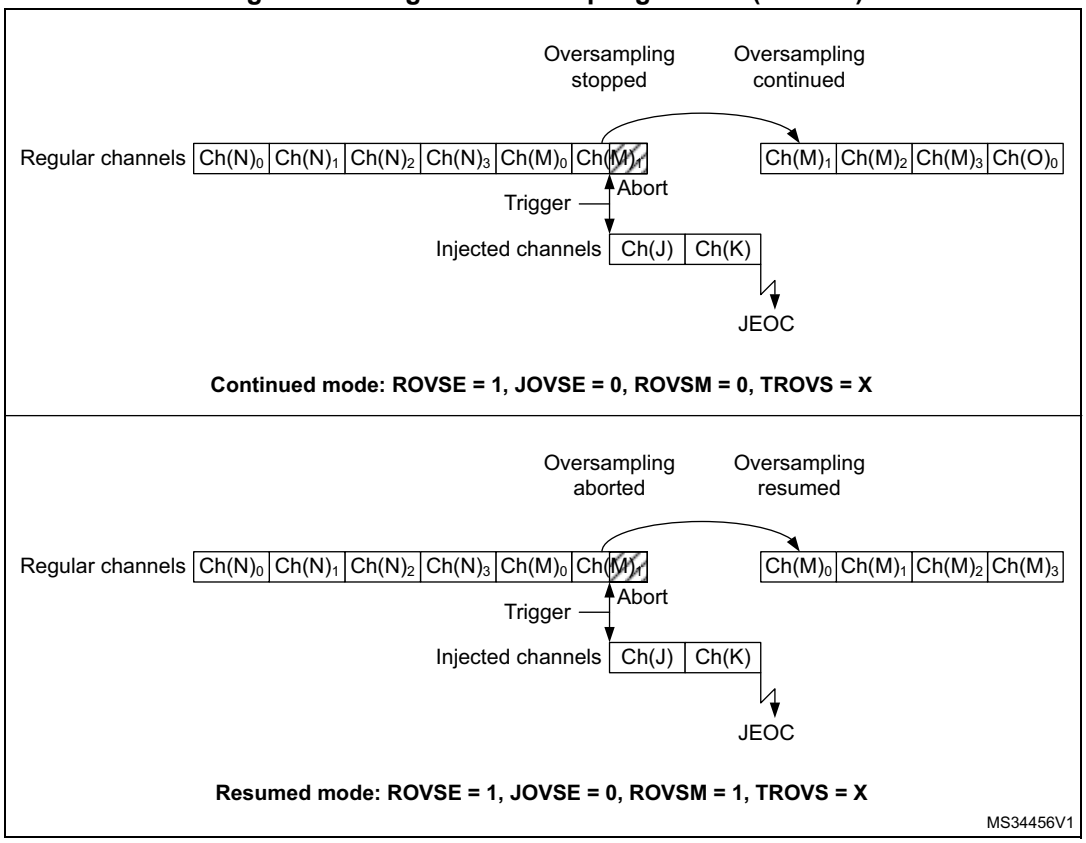
Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- in continued mode, the accumulation re-starts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling will be completed whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- in resumed mode, the accumulation re-starts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be completed and the regular sequencer will be blocked.

The *Figure 178* gives examples for a 4x oversampling ratio.

Figure 178. Regular oversampling modes (4x ratio)



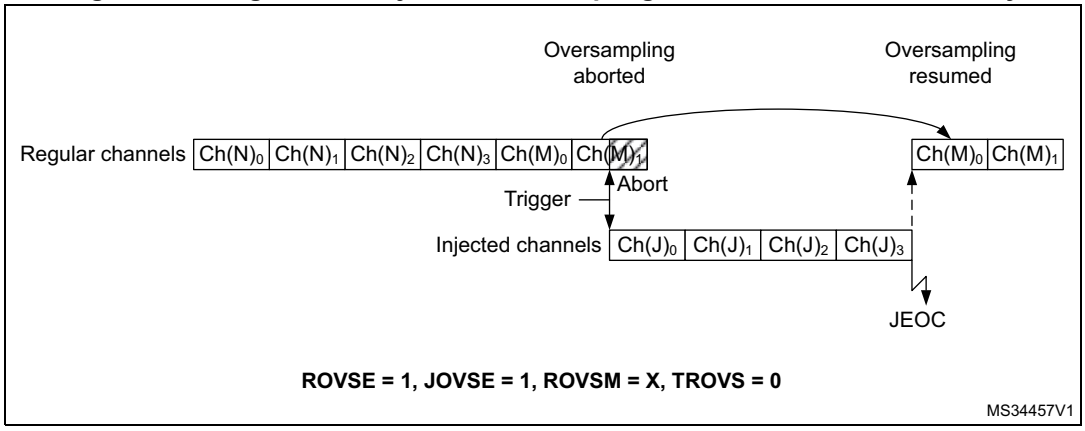
Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on *Figure 179* below.

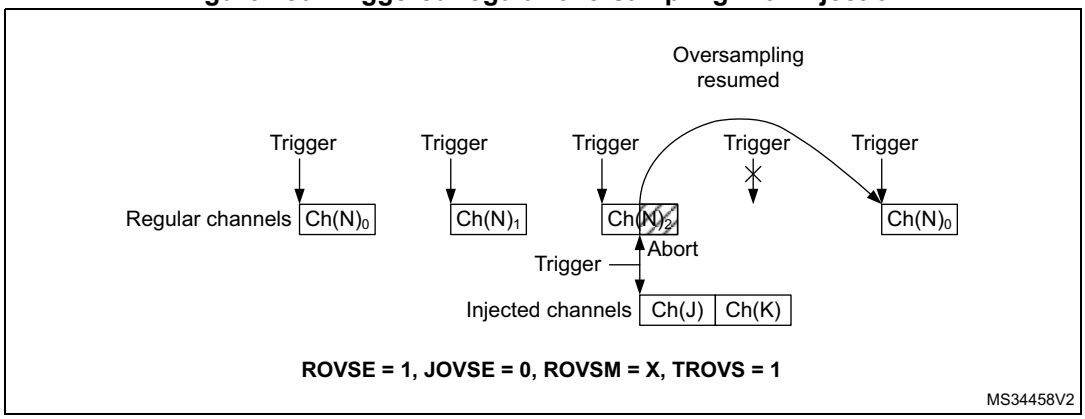
Figure 179. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on *Figure 180* below.

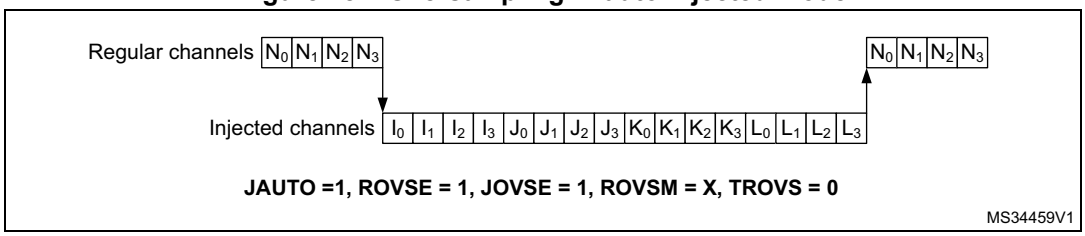
Figure 180. Triggered regular oversampling with injection



Autoinjected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The *Figure 181* below shows how the conversions are sequenced.

Figure 181. Oversampling in auto-injected mode



It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO=1, DISCEN=0, JDISCEN=0, ROVSE=1, JOVSE=1 and TROVSE=1.

Dual ADC modes support when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

Combined modes summary

The *Table 209* below summarizes all combinations, including modes not supported.

Table 209. Oversampler operating modes summary

Regular Over-sampling ROVSE	Injected Over-sampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered Regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

24.3.32 Dual ADC modes

In devices with two ADCs or more, dual ADC modes can be used (see *Figure 182*):

- ADC1 and ADC2 can be used together in dual mode (ADC1 is master)

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCx_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode
- Injected simultaneous mode + Interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADCx_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADC_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger,

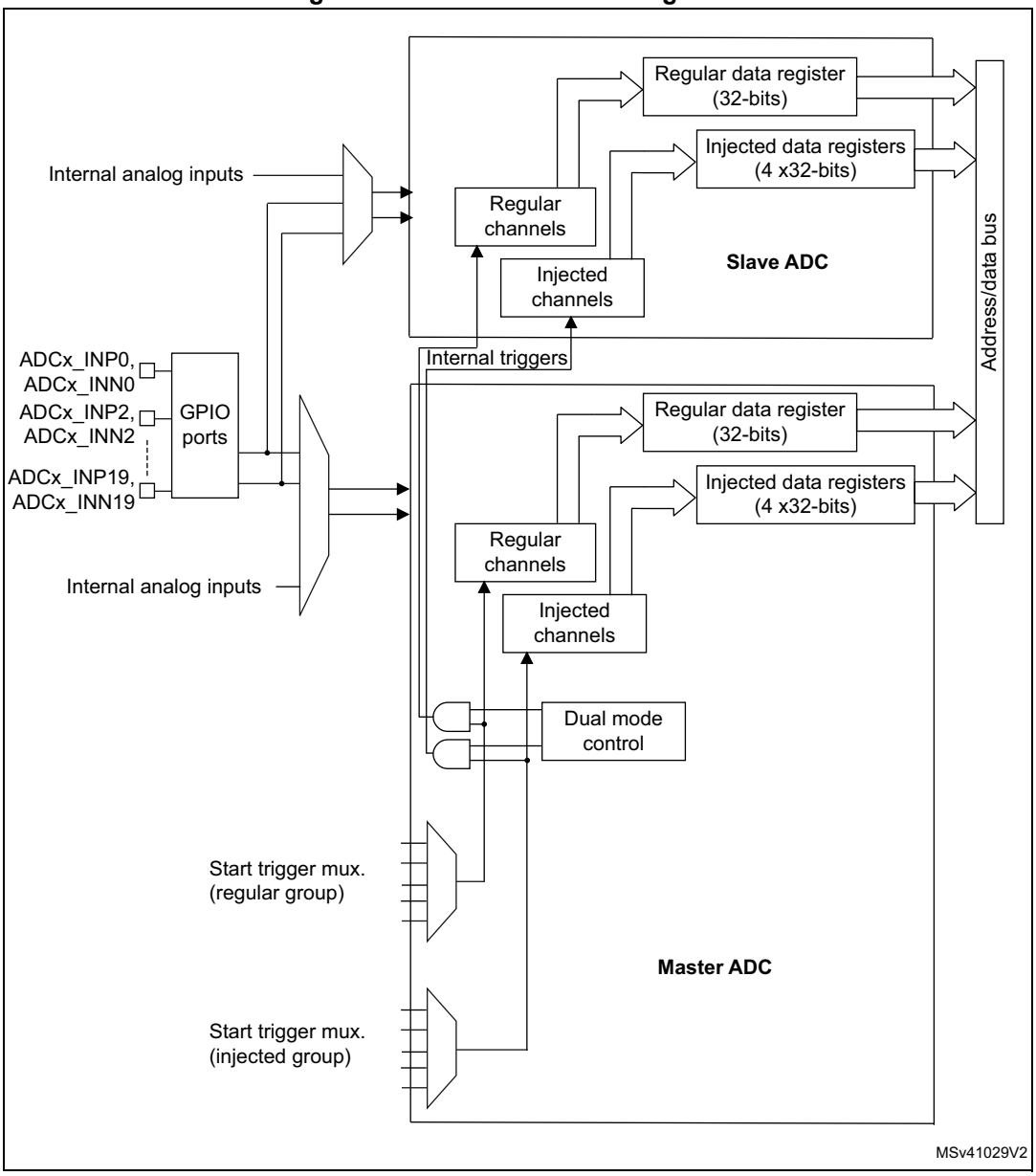
and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCx_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADCx_CSR).

Figure 182. Dual ADC block diagram(1)



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCx_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0]=00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL[4:0] bits in the ADC_JSQR register).

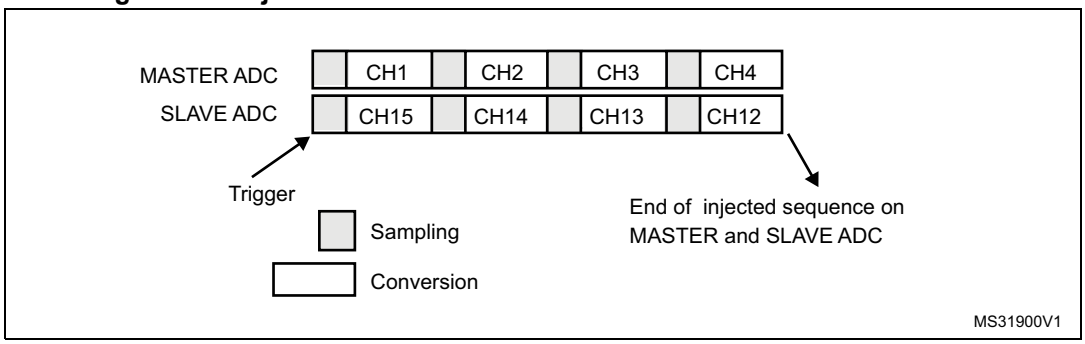
Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length and inside a sequence, the N-th conversion in master ans slave must be configured with the same sampling time.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in *Figure 183*), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 183. Injected simultaneous mode on 4 channels: dual ADC mode



If JDISCEN=1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored.
  There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL[4:0] bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) will abort the current simultaneous conversions, which are re-started once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length and inside a sequence, the N-th conversion in master ans slave must be configured with the same sampling time.

Software is notified by interrupts when it can read the data:

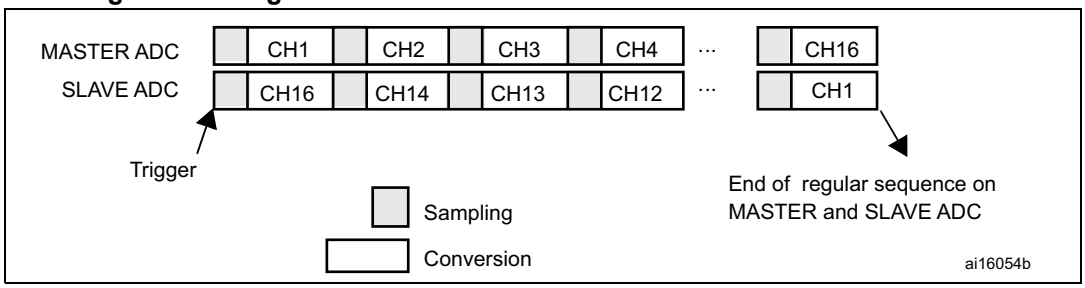
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in *Figure 184*), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADCx_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits DAMDF[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configuring Dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  - Configure DAMDF[1:0]=0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
  - both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Note: When DAMDF[1:0]=0b10 or 0b11, the user must program the same number of conversions in the master's sequence as in the slave's sequence. Otherwise, the remaining conversions will not generate a DMA request.

Figure 184. Regular simultaneous mode on 16 channels: dual ADC mode



If DISCEN=1 then each “n” simultaneous conversions of the regular sequence require a regular trigger event to occur (“n” is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADCx_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADCx_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADCx_CCR register. This delay starts to count after the end of the sampling phase of the master conversion. This way, an ADC cannot start a conversion if the

complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADCx_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as following:

- Configure DAMDF[1:0]=0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADCx_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCx_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADCx_CCR register.
- Both EOC flags are cleared when the DMA reads the ADCx_CCR register.

Figure 185. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode

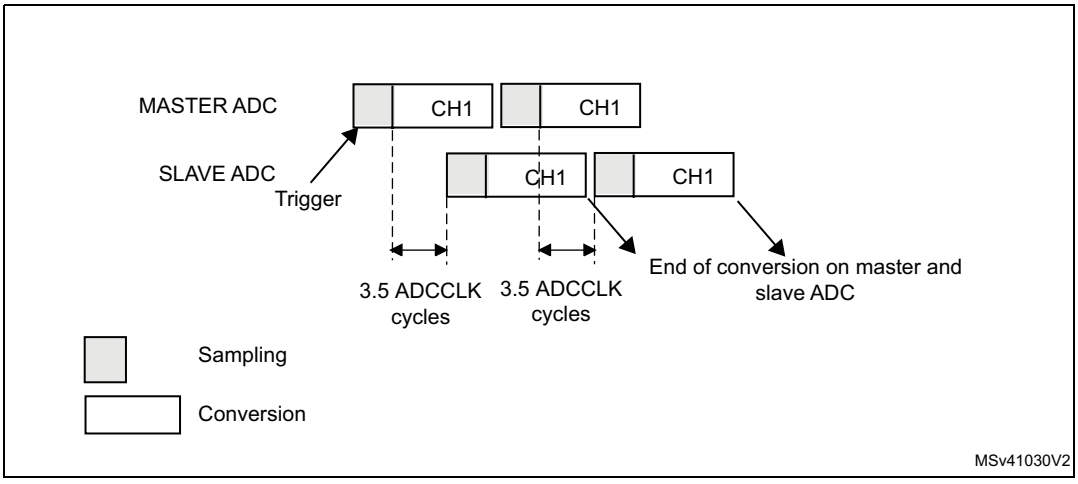
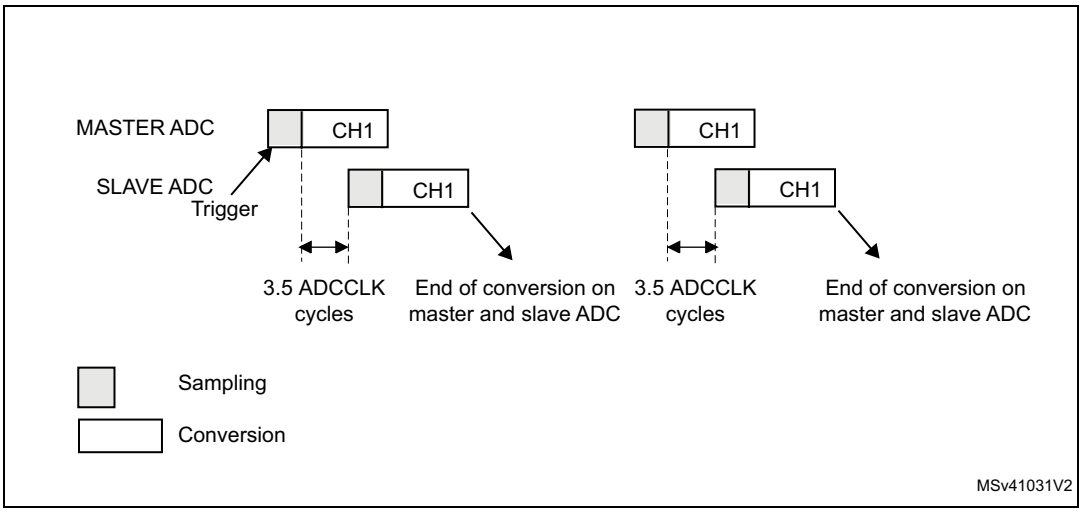


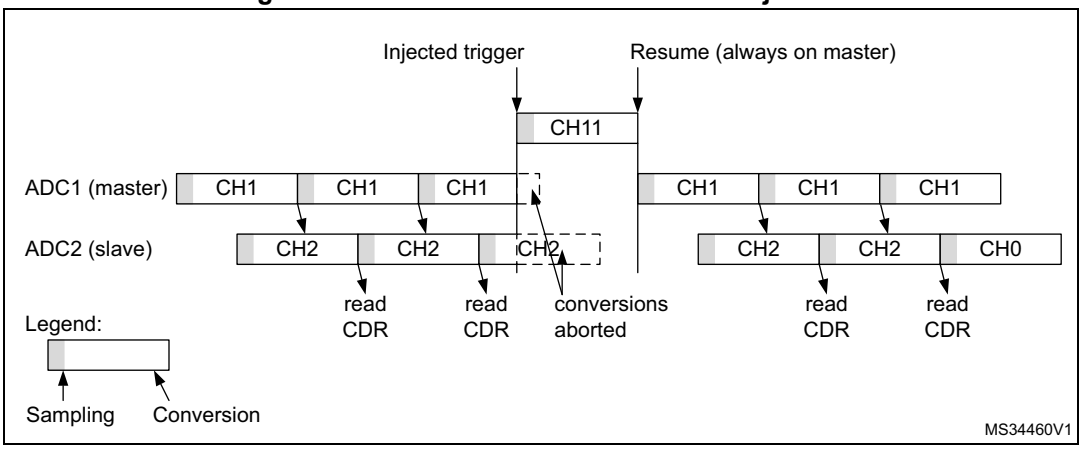
Figure 186. Interleaved mode on 1 channel in single conversion mode: dual ADC mode



If DISCEN=1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is re-started from the master (see *Figure 187* below).

Figure 187. Interleaved conversion with injection



Alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.

Injected discontinuous mode disabled (JDISCEN=0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

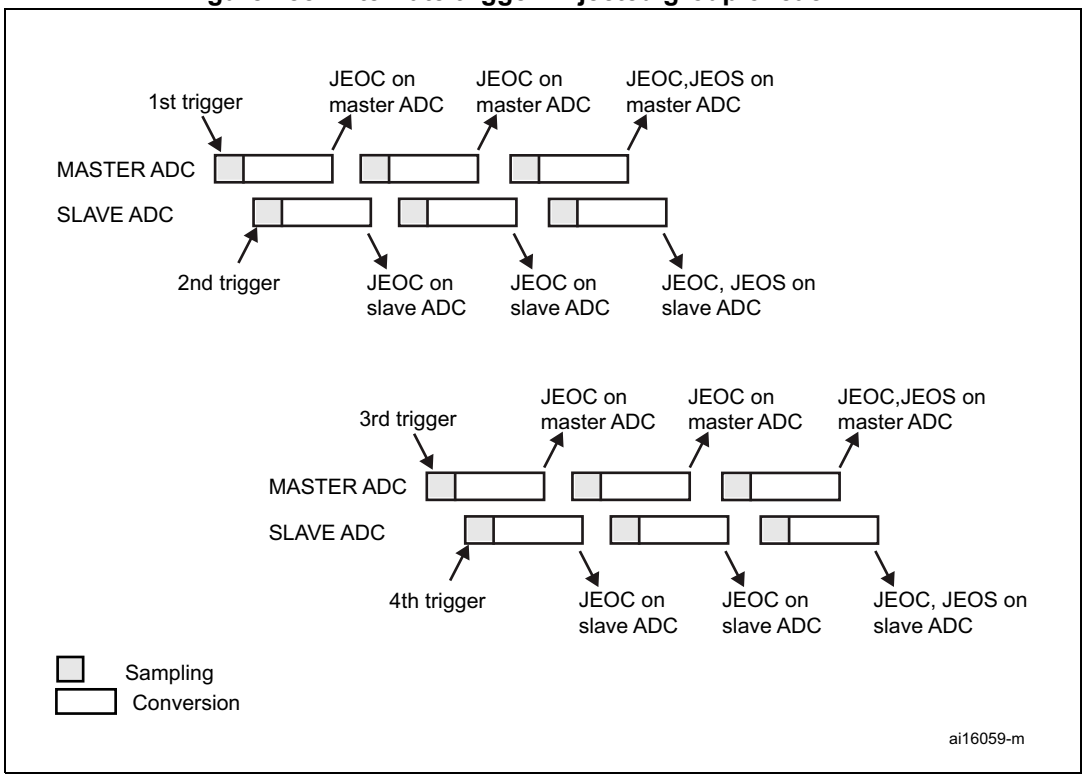
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 188. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN=1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

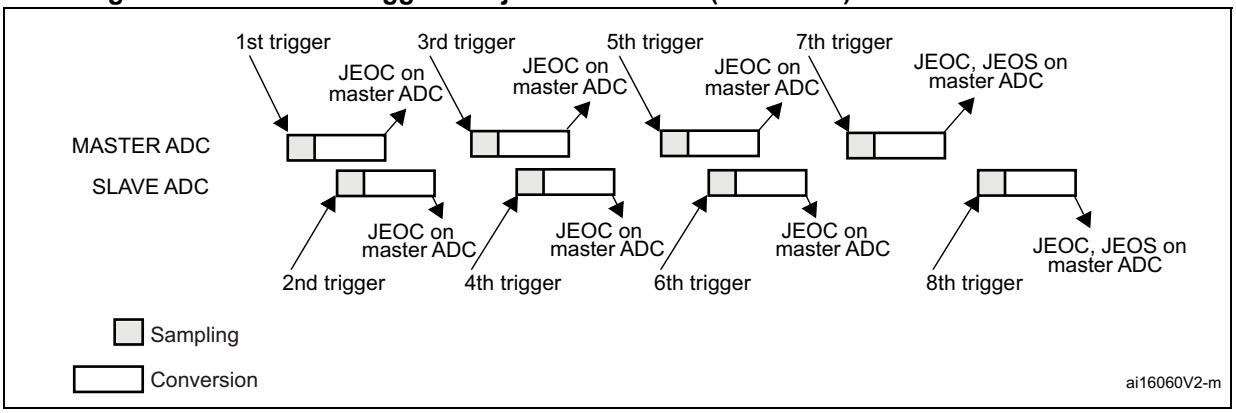
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 189. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: The sequences must be converted with the same length, the N-th conversion in master and slave mode must be configured with the same sampling time inside a given sequence, or the interval between triggers has to be longer than the long conversion time of the 2 sequences. If the above conditions are not respected, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

This mode is selected by programming bits DUAL[4:0]=00010.

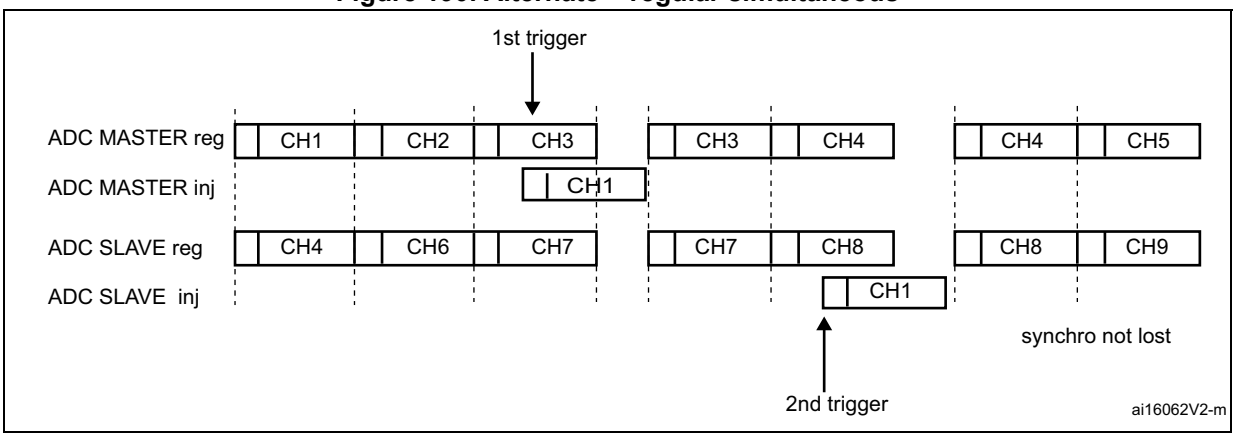
It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 190* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected

conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

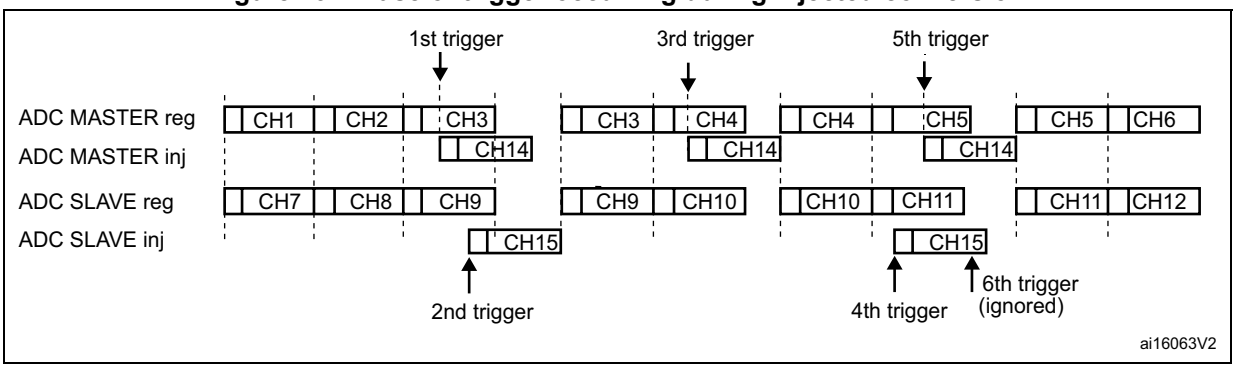
Note: The sequences must be converted with the same length, the N-th conversion in master and slave mode must be configured with the same sampling time inside a given sequence, or the interval between triggers has to be longer than the long conversion time of the 2 sequences. If the above conditions are not respected, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 190. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 191* shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 191. Case of trigger occurring during injected conversion



Combined injected simultaneous plus interleaved

This mode is selected by programming bits DUAL[4:0]=00011

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is alway the master's one. *Figure 192*, *Figure 193* and *Figure 194* show the behavior using an example.

Caution: In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.

Figure 192. Interleaved single channel CH0 with injected sequence CH11, CH12

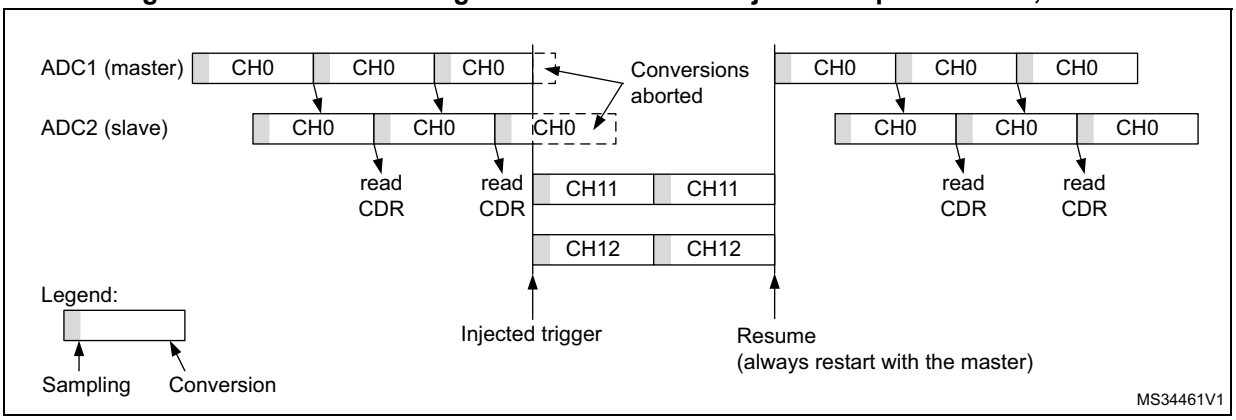


Figure 193. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

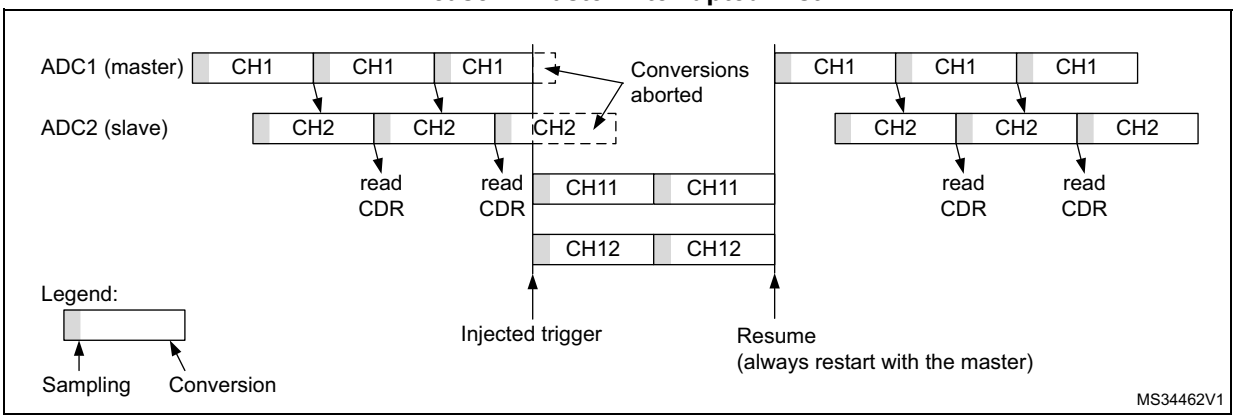
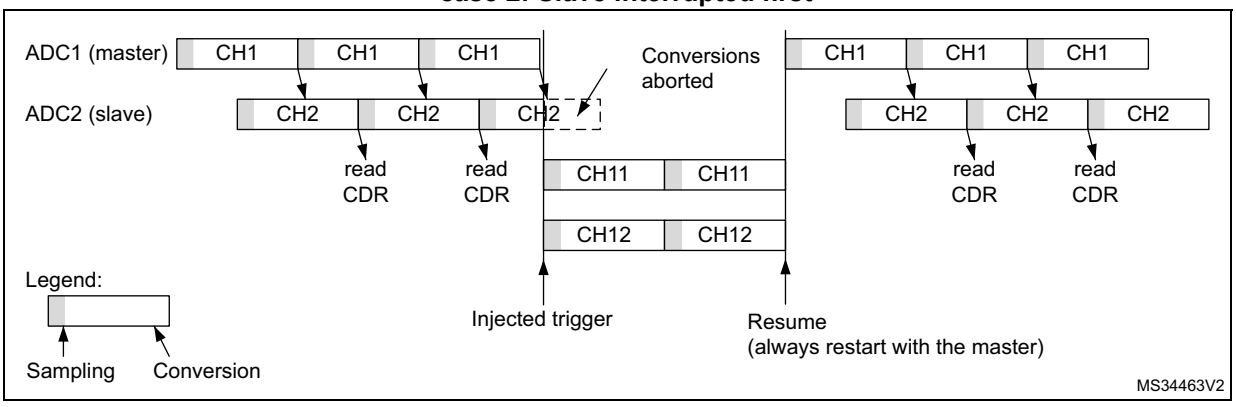


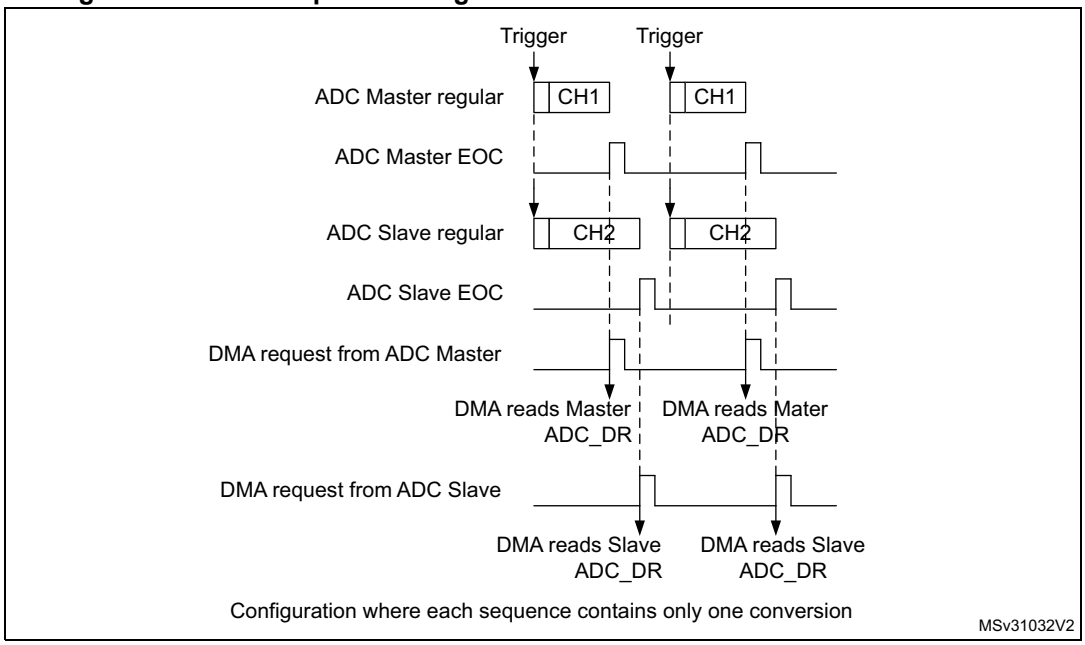
Figure 194. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 195: DMA Requests in regular simultaneous mode when DAMDF=0b00*).

Figure 195. DMA Requests in regular simultaneous mode when DAMDF=0b00



In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this DAMDF bits must be configured in the ADCx_CCR register:

- **DAMDF=0b10, 32-bit format**: A single DMA request is generated alternatively when either the master or slave EOC events have occurred. At that time, the data items are alternatively available in the ADCx_CDR2 32-bit register. This mode is used in interleaved mode and in regular simultaneous mode when resolution is above 16-bit.
  Example:

  Interleaved dual mode: a DMA request is generated each time a new 32-bit data is available:

  1st DMA request: ADCx_CDR2[31:0] = MST_ADC_DR[31:0]

  2nd DMA request: ADCx_CDR2[31:0] = SLV_ADC_DR[31:0]
- **DAMDF=0b10, 16-bit format**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCx_CDR contains the two half-words representing two ADC-

converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.

This mode is used in interleaved mode and in regular simultaneous mode when resolution is ranging from 10 to 16-bit. Any value above 16-bit in the master or the slave converter will be truncated to the least 16 significant bits.

Example:

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

2nd DMA request: ADCx_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 196. DMA requests in regular simultaneous mode when DAMDF=0b10

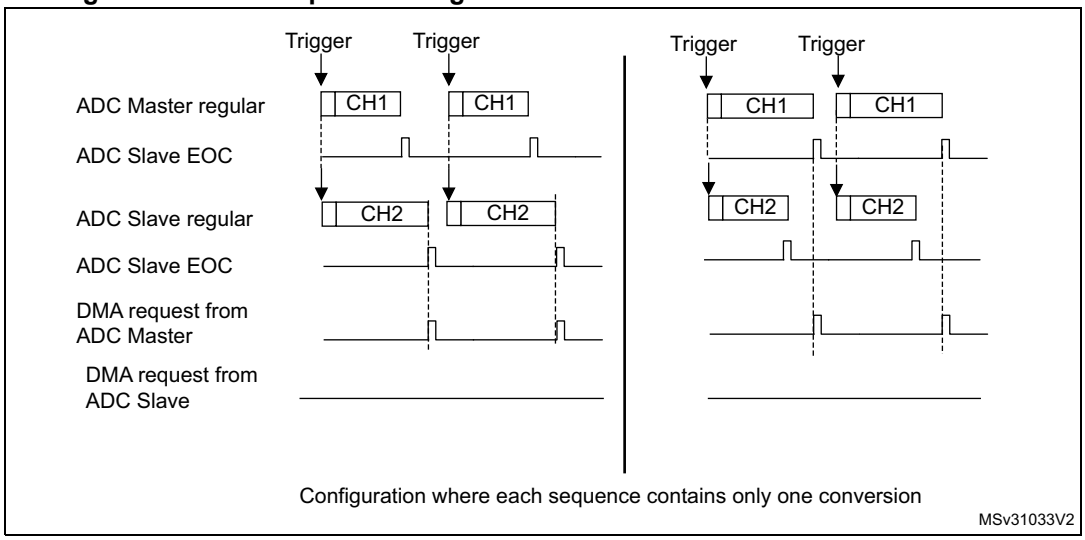
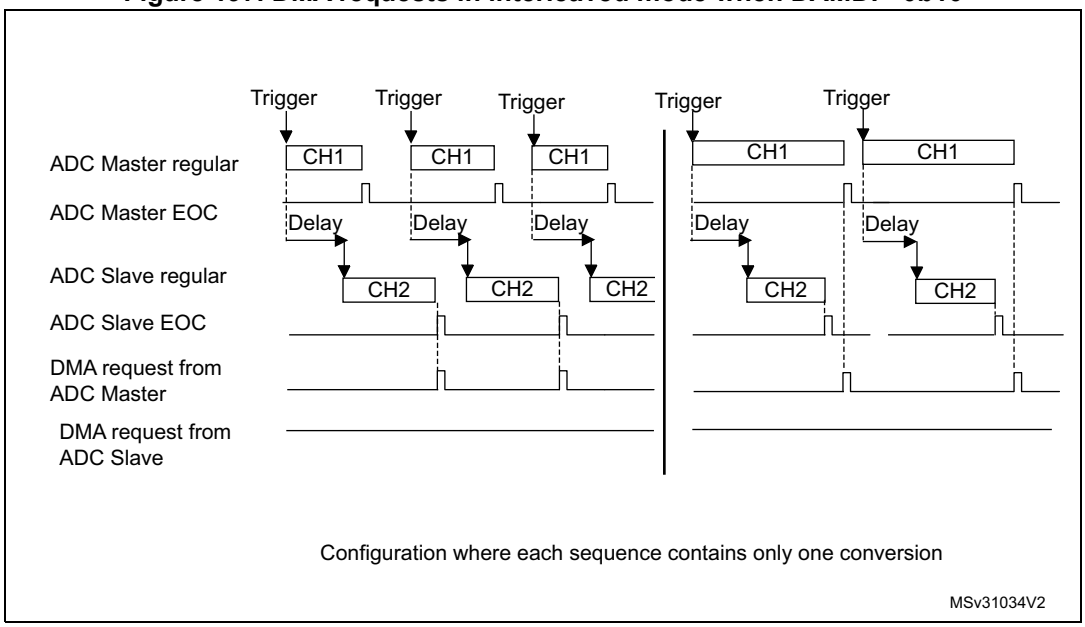


Figure 197. DMA requests in interleaved mode when DAMDF=0b10



Note: When using Multi ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **DAMDF=0b11**: This mode is similar to the DAMDF=0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.

  This mode is used in interleaved and regular simultaneous mode when the result is 8-bit. A new DMA request is issued when 4 new 8-bit values are available.

  Example:

  Interleaved dual mode: a DMA request is generated each time 4 data items are available (t0, t1,... are corresponding to the consecutive sampling instants)

  1st DMA request:

  ADCx_CDR[7:0] = MST_ADC_DR[7:0]$_{t0}$

  ADCx_CDR[15:8] = SLV_ADC_DR[7:0]$_{t0}$

  ADCx_CDR[23:16] = MST_ADC_DR[7:0]$_{t1}$

  ADCx_CDR[31:24] = SLV_ADC_DR[7:0]$_{t1}$

  2nd DMA request:

  ADCx_CDR[7:0] = MST_ADC_DR[7:0]$_{t2}$

  ADCx_CDR[15:8] = SLV_ADC_DR[7:0]$_{t2}$

  ADCx_CDR[23:16] = MST_ADC_DR[7:0]$_{t3}$

  ADCx_CDR[31:24] = SLV_ADC_DR[7:0]$_{t3}$

Overrun detection

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

DMA one shot mode/ DMA circular mode when Multi ADC mode is selected

When DAMDF mode is selected (0b10 or 0b11), bit DMNGT[1:0]=0b10 in the master ADC's ADCx_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*.

Stopping the conversions in dual ADC modes

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

DFSDM mode in dual ADC mode interleaved mode

In dual ADC interleaved modes, the ADC conversion results can be transferred directly to the Digital Filter for Sigma Delta Modulators (DFSDM).

This mode is enabled by setting the bits DMNGT[1:0] = 0b10 in the master ADC's ADC_CFGR register.

The ADC transfers alternatively the 16 least significant bits of the regular data register from the master and the slave converter to a single channel of the DFSDM.

The data format must be 16-bit signed:

ADC_DR[31:16] = 0x0000

ADC_DR[15] = sign

ADC_DR[14:0] = data

Any value above 16-bit signed format in any converter will be truncated.

DFSDM mode in dual ADC simultaneous mode

The dual mode is not required to use DFSDM in dual ADC simultaneous mode since conversion data will be treated by each individual channel. Single mode with same trigger source results in simultaneous conversion with DFSDM interface.

24.3.33 Temperature sensor

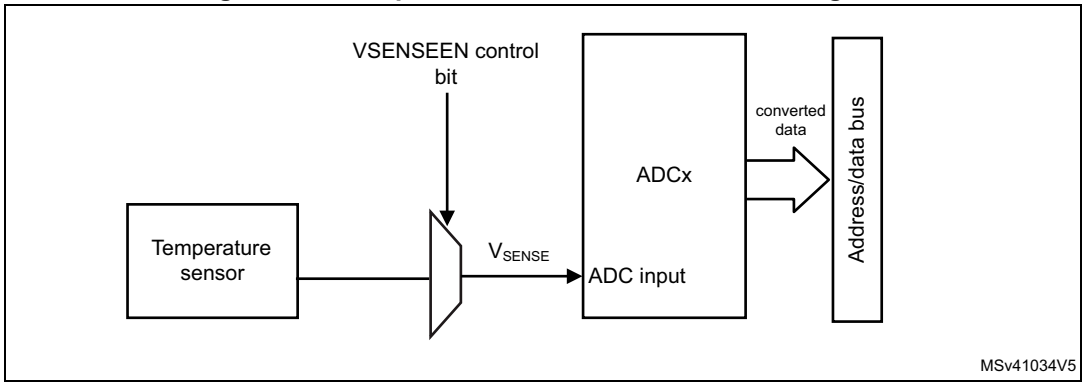
The temperature sensor can measure the junction temperature (T_J) of the device in the –40 to 125 °C temperature range.

The temperature sensor is internally connected to ADC3 V_{INP}[18] input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor's analog pin must be greater than the stabilization time specified in the product datasheet.

When not in use, the sensor can be put in power-down mode.

Figure 198 shows the block diagram of the temperature sensor.

Figure 198. Temperature sensor channel block diagram



Note: The VSENSEEN bit must be set to enable the conversion of internal channel ADC3 V_{INP}[18] (temperature sensor, V_{SENSE}).

Reading the temperature

To use the sensor:

1. Select the ADC3 V_{INP}[18] input channels (with the appropriate sampling time).
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the VSENSEEN bit in the ADCx_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{110\ °C - 30\ °C}{TS_CAL2 - TS_CAL1} \times (TS_DATA - TS_CAL1) + 30\ °C$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at 110°C
- TS_CAL1 is the temperature sensor calibration value acquired at 30°C
- TS_DATA is the actual temperature sensor output value converted by ADC

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and SENSEEN bits should be set at the same time.

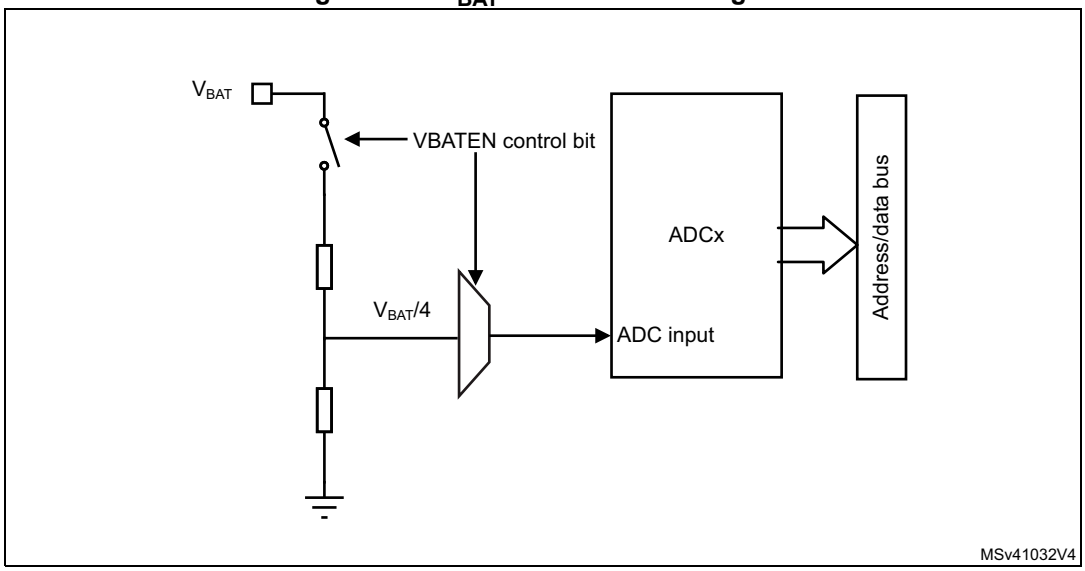
24.3.34 V_{BAT} supply monitoring

The VBATEN bit in the ADCx_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the ADC3 V_{INP}[17]input channels. As a consequence, the converted digital value is one fourth of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 199 shows the block diagram of the V_{BAT} sensing feature.

Figure 199. V_{BAT} channel block diagram



Note: The VBATEN bit must be set to enable the conversion of internal channels ADC3 V_{INP}[17] (V_{BAT}/4).

24.3.35 Monitoring the internal voltage reference

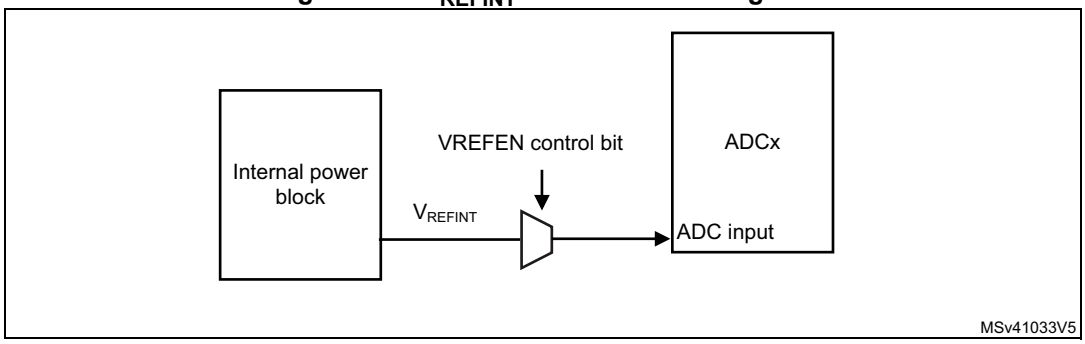
The internal voltage reference can be monitored to have a reference point for evaluating the ADC V_{REF+} voltage level.

The internal voltage reference is internally connected to the input channel ADC3 V_{INP}[19].

The sampling time for this channel must be greater than the stabilization time specified in the product datasheet.

Figure 199 shows the block diagram of the V_{REFINT} sensing feature.

Figure 200. V_{REFINT} channel block diagram



Note: The VREFEN bit into ADCx_CCR register must be set to enable the conversion of internal channels ADC3 V_{INP}[19] (V_{REFINT}).

Calculating the actual V_{DDA} voltage using the internal reference voltage

The power supply voltage applied to the microcontroller may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{DDA} = 3.3 V can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{DDA} voltage supplying the device:

$$V_{REF+} = 3.3\ V \times VREFINT_CAL / VREFINT_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For most applications V_{DDA} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{DDA}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{3.3\ V \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

For applications where V_{DDA} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL_SCALE} \times ADC_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- ADC_DATA is the value measured by the ADC on channel x (right-aligned)
- VREFINT_DATA is the actual VREFINT output value converted by the ADC
- FULL_SCALE is the maximum digital value of the ADC output. For example with 16-bit resolution, it will be 2^{16} - 1 = 65535 or with 8-bit resolution, 2^8 - 1 = 255.

Note: If ADC measurements are done using an output format other than 16-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

24.4 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (OVSR flag)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 210. ADC interrupts per each ADC

Interrupt event	Event flag	Enable control bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular group	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
End of conversion of a injected group	JEOC	JEOCIE
End of sequence of conversions of an injected group	JEOS	JEOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVSR	OVRIE
Injected context queue overflows	JQOVF	JQOVFIE

24.5 ADC registers (for each ADC)

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

24.5.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVSR	EOS	EOC	EOSMP	ADRDY
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 24.3.22: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 OVSR: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

24.5.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART is cleared to 0 (no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVSR bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	LINCALRDYW6	LINCALRDYW5	LINCALRDYW4	LINCALRDYW3	LINCALRDYW2	LINCALRDYW1	Res.	Res.	Res.	Res.	Res.	ADCALLIN
rs	rw	rw	rw	rw	rw	rw	rw	rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOST	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
							rw			rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN=0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL will launch a calibration in Single-ended inputs Mode.

1: Writing ADCAL will launch a calibration in Differential inputs Mode.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in deep-power-down mode.

0: ADC not in deep-power down

1: ADC in deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 24.3.6: ADC deep-power-down mode (DEEPPWD) & ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 27 **LINCALRDYW6**: Linearity calibration ready Word 6

This control / status bit allows to read/write the 6th linearity calibration factor.

When the linearity calibration is complete, this bit is set. A bit clear will launch the transfer of the linearity factor 6 into the LINCALFACT[29:0] of the ADC_CALFACT2 register. The bit will be reset by hardware when the ADC_CALFACT2 register can be read (software must poll the bit until it is cleared).

When the *LINCALRDYW6 bit is reset, a new linearity factor 6 value can be written into* the LINCALFACT[29:0] of the ADC_CALFACT2 register. A bit set will launch the linearity factor 6 update and the bit will be effectively set by hardware once the update will be done (software must poll the bit until it is set to indicate the write is effective).

Note: ADC_CALFACT2[29:10] *contains 0.* ADC_CALFACT2[9:0] *corresponds linearity correction factor bits[159:150].*

The software is allowed to toggle this bit only if the LINCALRDYW5, LINCALRDYW4, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged, see chapter 24.3.8: Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT) for details.

The software is allowed to update the linearity calibration factor by writing LINCALRDYWx only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 26 **LINCALRDYW5**: Linearity calibration ready Word 5

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *corresponds linearity correction factor bits[149:120].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 25 **LINCALRDYW4**: Linearity calibration ready Word 4

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *correspond linearity correction factor bits[119:90].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 24 **LINCALRDYW3**: Linearity calibration ready Word 3

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *corresponds linearity correction factor bits[89:60].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 23 **LINCALRDYW2**: Linearity calibration ready Word 2

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *corresponds linearity correction factor bits[59:30].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW3 and LINCALRDYW1 bits are left unchanged.

Bit 22 **LINCALRDYW1**: Linearity calibration ready Word 1

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] corresponds linearity correction factor bits[29:0].

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW3 and LINCALRDYW2 bits are left unchanged.

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **ADCALLIN**: Linearity calibration

This bit is set and cleared by software to enable the Linearity calibration.

0: Writing ADCAL will launch a calibration without the Linearity calibration.

1: Writing ADCAL will launch a calibration with he Linearity calibration.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOST**: Boost mode control

This bit is set and cleared by software to enable/disable the Boost mode.

0: Boost mode off. Used when ADC clock < 20 MHz to save power at lower clock frequency.

1: Boost mode on. Must be used when ADC clock > 20 MHz.

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

When dual mode is enabled (bits DAMDF of ADCx_CCR register are not equal to zero), the bit BOOST of the slave ADC is no more writable and its content must be equal to the master ADC BOOST bit.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command). It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set JADSTP only when JADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command). It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set ADSTP only when ADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

In dual ADC regular simultaneous mode and interleaved mode, the bit ADSTP of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion will start immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (JEXTSEL=0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.
- in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion will start immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode (CONT=0, DISCEN=0) when software trigger is selected (EXTEN=0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- In discontinuous conversion mode (CONT=0, DISCEN=1), when the software trigger is selected (EXTEN=0x0): at the end of conversion (EOC) flag.
- in all other cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN=1 and both ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)

24.5.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					RES[2:0]			DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **JQDIS**: Injected Queue disable

These bits are set and cleared by software to disable the Injected Queue mechanism:

0: Injected Queue enabled

1: Injected Queue disabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel-0 monitored by AWD1

00001: ADC analog input channel-1 monitored by AWD1

.....

10010: ADC analog input channel-19 monitored by AWD1

others: Reserved, must not be used

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: JSQR Mode 0: The Queue is never empty and maintains the last written configuration into JSQR.

1: JSQR Mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 24.3.22: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

When dual mode is enabled (bits DAMDF of ADCx_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADCx_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun Mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: Event 0
00001: Event 1
00010: Event 2
00011: Event 3
00100: Event 4
00101: Event 5
00110: Event 6
00111: Event 7
...
11111: Event 31

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 4:2 **RES[2:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

000: 16 bits
001: 14 bits
010: 12 bits
011: 10 bits
100: 8-bit
Others: Reserved.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data Management configuration

This bit is set and cleared by software to select how ADC interface output data are managed.

00: Regular conversion data stored in DR only
01: DMA One Shot Mode selected
10: DFSDM mode selected
11: DMA Circular Mode selected

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bit DAMDF of the ADCx_CCR register.

24.5.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RSHIFT4	RSHIFT3	RSHIFT2	RSHIFT1	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: Shift left 1-bit
0010: Shift left 2-bits
0011: Shift left 3-bits
0100: Shift left 4-bits
0101: Shift left 5-bits
0110: Shift left 6-bits
0111: Shift left 7-bits
1000: Shift left 8-bits
1001: Shift left 9-bits
1010: Shift left 10-bits
1011: Shift left 11-bits
1100: Shift left 12-bits
1101: Shift left 13-bits
1101: Shift left 14-bits
1111: Shift left 15-bits

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OVSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RSHIFT4**: Right-shift data after Offset 4 correction

Refer to RSHIFT1 description.

Bit 13 **RSHIFT3**: Right-shift data after Offset 3 correction
Refer to RSHIFT1 description

Bit 12 **RSHIFT2**: Right-shift data after Offset 2 correction
Refer to RSHIFT1 description

Bit 11 **RSHIFT1**: Right-shift data after Offset 1 correction
This bitfield is set and cleared by software to right-shift 1-bit data after offset1 correction. This bit can only be used for 8-bit and 16-bit data format (see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)* for details).
0: Right-shifting disabled
1: Data is right-shifted 1-bit.

Bit 10 **ROVSM**: Regular Oversampling mode
This bit is set and cleared by software to select the regular oversampling mode.
0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)
1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)
Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 9 **TROVS**: Triggered Regular Oversampling
This bit is set and cleared by software to enable triggered oversampling
0: All oversampled conversions for a channel are done consecutively following a trigger
1: Each oversampled conversion for a channel needs a new trigger
Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling right shift
This bit field is set and cleared by software to define the right shifting applied to the raw oversampling result.
0000: No right shift
0001: Shift right 1-bit
0010: Shift right 2-bits
0011: Shift right 3-bits
0100: Shift right 4-bits
0101: Shift right 5-bits
0110: Shift right 6-bits
0111: Shift right 7-bits
1000: Shift right 8-bits
1001: Shift right 9-bits
1010: Shift right 10-bits
1011: Shift right 11-bits
Others: Reserved.
Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **JOVSE**: Injected Oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected Oversampling disabled

1: Injected Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular Oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular Oversampling disabled

1: Regular Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

24.5.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMP[9:0][2:0]:** Channel x sampling time selection (x = 0 to 9)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles

001: 2.5 ADC clock cycles

010: 8.5 ADC clock cycles

011: 16.5 ADC clock cycles

100: 32.5 ADC clock cycles

101: 64.5 ADC clock cycles

110: 387.5.5 ADC clock cycles

111: 810.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMP[9:0][2:0]:** Channel x sampling time selection (x = 0 to 9)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 8.5 ADC clock cycles
011: 16.5 ADC clock cycles
100: 32.5 ADC clock cycles
101: 64.5 ADC clock cycles
110: 387.5 ADC clock cycles
111: 810.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL19	PCSEL18	PCSEL17	PCSEL16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCSEL 15	PCSEL 14	PCSEL 13	PCSEL 12	PCSEL 11	PCSEL 10	PCSEL9	PCSEL8	PCSEL7	PCSEL6	PCSEL5	PCSEL4	PCSEL3	PCSEL2	PCSEL1	PCSEL0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 PCSELx: Channel x ($V_{INP[i]}$) pre selection

These bits are written by software to pre select the input channel at IO instance to be converted.

0: Input Channel x (Vinp x) is not pre selcted for conversion, the ADC conversion result with this channel shows wrong result.

1: Input Channel x (Vinp x) is pre selcted for conversion

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.9 ADC watchdog threshold register 1 (ADC_LTR1)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LTR1[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR1[25:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 24.3.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.10 ADC watchdog threshold register 1 (ADC_HTR1)

Address offset: 0x24

Reset value: 0x03FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HTR1[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **HTR1[25:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 24.3.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 4th in the regular conversion sequence.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion

0001: 2 conversions

...

1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

24.5.12 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 8th in the regular conversion sequence

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 7th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

24.5.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 11th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 10th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

24.5.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

24.5.15 ADC regular Data Register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA[31:0]**: Regular Data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 24.3.27: Data management*.

24.5.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bits 8:7 **JEXTEN[1:0]**: External Trigger Enable and Polarity Selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS=0 (queue enabled), Hardware and software trigger detection disabled

00: If JQDIS=1 (queue disabled), Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

If JQM=1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 24.3.22: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: Event 0

00001: Event 1

00010: Event 2

00011: Event 3

00100: Event 4

00101: Event 5

00110: Event 6

00111: Event 7

...

11111: Event 31:

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

24.5.17 ADC offset register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y-1), y= 1 to 4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSATE	OFFSETy_CH[4:0]					OFFSETy[25:16]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSETy[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SSATE:** Signed saturation Enable

This bit is written by software to enable or disable the Signed saturation feature.

This bit can be enabled only for 8-bit and 16-bit data format (see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)* for details).

0: Offset is subtracted maintaining data integrity and extending result size (9-bit and 17-bit signed format).

1: Offset is subtracted and result is saturated to maintain result size.

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSETy_CH[4:0]:** Channel selection for the Data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[25:0] will apply.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 25:0 **OFFSETy[25:0]:** Data offset y for the channel programmed into bits OFFSETy_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

When OFFSETy[25:0] bitfield is reset, the offset compensation is disabled.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.
Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[25:0] which is subtracted when converting channel 4.

24.5.18 ADC injected data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y-1), y= 1 to 4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 24.3.27: Data management*.

24.5.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD2CH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel-i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel-i is monitored by AWD2

When AWD2CH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD3CH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel-i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel-i is monitored by AWD3

When AWD3CH[19:0] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers. The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.21 ADC watchdog lower threshold register 2 (ADC_LTR2)

Address offset: 0xB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LTR2[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR2[25:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to *Section 24.3.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.22 ADC watchdog higher threshold register 2 (ADC_HTR2)

Address offset: 0xB4

Reset value: 0x03FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HTR2[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **HTR2[25:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to *Section 24.3.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.23 ADC watchdog lower threshold register 3 (ADC_LTR3)

Address offset: 0xB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LTR3[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR3[25:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

Refer to *Section 24.3.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.24 ADC watchdog higher threshold register 3 (ADC_HTR3)

Address offset: 0xBC

Reset value: 0x03FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HTR3[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **HTR3[25:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 24.3.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

24.5.25 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0xC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]**: Differential mode for channels 19 to 0

These bits are set and cleared by software. They allow to select if a channel is configured as single ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel-i is configured in single ended mode

DIFSEL[i] = 1: ADC analog input channel-i is configured in differential mode

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

24.5.26 ADC calibration factors register (ADC_CALFACT)

Address offset: 0xC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CALFACT_D[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CALFACT_S[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **CALFACT_D[10:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new differential calibration is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **CALFACT_S[10:0]**: Calibration Factors In Single-Ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

24.5.27 ADC calibration factor register 2 (ADC_CALFACT2)

Address offset: 0xC8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	LINCALFACT[29:16]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINCALFACT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **LINCALFACT[29:0]**: Linearity Calibration Factor

These bits are written by hardware or by software.

They hold 30-bit out of the 160-bit linearity calibration factor.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it will then be applied once a new single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

24.6 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

24.6.1 ADC x common status register (ADCx_CSR) (x=1/2 or 3)

Address offset: 0x00 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

ADC1 and ADC2 are controlled by the same interface, while ADC3 is controlled separately.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected Context Queue Overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADCx+1_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADCx+1_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADCx+1_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADCx+1_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADCx+1_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADCx+1_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVSR bit in the corresponding ADCx+1_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC
This bit is a copy of the EOS bit in the corresponding ADCx+1_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADCx+1_ISR register.

Bit 17 **EOSMP_SLV:** End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADCx+1_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADCx+1_ISR register.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVSR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

24.6.2 ADC x common control register (ADCx_CCR) (x=1/2 or 3)

Address offset: 0x08 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

ADC1 and ADC2 are controlled by the same interface, while ADC3 is controlled separately.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	VSENSEEN	VREFEN	PRESC[3:0]				CKMODE[1:0]	
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw			rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable

This bit is set and cleared by software to control V_{BAT} channel.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 23 **VSENSEEN**: Temperature sensor voltage enable

This bit is set and cleared by software to control V_{SENSE} channel.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
Others: Reserved.

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: CK_ADCx (x=1 to 3) (Asynchronous clock mode), generated at product level (refer to *Section Reset and Clock Control (RCC)*)
01: adc_hclk/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0xxx in RCC_CFGR register) and if the system clock has a 50% duty cycle.
10: adc_hclk/2 (Synchronous clock mode)
11: adc_hclk/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 15:14 **DAMDF[1:0]:** Dual ADC Mode Data Format

This bit-field is set and cleared by software. It specifies the data format in the common data register ADCx_CDR.

00: Dual ADC mode without data packing (ADCx_CDR and ADCx_CDR2 registers not used).
01: Reserved.
10: Data formatting mode for 32 down to 10-bit resolution
11: Data formatting mode for 8-bit resolution

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 211* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.
All the ADCs independent:
00000: Independent mode

00001 to 01001: Dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Combined Interleaved mode + injected simultaneous mode
00100: Reserved.
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Table 211. DELAY bits versus ADC resolution

DELAY bits	16-bit resolution	14-bit resolution	12-bit resolution	10-bit resolution	8-bit resolution
0000	1.5 * $T_{adc_ker_ck}$	1.5 * $T_{adc_ker_ck}$	1.5 * $T_{adc_ker_ck}$	1.5 * $T_{adc_ker_ck}$	1.5 * $T_{adc_ker_ck}$
0001	2.5 * $T_{adc_ker_ck}$	2.5 * $T_{adc_ker_ck}$	2.5 * $T_{adc_ker_ck}$	2.5 * $T_{adc_ker_ck}$	2.5 * $T_{adc_ker_ck}$
0010	3.5 * $T_{adc_ker_ck}$	3.5 * $T_{adc_ker_ck}$	3.5 * $T_{adc_ker_ck}$	3.5 * $T_{adc_ker_ck}$	3.5 * $T_{adc_ker_ck}$
0011	4.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$
0100	5.5 * $T_{adc_ker_ck}$	5.5 * $T_{adc_ker_ck}$	5.5 * $T_{adc_ker_ck}$	5.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$
0101	6.5 * $T_{adc_ker_ck}$	6.5 * $T_{adc_ker_ck}$	6.5 * $T_{adc_ker_ck}$	5.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$
0110	7.5 * $T_{adc_ker_ck}$	7.5 * $T_{adc_ker_ck}$	6.5 * $T_{adc_ker_c}$	5.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$
0111	8.5 * $T_{adc_ker_ck}$	7.5 * $T_{adc_ker_ck}$	6.5 * $T_{adc_ker_ck}$	5.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$
1000	8.5 * $T_{adc_ker_ck}$	7.5 * $T_{adc_ker_ck}$	6.5 * $T_{adc_ker_ck}$	5.5 * $T_{adc_ker_ck}$	4.5 * $T_{adc_ker_ck}$
others: reserved	-	-	-	-	-

24.6.3 ADC x common regular data register for dual mode (ADCx_CDR) (x=1/2 or 3)

Address offset: 0x0C (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

ADC1 and ADC2 are controlled by the same interface, while ADC3 is controlled separately.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_SLV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_MST[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 24.3.32: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 24.3.32: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

24.6.4 ADC x common regular data register for 32-bit dual mode (ADCx_CDR2) (x=1/2 or 3)

Address offset: 0x10 (this offset address is relative to the master ADC base address + 0x300)

Reset value: 0x0000 0000

ADC1 and ADC2 are controlled by the same interface, while ADC3 is controlled separately.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_ALT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_ALT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA_ALT[31:0]**: Regular data of the master/slave alternated ADCs

In dual mode, these bits alternatively contains the regular 32-bit data of the master and the slave ADC. Refer to *Section 24.3.32: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)*.

24.6.5 ADC register map

The following table summarizes the ADC registers.

Table 212. ADC global register map

Offset	Register
0x000 - 0x0D0	Master ADC1 or Master ADC3
0x0D4 - 0x0FC	Reserved
0x100 - 0x1D0	Slave ADC2
0x1D4 - 0x2FC	Reserved
0x300 - 0x310	Master and slave ADCs common registers (ADC1/2 or ADC3)

Table 213. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVSR	EOS	EOC	EOSMP	ADRDY
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x04	**ADC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR**	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	LINCALRDYW6	LINCALRDYW5	LINCALRDYW4	LINCALRDYW3	LINCALRDYW2	LINCALRDYW1	Res.	Res.	Res.	Res.	Res.	ADCALLIN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOST	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0	0	0	0	0	0	0						0							0	0			0	0	0	0	0	0
0x0C	**ADC_CFGR**	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM [2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL [4:0]					RES [2:0]			DMN GT [1:0]	
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**ADC_CFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	OSR[9:0]										Res.	RSHIFT4	RSHIFT3	RSHIFT2	RSHIFT1	ROVSM	TROVS	OVSR[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
	Reset value							0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0				0	0
0x14	**ADC_SMPR1**	Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**ADC_SMPR2**	Res.	Res.	Res.	Res.	Res.	SMP18 [2:0]			SMP17 [2:0]			SMP16 [2:0]			SMP15 [2:0]			SMP14 [2:0]			SMP13 [2:0]			SMP12 [2:0]			SMP11 [2:0]			SMP10 [2:0]	
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 213. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**ADC_PCSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL19	PCSEL18	PCSEL17	PCSEL16	PCSEL15	PCSEL14	PCSEL13	PCSEL12	PCSEL11	PCSEL10	PCSEL9	PCSEL8	PCSEL7	PCSEL6	PCSEL5	PCSEL4	PCSEL3	PCSEL2	PCSEL1	PCSEL0
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**ADC_LTR1**	Res.	Res.	Res.	Res.	Res.	Res.	LTR1[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_HTR1**	Res.	Res.	Res.	Res.	Res.	Res.	HTR1[25:0]																									
	Reset value							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x28	Reserved	Res.																															
0x2C	Reserved	Res.																															
0x30	**ADC_SQR1**	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x34	**ADC_SQR2**	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	Res.	SQ5[4:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0	0
0x38	**ADC_SQR3**	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	Res.	SQ10[4:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0	0
0x3C	**ADC_SQR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x40	**ADC_DR**	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44-0x48	Reserved	Res.																															
0x4C	**ADC_JSQR**	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]				Res.	JSQ1[4:0]						JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50-0x5C	Reserved	Res.																															
0x60	**ADC_OFR1**	SSATE	OFFSET1_CH[4:0]					OFFSET1[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	**ADC_OFR2**	SSATE	OFFSET2_CH[4:0]					OFFSET2[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	**ADC_OFR3**	SSATE	OFFSET3_CH[4:0]					OFFSET3[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 213. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6C	**ADC_OFR4**	SSATE	OFFSET4_CH[4:0]					OFFSET4[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70-0x7C	Reserved	Res.																															
0x80	**ADC_JDR1**	JDATA1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	**ADC_JDR2**	JDATA2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	**ADC_JDR3**	JDATA3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	**ADC_JDR4**	JDATA4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C-0x9C	Reserved	Res.																															
0xA0	**ADC_AWD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	**ADC_AWD3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8-0xAC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xB0	**ADC_LTR2**	Res.	Res.	Res.	Res.	Res.	Res.	LTR2[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4	**ADC_HTR2**	Res.	Res.	Res.	Res.	Res.	Res.	HTR2[25:0]																									
	Reset value							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0xB8	**ADC_LTR3**	Res.	Res.	Res.	Res.	Res.	Res.	LTR3[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	**ADC_HTR3**	Res.	Res.	Res.	Res.	Res.	Res.	HTR3[25:0]																									
	Reset value							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0xC0	**ADC_DIFSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC4	**ADC_CALFACT**	Res.	Res.	Res.	Res.	Res.	CALFACT_D[10:0]											Res.	Res.	Res.	Res.	Res.	CALFACT_S[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xC8	**ADC_CALFACT2**	Res.	Res.	LINCALFACT[29:0]																													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0D	Reserved																																

Table 214. ADC register map and reset values (master and slave ADC common registers) offset =0x300)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADCx_CSR**	Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
							slave ADC2																master ADC1										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Res.																															
0x08	**ADCx_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	VSENSEEN	VREFEN	PRESC[3:0]				CKMODE[1:0]		DAMDF[1:0]		DMACFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0				0	0	0	0	0
0x0C	**ADCx_CDR**	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**ADCx_CDR2**	RDATA_ALT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

25 Digital-to-analog converter (DAC)

25.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, V_{REF+} (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DAC_OUTx pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to allow a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and Hold mode.

25.2 DAC main features

The DAC main features are the following (see *Figure 201: Dual-channel DAC block diagram*)

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DAC_OUTx output pin
- DAC output connection to on chip peripherals
- Sample and Hold mode for low power operation in Stop mode
- Input voltage reference from V_{REF+} pin or internal VREFBUF reference

Figure 201 shows the block diagram of a DAC channel and *Table 216* gives the pin description.

25.3 DAC implementation

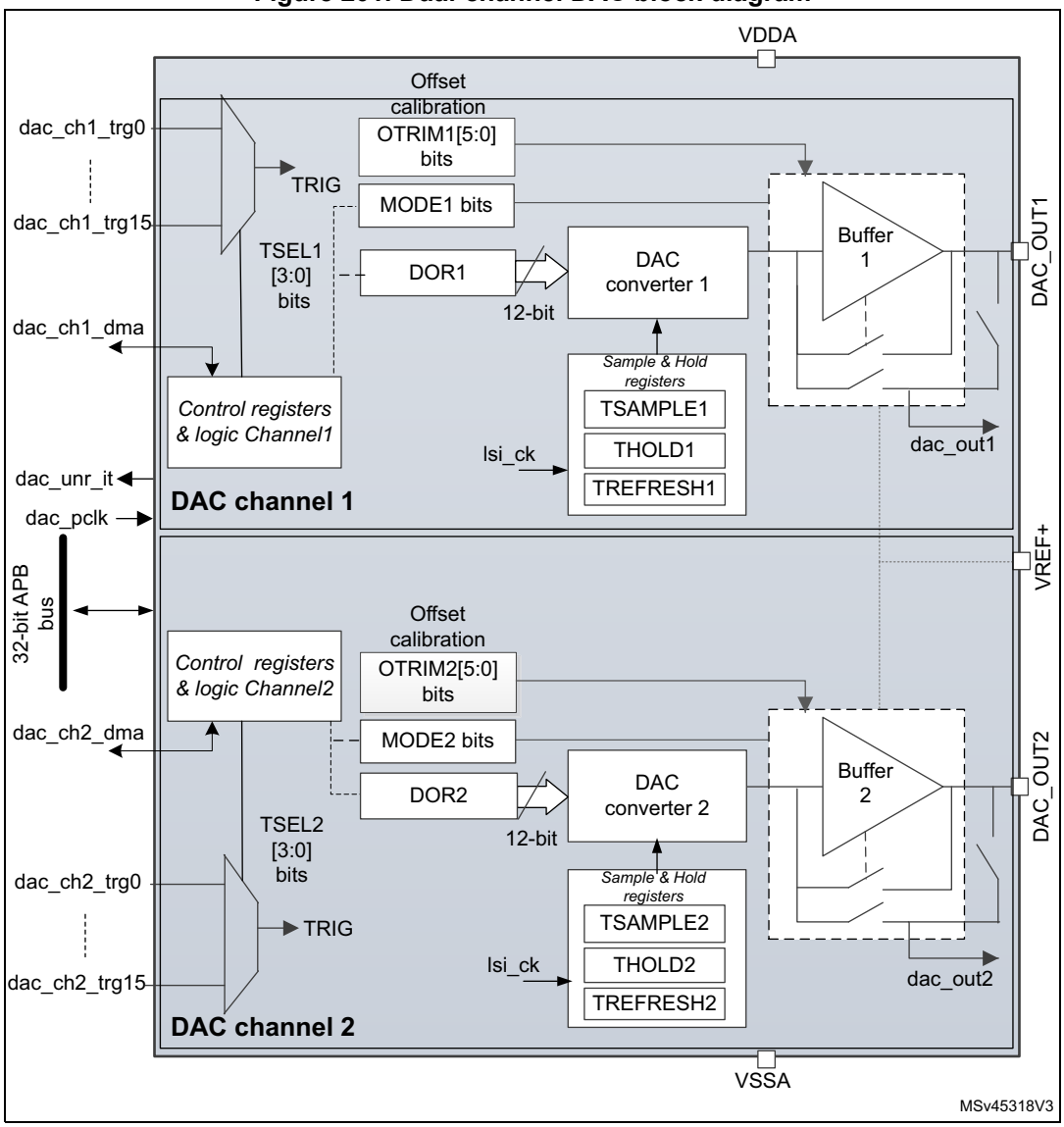
Table 215. DAC implementation

DAC features	DAC1
Dual channel	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5

25.4 DAC functional description

25.4.1 DAC block diagram

Figure 201. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the Normal mode in buffer/unbuffered configuration and the Sample and Hold mode.
2. Refer to *Section 25.3: DAC implementation* for channel 2 availability.

25.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DAC_OUTx can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and Hold block and registers operational in Stop mode, using LSI clock source for static conversion

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DAC_OUTx output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and Hold block and its associated registers can run in Stop mode using the LSI clock source.

Table 216. DAC input/output pins

Pin name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Table 217. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel 1 DMA request
dac_ch2_dma	Bidirectional	DAC channel 2 DMA request
dac_ch1_trg[0:15]	Inputs	DAC channel 1 trigger inputs
dac_ch2_trg[0:15]	Inputs	DAC channel 2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_pclk	Input	DAC peripheral clock
dac_out1	Analog output	DAC channel 1 output for on-chip peripherals
dac_out2	Analog output	DAC channel 2 output for on-chip peripherals

25.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

Note: The ENx bit enables the analog DAC Channelx only. The DAC Channelx digital interface is enabled even if the ENx bit is reset.

25.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

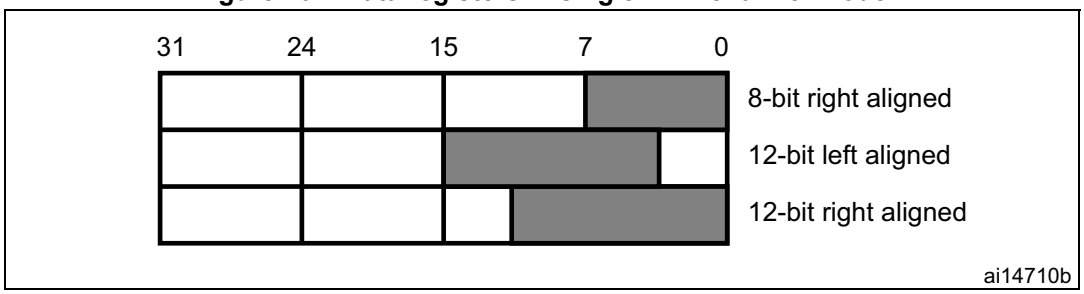
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

Figure 202. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

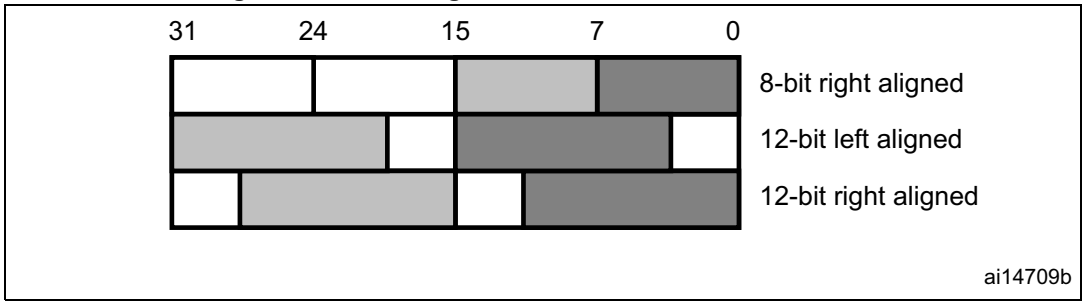
  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1

and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 203. Data registers in dual DAC channel mode



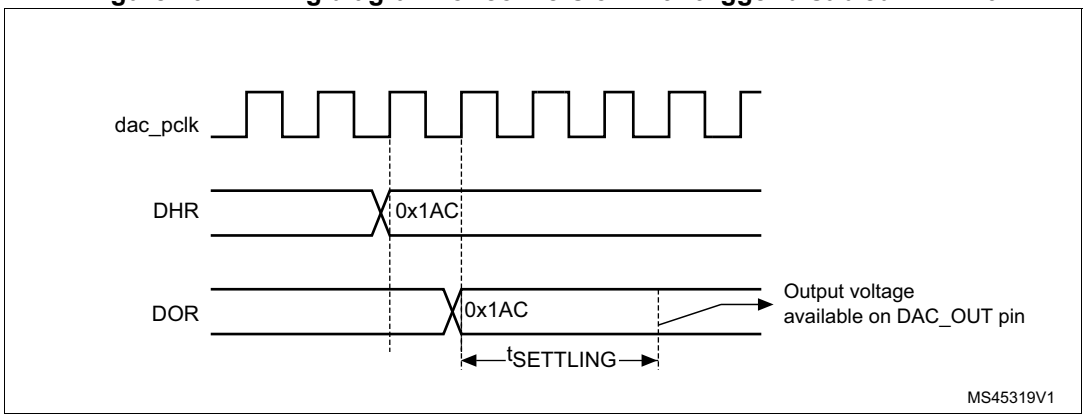
25.4.5 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_pclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_pclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 204. Timing diagram for conversion with trigger disabled TEN = 0



25.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

25.4.7 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events will trigger conversion as shown in bits TSEL1[3:0] and TSEL2[3:0] in *Table 218: DAC trigger selection*.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_pclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_pclk clock cycle.

Table 218. DAC trigger selection

Source	Type	TSELx[3:0]
SWTRIG	Software control bit	0000
TIM1_TRGO	Internal signal from on-chip timers	0001
TIM2_TRGO	Internal signal from on-chip timers	0010
TIM4_TRGO	Internal signal from on-chip timers	0011
TIM5_TRGO	Internal signal from on-chip timers	0100
TIM6_TRGO	Internal signal from on-chip timers	0101
TIM7_TRGO	Internal signal from on-chip timers	0110
TIM8_TRGO	Internal signal from on-chip timers	0111
TIM15_TRGO	Internal signal from on-chip timers	1000
HRTIM1_DACTRG1	Internal signal from on-chip timers	1001
HRTIM1_DACTRG2	Internal signal from on-chip timers	1010
LPTIM1_OUT	Internal signal from on-chip timers	1011
LPTIM2_OUT	Internal signal from on-chip timers	1100
EXTI9	External pin	1101
Reserved	-	1110
Reserved	-	1111

25.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit should be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

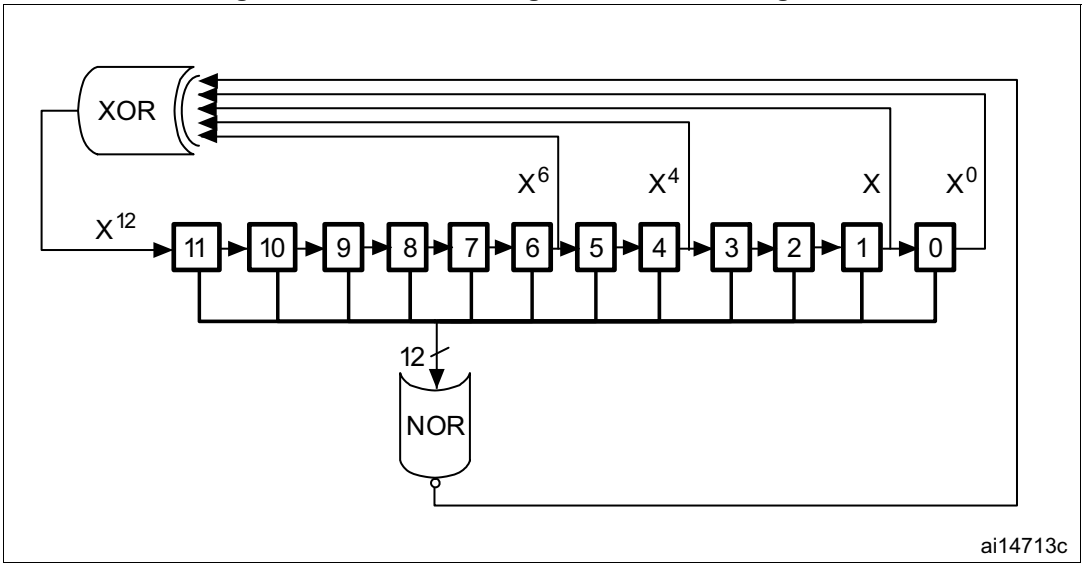
The software should clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

25.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01”. The preloaded value in LFSR is 0xAAA. This register is updated three dac_pclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 205. DAC LFSR register calculation algorithm

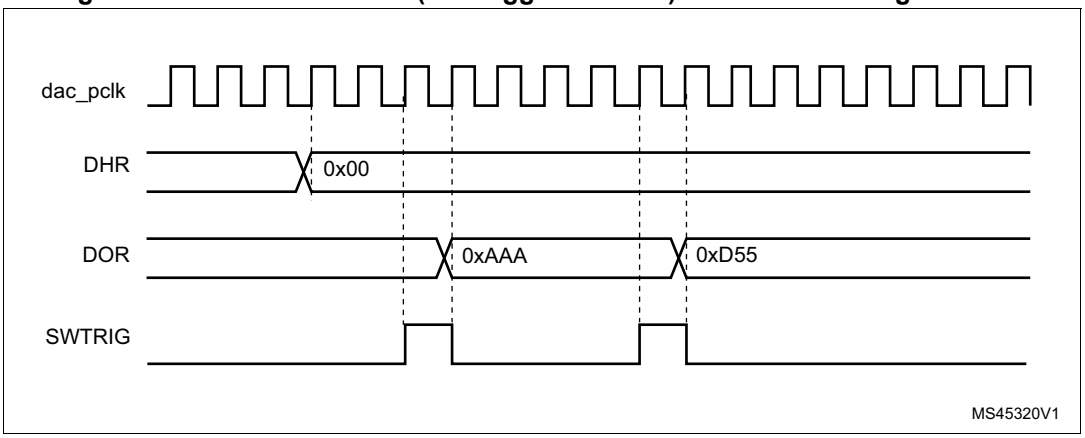


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 206. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

25.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_pclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 207. DAC triangle wave generation

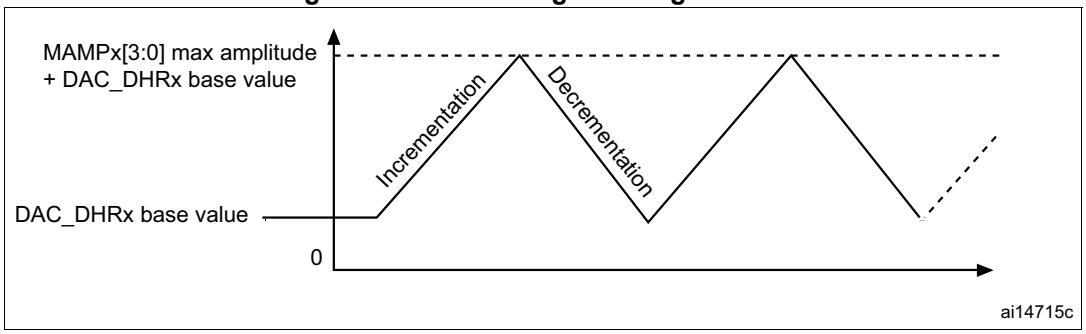
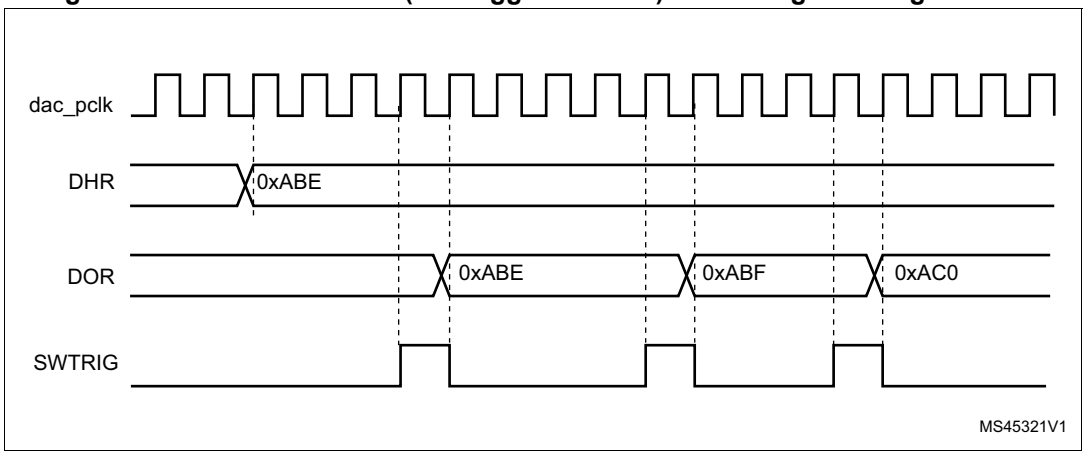


Figure 208. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

25.4.11 DAC channel modes

Each DAC channel can be configured in Normal mode or Sample and Hold mode. The output buffer can be enabled to allow a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DAC_OUTx pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register should be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register should be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and Hold mode

In sample and Hold mode, the DAC core converts data on a triggered conversion, then, holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A new stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the low-speed clock (LSI) in addition to the dac_pclk clock, allowing to use the DAC channels in deep low power modes such as Stop mode.

The sample/hold mode operations can be divided into 3 phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits; the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI clocks. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI ~32 KHz is selected:

12 cycles are required for sample phase: TSAMPLEx[9:0] = 11,

62 cycles are required for hold phase: THOLDx[9:0] = 62,

and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 219. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in Sample and Hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Please refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{DDA} = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs

(where N_{LSB} = 10 (10 LSB drop during the hold phase)

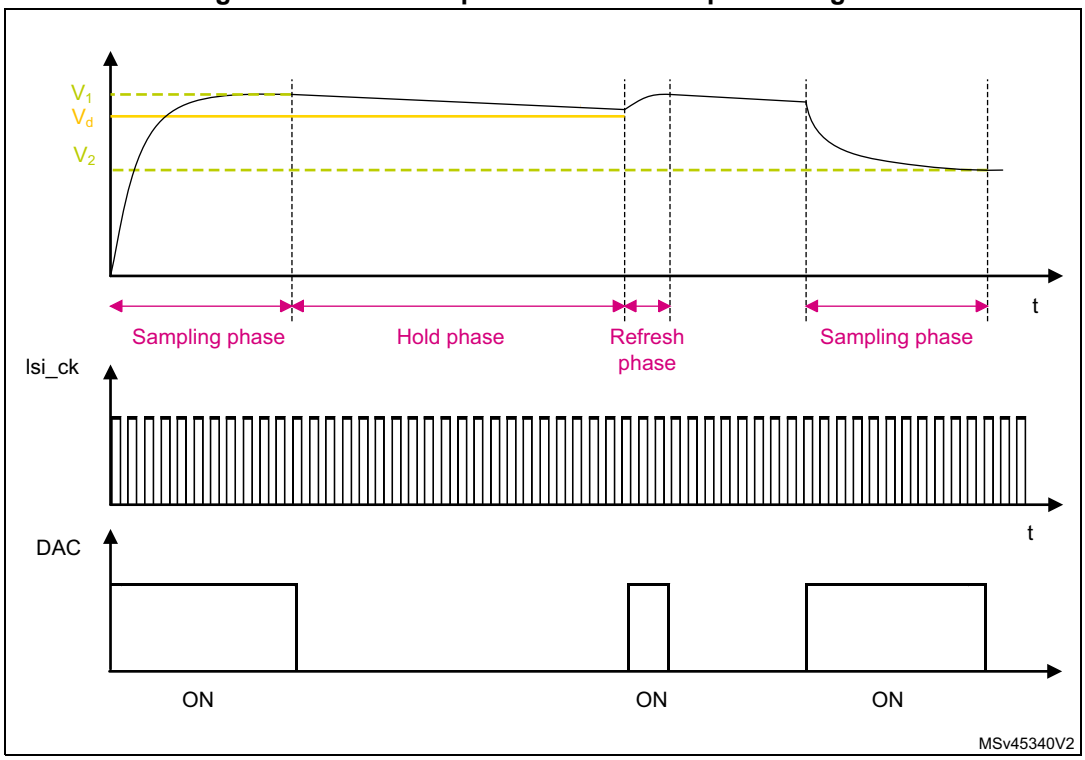
Hold phase:

D_v = i_{leak} * t_{hold} / C_{SH} = 0.0073 V (10 LSB of 12bit at 3 V)

i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)

t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 209. DAC Sample and Hold mode phase diagram



Like in Normal mode, the Sample and Hold mode has different configurations.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register should be:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, The MODEx[2:0] bits in DAC_MCR register should be:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

When MODEx[2:0] bits in DAC_MCR register is equal to 111. An internal capacitor, C_{Lint}, will hold the voltage output of the DAC Core and then drive it to on-chip peripherals.

All Sample and Hold phases are interruptible and any change in DAC_DHRx will trigger immediately a new sample phase.

Table 220. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (ex, comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (ex, comparators)

Table 220. Channel output modes summary (continued)

MODEx[2:0]			Mode	Buffer	Output connections
1	0	0	Sample and Hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (ex, comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (ex, comparators)
1	1	1			Connected to on chip peripherals (ex, comparators)

25.4.12 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D / 2^{N-1}) \times G \times V_{ref}) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{ref} is the nominal full-scale voltage, and V_{os} is the offset voltage. For an ideal DAC channel, G = 1 and V_{os} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{os}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 000b or 001b or 100b or 101b). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output will be disconnected from the pin internal/external connections and put in tristate mode (HiZ),
- The buffer will act as a comparator, to sense the middle-code value 0x800 and compare it to VREF+/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit)

Two calibration techniques are provided:

- Factory trimming (always enabled)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DD}/V_{DDA} voltage, temperature, VREF+ values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when V_{DD}/V_{DDA} is removed (example the device enters in STANDBY or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, Write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 000b or 001b or 100b or 101b,
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1,
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 00000b.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) if CAL_FLAGx is set to 1 the trimming code OTRIMx[4:0] is found and will be used during operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way,

The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If the V_{DD}/V_{DDA}, VREF+ and temperature conditions will not change during the device operation while it enters more often in standby and VBAT mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

25.4.13 Dual DAC channel conversion (if available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

11 possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[3:0] and TSEL2[3:0] bits.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_pclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_pclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[3:0] and TSEL2[3:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[3:0] and TSEL2[3:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[3:0] and TSEL2[3:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1[3:0] and TSEL2[3:0] bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

In this configuration, one dac_pclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[3:0] and TSEL2[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_pclk clock cycles).

Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[3:0] and TSEL2[3:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[3:0] and TSEL2[3:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[3:0] and TSEL2[3:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[3:0] and TSEL2[3:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the DAC channel2 triangle counter is updated.

25.5 DAC low-power modes

Table 221. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC can be used with DMA
Stop	DAC remains active with a static output value if Sample and Hold mode is selected using lsi_ck clock
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.

25.6 DAC interrupts

Table 222. DAC interrupts

Interrupt event	Event flag	Enable control bit
DMA underrun	DMAUDRx	DMAUDRIEx

25.7 DAC registers

Refer to *Section 1 on page 97* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

25.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL23	TSEL22	TSEL21	TSEL20	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL13	TSEL12	TSEL11	TSEL10	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC Channel 2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel 2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel 2 in Normal operating mode

1: DAC channel 2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Note: These bits are available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

These bits are available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

Refer to the trigger selection tables for the details on trigger configuration and mapping.

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).

These bits are available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_pclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_pclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_pclk clock cycle.

These bits are available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled
1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC Channel 1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel 1 calibration, it can be written only if bit EN1=0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel 1 in Normal operating mode

1: DAC channel 1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled

01: Noise wave generation enabled

1x: Triangle wave generation enabled

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
Refer to the trigger selection tables for the details on trigger configuration and mapping.
Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_pclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_pclk clock cycles later to the DAC_DOR1 register
Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_pclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

25.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger
This bit is set by software to trigger the DAC in software trigger mode.
0: No trigger
1: Trigger
Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.
This bit is available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger
This bit is set by software to trigger the DAC in software trigger mode.
0: No trigger
1: Trigger
Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

25.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel1.

25.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

25.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

25.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 25.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

25.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 25.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

25.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 25.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

25.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

25.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

25.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

25.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

25.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 25.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

25.7.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_ FLAG2	DMAU DR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_ FLAG1	DMAU DR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													

Bit 31 **BWST2**: DAC Channel 2 busy writing sample time flag

This bit is systematically set just after Sample and Hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC Channel 2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC Channel 1 busy writing sample time flag

This bit is systematically set just after Sample and Hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC Channel 1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:0 Reserved, must be kept at reset value.

25.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC Channel 2 offset trimming value
These bits are available only on dual-channel DACs. Refer to *Section 25.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC Channel 1 offset trimming value

25.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
													rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC Channel 2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2=0 and bit CEN2 =0 in the DAC_CR register). If EN2=1 or CEN2 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC Channel 2 mode:

– DAC Channel 2 in Normal mode

000: DAC Channel 2 is connected to external pin with Buffer enabled

001: DAC Channel 2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC Channel 2 is connected to external pin with buffer disabled

011: DAC Channel 2 is connected to on chip peripherals with Buffer disabled

– DAC Channel 2 in Sample and Hold mode

100: DAC Channel 2 is connected to external pin with Buffer enabled

101: DAC Channel 2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC Channel 2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC Channel 2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2=0.

Refer to Section 25.3: DAC implementation for the availability of DAC channel 2.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC Channel 1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1=0 and bit CEN1 =0 in the DAC_CR register). If EN1=1 or CEN1 =1 the write operation is ignored.

They can be set and cleared by software to select the DAC Channel 1 mode:

– DAC Channel 1 in Normal mode

000: DAC Channel 1 is connected to external pin with Buffer enabled

001: DAC Channel 1 is connected to external pin and to on chip peripherals with Buffer enabled

010: DAC Channel 1 is connected to external pin with Buffer disabled

011: DAC Channel 1 is connected to on chip peripherals with Buffer disabled

– DAC Channel 1 in sample & hold mode

100: DAC Channel 1 is connected to external pin with Buffer enabled

101: DAC Channel 1 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC Channel 1 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC Channel 1 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN1=0.

25.7.17 DAC channel 1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC Channel 1 sample Time (only valid in Sample and Hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWSTx of DAC_SCR register is low, If BWSTx=1, the write operation is ignored.

*Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI clock period.*

25.7.18 DAC channel 2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 25.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE2[9:0]:** DAC Channel 2 sample Time (only valid in Sample and Hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWSTx of DAC_SR register is low, if BWSTx=1, the write operation is ignored.

*Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI clock period.*

25.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]:** DAC Channel 2 hold time (only valid in Sample and Hold mode).
Hold time= (THOLD[9:0]) x LSI clock period

Note: This register can be modified only when EN2=0.

These bits are available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC Channel 1 hold Time (only valid in Sample and Hold mode)
Hold time= (THOLD[9:0]) x LSI clock period

Note: This register can be modified only when EN2=0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CEN2x=0 in the DAC_CR register). If ENx=1 or CENx=1 the write operation is ignored.

25.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC Channel 2 refresh Time (only valid in Sample and Hold mode)
Refresh time= (TREFRESH[7:0]) x LSI clock period

Note: This register can be modified only when EN2=0.
These bits are available only on dual-channel DACs. Refer to Section 25.3: DAC implementation.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC Channel 1 refresh Time (only valid in Sample and Hold mode)
Refresh time= (TREFRESH[7:0]) x LSI clock period

Note: This register can be modified only when EN2=0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx=0 and bit CEN2x=0 in the DAC_CR register). If ENx=1 or CENx=1 the write operation is ignored.

25.7.21 DAC register map

Table 223 summarizes the DAC registers.

Table 223. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL23	TSEL22	TSEL21	TSEL20	TEN2	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[2:0]		TSEL13	TSEL12	TSEL11	TSEL10	TEN1	EN1
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_DHR12L1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_DHR12L2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x20	DAC_DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_DHR12LD	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x28	DAC_DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0	0	0													

Table 223. DAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38	**DAC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	**DAC_MCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value														0	0	0														0	0	0
0x40	**DAC_ SHSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	**DAC_ SHSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	**DAC_ SHHR**	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	**DAC_ SHRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

26 Voltage reference buffer (VREFBUF)

26.1 Introduction

The STM32H743/753 and STM32H750 devices embed a voltage reference buffer which can be used as voltage reference for ADCs, DACs and also as voltage reference for external components through the VREF+ pin.

26.2 VREFBUF functional description

The internal voltage reference buffer supports four voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: around 2.5 V.
- VRS = 001: around 2.048 V.
- VRS = 010: around 1.8 V.
- VRS = 011: around 1.5 V.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 224. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer OFF: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer OFF – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer ON – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREFBUF buffer OFF – V_{REF+} pin floating. The voltage is held with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

26.3 VREFBUF registers

26.3.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
									rw	rw	rw	r		rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

000: Voltage reference set to 2.5 V

001: Voltage reference set to 2.048 V

010: Voltage reference set to 1.8 V

011: Voltage reference set to 1.5 V

Others: Reserved

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.

0: V_{REF+} pin is internally connected to the voltage reference buffer output.

1: V_{REF+} pin is high impedance.

Refer to *Table 224: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.

0: Internal voltage reference mode disable (external voltage reference mode).

1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

26.3.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]**: Trimming code

These bits are automatically initialized after reset with the trimming value stored in the Flash memory during the production test. Writing into these bits allows to tune the internal reference buffer voltage.

26.3.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 225. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**VREFBUF_CSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	**VREFBUF_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

27 Comparator (COMP)

27.1 Introduction

The device embeds two ultra-low-power comparator channels (COMP1 and COMP2). They can be used for a variety of functions including:

- wake up from low-power mode triggered by an analog signal
- analog signal conditioning
- cycle-by-cycle current control loop when combined with a PWM output from a timer

27.2 COMP main features

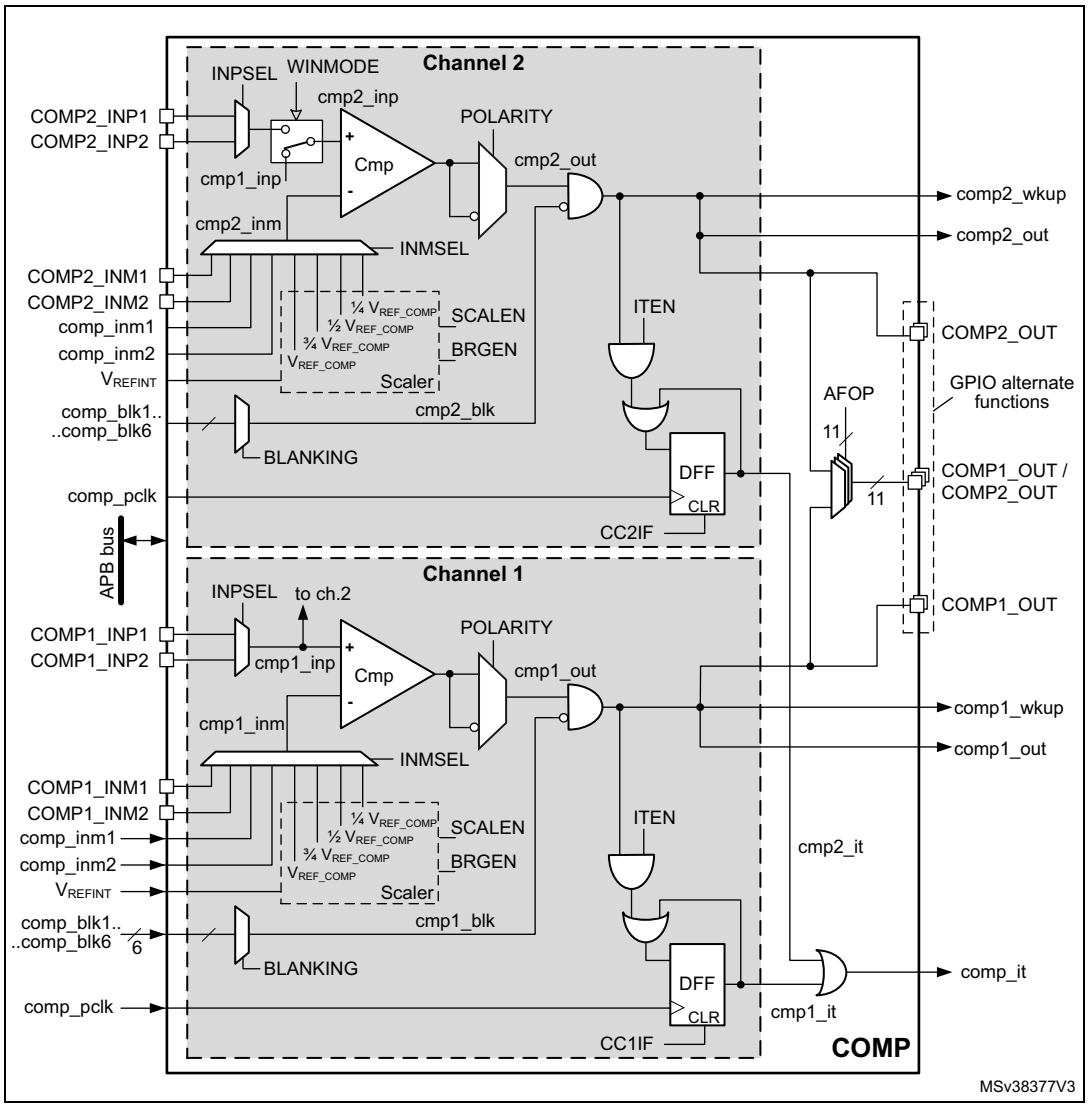
- Selectable inverting analog inputs:
  - I/O pins (different for either channel)
  - DAC Channel1 and Channel2 outputs
  - internal reference voltage and three sub multiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Two I/O pins per channel selectable as non-inverting analog inputs
- Programmable hysteresis
- Programmable speed / consumption
- Mapping of outputs to I/Os
- Redirection of outputs to timer inputs for triggering:
  - capture events
  - OCREF_CLR events (for cycle-by-cycle current control)
  - break events for fast PWM shutdowns
- Blanking of comparator outputs
- Window comparator
- Interrupt generation capability with wake up from Sleep and Stop modes (through the EXTI controller)
- Direct interrupt output to the CPU

27.3 COMP functional description

27.3.1 COMP block diagram

The block diagram of the comparators is shown in *Figure 210: Comparator functional block diagram*.

Figure 210. Comparator functional block diagram



27.3.2 COMP pins and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

The comparator outputs can be connected to the I/Os through their alternate functions. Refer to the product datasheet.

The outputs can also be internally redirected to a variety of timer inputs for the following purposes:

- emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- cycle-by-cycle current control, using ETR inputs of timers
- input capture for timing measurements

The comparator output can be routed simultaneously internally and to the I/O pins.

Table 226. COMP input/output internal signals

Signal name	Signal type	Description
comp_inm1	Analog input	Inverting input source for both COMP channels: DAC ch.1
comp_inm2	Analog input	Inverting input source for both COMP channels: DAC ch.2
comp_blk1	Digital input	Blanking input source for both COMP channels: TIM1 OC5
comp_blk2	Digital input	Blanking input source for both COMP channels: TIM2 OC3
comp_blk3	Digital input	Blanking input source for both COMP channels: TIM3 OC3
comp_blk4	Digital input	Blanking input source for both COMP channels: TIM3 OC4
comp_blk5	Digital input	Blanking input source for both COMP channels: TIM8 OC5
comp_blk6	Digital input	Blanking input source for both COMP channels: TIM15 OC1
comp_pclk	Digital input	APB clock for both COMP channels
comp1_wkup	Digital output	COMP channel 1 wakeup out
comp1_out	Digital output	COMP channel 1 out
comp2_wkup	Digital output	COMP channel 2 wakeup out
comp2_out	Digital output	COMP channel 2 out
comp_it	Digital output	COMP interrupt out

Table 227. COMP input/output pins

Signal name	Signal type	Description
COMP1_INM1	Analog input	COMP channel 1 inverting input source 1 (PB1)
COMP1_INM2	Analog input	COMP channel 1 inverting input source 2 (PC4)
COMP1_INP1	Analog input	COMP channel 1 non-inverting input source 1 (PB0)
COMP1_INP2	Analog input	COMP channel 1 non-inverting input source 2 (PB2)
COMP2_INM1	Analog input	COMP channel 2 inverting input source 1 (PE10)
COMP2_INM2	Analog input	COMP channel 2 inverting input source 2 (PE7)
COMP2_INP1	Analog input	COMP channel 2 non-inverting input source 1 (PE9)
COMP2_INP2	Analog input	COMP channel 2 non-inverting input source 2 (PE11)
COMP1_OUT	Digital output	COMP channel 1 output: see *Section 27.3.8: Comparator output on GPIOs*.
COMP2_OUT	Digital output	COMP channel 2 output: see *Section 27.3.8: Comparator output on GPIOs*.

27.3.3 COMP reset and clocks

The clock comp_pclk provided by the clock controller is synchronous with the APB clock.

Note: ***Important:*** *The polarity selection logic and the output redirection to the port works independently from the APB clock. This allows the comparator to work even in Stop mode. The interrupt line, connected to the NVIC of CPU, requires the APB clock (comp_pclk) to work. In absence of the APB clock, the interrupt signal comp_it cannot be generated.*

27.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications with specific functional safety requirements, the comparator configuration can be protected against undesired alteration that could happen, for example, at program counter corruption.

For this purpose, the comparator configuration registers can be write-protected (read-only).

Upon configuring a comparator channel, its LOCK bit is set to 1. This causes the whole register set of the comparator channel, as well as the common COMP_OR register, to become read-only, the LOCK bit inclusive.

The write protection can only be removed through the MCU reset.

The COMP_OR register is locked by the LOCK bit of COMP_CFGR1 OR COMP_CFGR2.

27.3.5 Window comparator

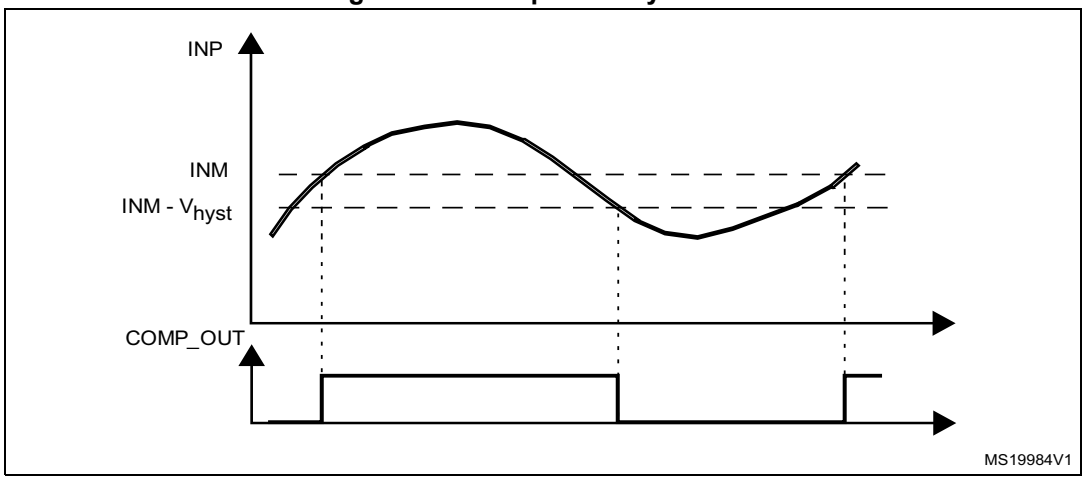
The purpose of the window comparator is to monitor the analog voltage and check that it is comprised within the specified voltage range defined by lower and upper thresholds.

The window comparator requires both COMP channels. The monitored analog voltage is connected to their non-inverting (plus) inputs and the upper and lower threshold voltages are connected to the inverting (minus) input of either comparator, respectively. The non-inverting input of the COMP channel 2 can be connected internally with the non-inverting input of the COMP channel 1 by enabling WINMODE bit. This can save the input pins of COMP channel 2 for other purposes. See *Figure 210: Comparator functional block diagram*.

27.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

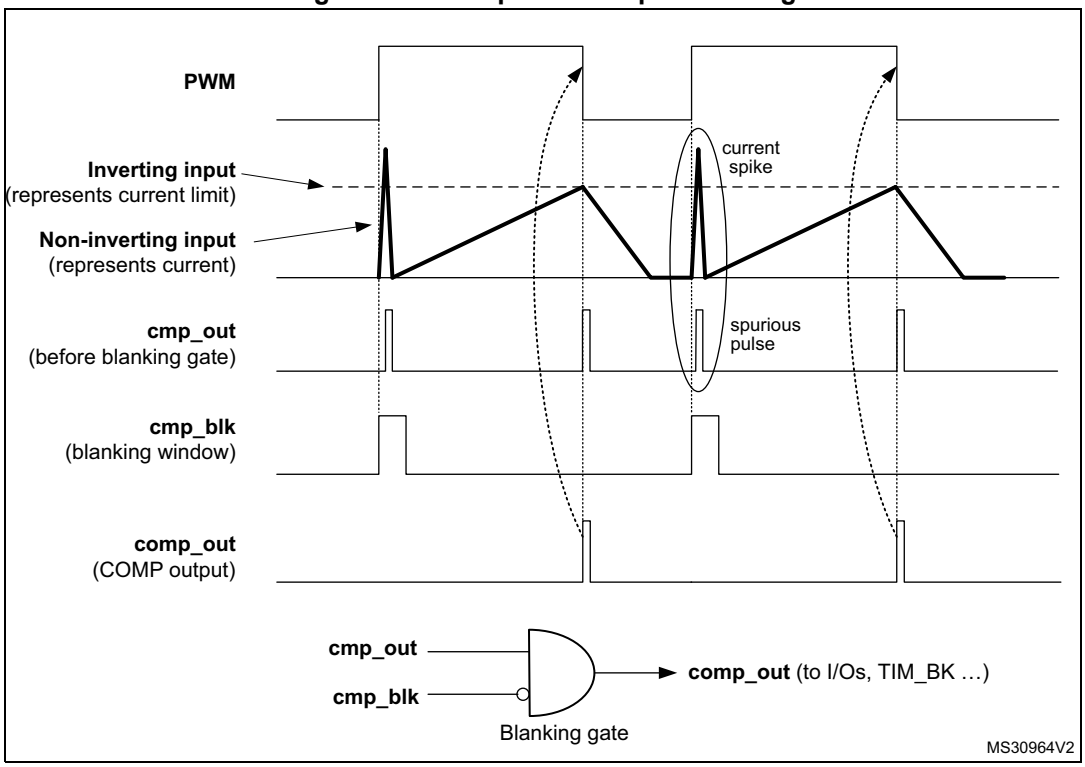
Figure 211. Comparator hysteresis



27.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It uses a blanking window defined with a timer output compare signal. Refer to the register description for selectable blanking signals. The blanking signal gates the internal comparator output such as to clean the comp_out from spurious pulses due to current spikes, as depicted in *Figure 212* (the COMP channel number is not represented).

Figure 212. Comparator output blanking



27.3.8 Comparator output on GPIOs

The COMP1_OUT and COMP2_OUT outputs of the comparator channels are mapped to GPIOs through the AFOP field of the COMP_OR register, bits [10:0], and through the GPIO alternate function.

Table 228. COMP1_OUT assignment to GPIOs

COMP1_OUT	Alternate Function
PC5	AF13
PE12	AF13
PA6	AF10, AF12 (can be used as timer break in)
PA8	AF12 (can be used as timer break in)
PB12	AF13 (can be used as timer break in)
PE6	AF11 (can be used as timer break in)
PE15	AF13 (can be used as timer break in)
PG2	AF11 (can be used as timer break in)
PG3	AF11 (can be used as timer break in)
PG4	AF11 (can be used as timer break in)
PI1	AF11 (can be used as timer break in)
PI4	AF11 (can be used as timer break in)
PK2	AF10, AF11 (can be used as timer break in)

Table 229. COMP2_OUT assignment to GPIOs

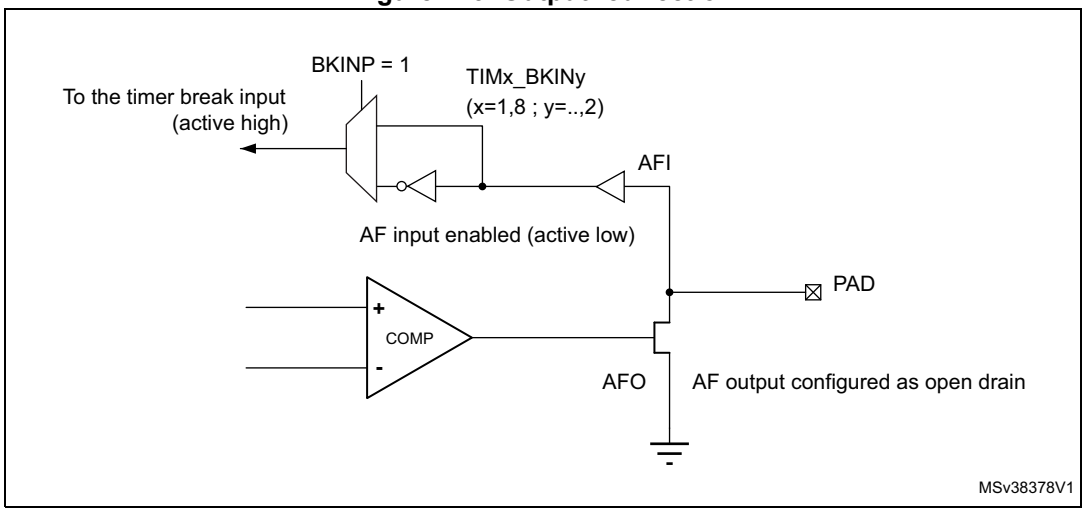
COMP2_OUT	Alternate Function
PE8	AF13
PE13	AF13
PA6	AF10, AF12 (can be used as timer break in)
PA8	AF12 (can be used as timer break in)
PB12	AF13 (can be used as timer break in)
PE6	AF11 (can be used as timer break in)
PE15	AF13 (can be used as timer break in)
PG2	AF11 (can be used as timer break in)
PG3	AF11 (can be used as timer break in)
PG4	AF11 (can be used as timer break in)
PI1	AF11 (can be used as timer break in)
PI4	AF11 (can be used as timer break in)
PK2	AF10, AF11 (can be used as timer break in)

The assignment to GPIOs for both comparator channel outputs must be done before locking registers of any channel, because the common COMP_OR register is locked when locking the registers of either comparator channel.

27.3.9 Comparator output redirection

The outputs of either COMP channel can be redirected to timer break inputs (TIMx_BKIN or TIMx_BKIN2), as shown in *Figure 213*. For that end, the COMP channel output is connected to one of GPIOs programmable in alternate function as timer break input. See *Table 228* and *Table 229*. The selected GPIO(s) must be set in open drain mode. The COMP output passes through the GPIO to the timer break input. With a pull-up resistor, the selected GPIO can be used as timer break input logic OR-ed with the comparator output.

Figure 213. Output redirection



27.3.10 COMP power and speed modes

The power consumption of the COMP channels versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bits PWRMODE[1:0] in COMP_CFGRx registers can be programmed as follows:

00: High speed / full power

01: Medium speed / medium power

10: Medium speed / medium power

11: Very-low speed / ultra-low-power

27.4 COMP low-power modes

Table 230. Comparator behavior in the low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.

Note: The comparators cannot be used to exit the device from Sleep or Stop mode when the internal reference voltage is switched off.

27.5 COMP interrupts

There are two ways to use the comparator as interrupt source.

The comparator outputs are internally connected to the Extended interrupt and event controller. Each comparator has its own EXTI line and can generate either interrupts or events to make the device exit low-power modes.

The comparators also provide an interrupt line to the NVIC of CPU. This functionality is used when the CPU is active to handle low latency interrupt. It requires APB clock running.

27.5.1 Interrupt through EXTI block

Refer to Interrupt and events section for more details.

Sequence to enable the COMPx interrupt through EXTI block:

1. Configure the EXTI line, receiving the comp_wkup signal, in interrupt mode, select the rising, falling or either-edge sensitivity and enable the EXTI line.
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.
3. Enable the COMPx.

Table 231. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop modes	Exit from Standby mode
comp1_wkup	through EXTI	through EXTI	yes	yes	N/A
comp2_whup	through EXTI	through EXTI	yes	yes	N/A

27.5.2 Interrupt through NVIC of the CPU

Sequence to enable the COMPx interrupt through NVIC of the CPU:

1. Configure and enable the NVIC IRQ channel mapped to the comp_it line.
2. Configure and enable the ITEN in COMP_CFGRx.
3. Enable the COMPx.

Table 232. Interrupt control bits

Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit from Sleep mode	Exit from Stop modes
comp_it	C1IF in	ITEN in COMP_CFGR1	CC1IF	yes (With APB clock)	no
comp_it	C2IF in	ITEN in COMP_CFGR2	CC2IF	yes (With APB clock)	no

Note: It is mandatory to enable APB clock to use this interrupt. If clock is not enabled, interrupt is not generated.

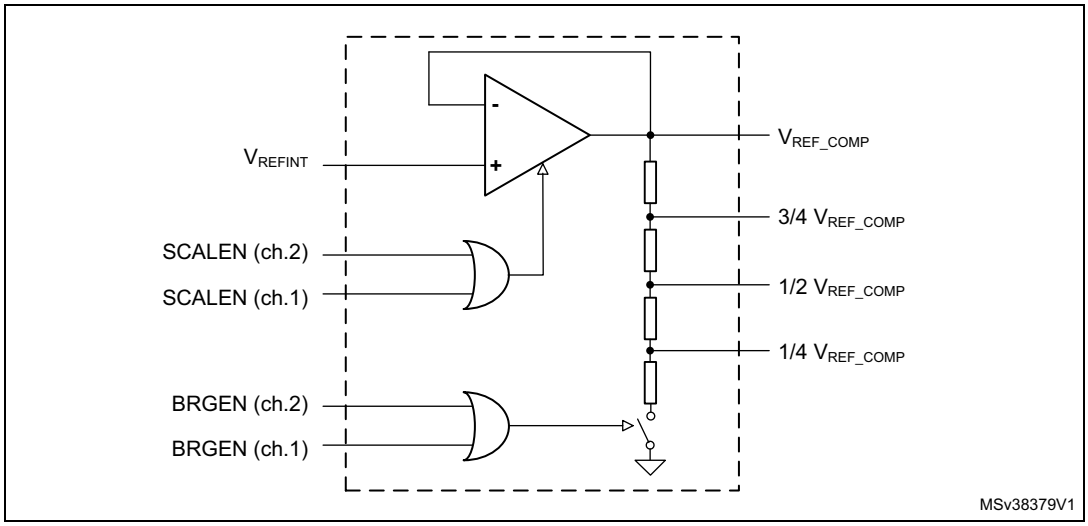
27.6 SCALER function

The scaler block is available to provide the different voltage reference levels to the comparator inputs. It is based on an amplifier driving a resistor bridge. The amplifier input is connected to the internal voltage reference.

The amplifier and the resistor bridge can be enabled separately. The amplifier is enabled by the SCALEN bits of the COMP_CFGRx registers. The resistor bridge is enabled by the BRGEN bits of the COMP_CFGRx registers.

When the resistor divided voltage is not used, the resistor bridge can be disconnected in order to reduce the consumption. When it is disconnected, the 1/4 V_{REF_COMP}, 1/2 V_{REF_COMP} and 3/4 V_{REF_COMP} levels are equal to V_{REF_COMP}.

Figure 214. Scaler block diagram



27.7 COMP registers

27.7.1 Comparator status register (COMP_SR)

The COMP_SR is the comparator status register.

Address offset: 0x00

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2IF	C1IF
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2VAL	C1VAL
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **C2IF**: COMP channel 2 Interrupt Flag
This bit is set by hardware when the COMP channel 2 output is set
This bit is cleared by software writing 1 the CC2IF bit in the COMP_ICFR register.

Bit 16 **C1IF**: COMP channel 1 Interrupt Flag
This bit is set by hardware when the COMP channel 1 output is set
This bit is cleared by software writing 1 the CC1IF bit in the COMP_ICFR register.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **C2VAL:** COMP channel 2 output status bit
This bit is read-only. It reflects the current COMP channel 2 output taking into account POLARITY and BLANKING bits effect.

Bit 0 **C1VAL:** COMP channel 1 output status bit
This bit is read-only. It reflects the current COMP channel 1 output taking into account POLARITY and BLANKING bits effect.

27.7.2 Comparator interrupt clear flag register (COMP_ICFR)

The COMP_ICFR is the Comparator interrupt clear flag register.

Address offset: 0x00

System reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	CC1IF
														w1o	w1o
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC2IF**: Clear COMP channel 2 Interrupt Flag
Writing 1 clears the C2IF flag in the COMP_SR register.

Bit 16 **CC1IF**: Clear COMP channel 1 Interrupt Flag
Writing 1 clears the C1IF flag in the COMP_SR register.

Bits 15:0 Reserved, must be kept at reset value.

27.7.3 Comparator option register (COMP_OR)

The COMP_OR is the Comparator option register.

Address offset: 0x08

System reset value: 0x0000 0000

When OR_CFG=0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OR15	OR14	OR13	OR12	OR11	AFOP										
rw	rw	rw	rw	rw	rw										

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **OR**: Option Register (When OR_CFG=0)

Bits 10:0 **AFOP[10:0]:** Selection of source for alternate function of output ports
Bits of this field are set end cleared by software (only if LOCK not set).
Output port (GPIO) correspondence:

bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PK2	PI4	PI1	PG4	PG3	PG2	PE15	PE6	PB12	PA8	PA6

For each bit:
0: COMP1_OUT is selected for the alternate function of the corresponding GPIO
1: COMP2_OUT is selected for the alternate function of the corresponding GPIO

When OR_CFG=1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OR31	OR30	OR29	OR28	OR27	OR26	OR25	OR24	OR23	OR22	OR21	OR20	OR19	OR18	OR17	OR16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OR15	OR14	OR13	OR12	OR11	AFOP										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 **OR**: Option Register (When OR_CFG=1)

Bits 10:0 **AFOP[10:0]:** Selection of source for alternate function of output ports
Bits of this field are set end cleared by software (only if LOCK not set).
Output port (GPIO) correspondence:

bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PK2	PI4	PI1	PG4	PG3	PG2	PE15	PE6	PB12	PA8	PA6

For each bit:
0: COMP1_OUT is selected for the alternate function of the corresponding GPIO
1: COMP2_OUT is selected for the alternate function of the corresponding GPIO

27.7.4 Comparator configuration register 1 (COMP_CFGR1)

The COMP_CFGR1 is the COMP channel 1 configuration register.

Address offset: 0x0C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	BLANKING[3:0]				Res.	Res.	Res.	INPSEL	Res.	INMSEL[2:0]		
rw				rw							rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PWRMODE[1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	Res.	POLARITY	SCALEN	BRGEN	EN
		rw				rw			rw			rw	rw	rw	rw

Bit 31 **LOCK:** Lock bit
This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP channel 1 configuration register COMP_CFGR1[31:0], and COMP_OR register
0: COMP_CFGR1[31:0] register is read/write
1: COMP_CFGR1[31:0] and COMP_OR registers are read-only

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **BLANKING[3:0]:** COMP channel 1 blanking source selection bits
Bits of this field are set and cleared by software (only if LOCK not set).
The field selects the input source for COMP channel 1 output blanking:
0000: No blanking
0001: comp_blk1
0010: comp_blk2
0011: comp_blk3
0100: comp_blk4
0101: comp_blk5
0110: comp_blk6
All other values: reserved

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **INPSEL**: COMP channel 1 non-inverting input selection bit
This bit is set and cleared by software (only if LOCK not set).
0: COMP1_INP1 (PB0)
1: COMP1_INP2 (PB2)

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **INMSEL[2:0]**: COMP channel 1 inverting input selection field

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of COMP channel 1.

000 = 1/4 V_{REF_COMP}
001 = 1/2 V_{REF_COMP}
010 = 3/4 V_{REF_COMP}
011 = V_{REF_COMP}
100 = comp_inm1 (DAC channel 1 output)
101 = comp_inm2 (DAC channel 2 output)
110 = COMP1_INM1 (PB1)
111 = COMP1_INM2 (PC4)

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **PWRMODE[1:0]:** Power Mode of the COMP channel 1

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the COMP channel 1.

00: High speed / full power
01: Medium speed / medium power
10: Medium speed / medium power
11: Ultra low power / ultra-low-power

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **HYST[1:0]:** COMP channel 1 hysteresis selection bits

These bits are set and cleared by software (only if LOCK not set). They select the Hysteresis voltage of the COMP channel 1.

00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ITEN:** COMP channel 1 interrupt enable

This bit is set and cleared by software (only if LOCK not set). This bit enable the interrupt generation of the COMP channel 1.

0: Interrupt generation disabled for COMP channel 1
1: Interrupt generation enabled for COMP channel 1

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **POLARITY:** COMP channel 1 polarity selection bit

This bit is set and cleared by software (only if LOCK not set). It inverts COMP channel 1 polarity.

0: COMP channel 1 output is not inverted
1: COMP channel 1output is inverted

Bit 2 **SCALEN**: Voltage scaler enable bit

This bit is set and cleared by software (only if LOCK not set). This bit enables the V_{REFINT} scaler for the COMP channels.

0: V_{REFINT} scaler disabled (if SCALEN bit of COMP_CFGR2 register is also low)

1: V_{REFINT} scaler enabled

Bit 1 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enables the bridge of the scaler.

0: Scaler resistor bridge disabled (if BRGEN bit of COMP_CFGR2 register is also low)

1: Scaler resistor bridge enabled

If SCALEN is set and BRGEN is reset, all four scaler outputs provide the same level V_{REF_COMP} (similar to V_{REFINT}).

If SCALEN and BRGEN are set, the four scaler outputs provide V_{REF_COMP}, 3/4 V_{REF_COMP}, 1/2 V_{REF_COMP} and 1/4 V_{REF_COMP} levels, respectively.

Bit 0 **EN:** COMP channel 1 enable bit

This bit is set and cleared by software (only if LOCK not set). It enables the COMP channel 1.

0: Disable

1: Enable

27.7.5 Comparator configuration register 2 (COMP_CFGR2)

The COMP_CFGR2 is the COMP channel 2 configuration register.

Address offset: 0x10

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	BLANKING[3:0]				Res.	Res.	Res.	INPSEL	Res.	INMSEL[2:0]		
rw				rw							rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PWRMODE[1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	WINMODE	POLARITY	SCALEN	BRGEN	EN
		rw				rw			rw		rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP channel 2 configuration register COMP_CFGR2[31:0], and COMP_OR register

0: COMP_CFGR2[31:0] register is read/write

1: COMP_CFGR2[31:0] and COMP_OR registers are read-only

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **BLANKING[3:0]:** COMP channel 2 blanking source selection bits

These bits are set and cleared by software (only if LOCK not set). These bits select which timer output controls the COMP channel 2 output blanking.

0000: No blanking

0001: TIM1 OC5 selected as blanking source

0010: TIM2 OC3 selected as blanking source

0011: TIM3 OC3 selected as blanking source

0100: TIM3 OC4 selected as blanking source

0101: TIM8 OC5 selected as blanking source

0110: TIM15 OC1 selected as blanking source

All other values: reserved

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **INPSEL**: COMP channel 2 non-inverting input selection bit

This bit is set and cleared by software (only if LOCK not set).

0: COMP2_INP1 (PE9)

1: COMP2_INP2 (PE11)

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **INMSEL[2:0]**: COMP channel 2 inverting input selection field

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of COMP channel 2.

000 = 1/4 V_{REF_COMP}

001 = 1/2 V_{REF_COMP}

010 = 3/4 V_{REF_COMP}

011 = V_{REF_COMP}

100 = comp_inm1 (DAC channel 1 output)

101 = comp_inm2 (DAC channel 2 output)

110 = COMP2_INM1 (PE10)

111 = COMP2_INM2 (PE7)

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **PWRMODE[1:0]:** Power Mode of the COMP channel 2

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the COMP channel 2.

00: High speed / full power

01: Medium speed / medium power

10: Medium speed / medium power

11: Ultra low power / ultra-low-power

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **HYST[1:0]:** COMP channel 2 hysteresis selection bits
These bits are set and cleared by software (only if LOCK not set). They select the Hysteresis voltage of the COMP channel 2.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ITEN:** COMP channel 2 interrupt enable
This bit is set and cleared by software (only if LOCK not set). This bit enable the interrupt generation of the COMP channel 2.
0: Interrupt generation disabled for COMP channel 2
1: Interrupt generation enabled for COMP channel 2

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WINMODE**: Window comparator mode selection bit
This bit is set and cleared by software (only if LOCK not set). This bit selects the window mode of the comparators. If set, the non-inverting input of COMP channel 2 is connected to the non-inverting input of the COMP channel 1.
Depending on the bit value, the non-inverting input of COMP channel 2 is connected to:
0: COMP2_INP input selector
1: Non-inverting input comp1_inp of COMP channel 1

Bit 3 **POLARITY:** COMP channel 2 polarity selection bit
This bit is set and cleared by software (only if LOCK not set). It inverts COMP channel 2 polarity.
0: COMP channel 2 output is not inverted
1: COMP channel 2 output is inverted

Bit 2 **SCALEN**: Voltage scaler enable bit
This bit is set and cleared by software (only if LOCK not set). This bit enables the V_{REFINT} scaler for the COMP channels.
0: V_{REFINT} scaler disabled (if SCALEN bit of COMP_CFGR1 register is also low)
1: V_{REFINT} scaler enabled

Bit 1 **BRGEN:** Scaler bridge enable
This bit is set and cleared by software (only if LOCK not set). This bit enables the bridge of the scaler.
0: Scaler resistor bridge disabled (if BRGEN bit of COMP_CFGR1 register is also low)
1: Scaler resistor bridge enabled
If SCALEN is set and BRGEN is reset, all four scaler outputs provide the same level V_{REF_COMP} (similar to V_{REFINT}).
If SCALEN and BRGEN are set, the four scaler outputs provide V_{REF_COMP}, 3/4 V_{REF_COMP}, 1/2 V_{REF_COMP} and 1/4 V_{REF_COMP} levels, respectively.

Bit 0 **EN:** COMP channel 2 enable bit
This bit is set and cleared by software (only if LOCK not set). It enables the COMP channel 2.
0: Disable
1: Enable

27.7.6 COMP register map

The following table summarizes the comparator registers.

Table 233. COMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**COMP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2IF	C1IF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2VAL	C1VAL
	Reset value															0	0															0	0
0x04	**COMP_ICFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	CC1IF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0																
0x08	**COMP_OR (OR_CFG=0)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR15	OR14	OR13	OR12	OR11	AFOP										
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**COMP_OR (OR_CFG=1)**	OR31	OR30	OR29	OR28	OR27	OR26	OR25	OR24	OR23	OR22	OR21	OR20	OR19	OR18	OR17	OR16	OR15	OR14	OR13	OR12	OR11	AFOP										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**COMP_CFG R1**	LOCK	Res.	Res.	Res.	BLANKING				Res.	Res.	Res.	INPSEL	Res.	INMSEL			Res.	Res.	PWRMODE		Res.	Res.	HYST		Res.	ITEN	Res.	Res.	POLARITY	SCALEN	BRGEN	EN
	Reset value	0				0	0	0	0				0		0	0	0			0	0			0	0		0			0	0	0	0
0x10	**COMP_CFG R2**	LOCK	Res.	Res.	Res.	BLANKING				Res.	Res.	Res.	INPSEL	Res.	INMSEL			Res.	Res.	PWRMODE		Res.	Res.	HYST		Res.	ITEN	Res.	WINMODE	POLARITY	SCALEN	BRGEN	EN
	Reset value	0				0	0	0	0				0		0	0	0			0	0			0	0		0		0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

28 Operational amplifiers (OPAMP)

28.1 Introduction

The devices embed two operational amplifiers with two inputs and one output each. The three I/Os can be connected to the external pins, thus enabling any type of external interconnections. The operational amplifiers can be configured internally as a follower, as an amplifier with a non-inverting gain ranging from 2 to 16 or with inverting gain ranging from -1 to -15.

Refer to *Section 28.3.3: Signal routing* for detailed information on OPAMP input and output connection to internal peripherals.

28.2 OPAMP main features

- Rail-to-rail input and output voltage range
- Low input bias current (down to 1 nA)
- Low input offset voltage (1.5 mV after calibration, 10 mV with factory calibration)
- 7 MHz gain bandwidth
- High-speed mode to achieve a better slew rate

Note: Refer to the product datasheet for detailed OPAMP characteristics.

28.3 OPAMP functional description

The OPAMP has several modes.

Each OPAMP can be individually enabled, when disabled the output is high-impedance.

When enabled, it can be in calibration mode, all input and output of the OPAMP are then disconnected, or in functional mode.

There are two functional modes, the high-speed mode and the normal mode. In functional mode the inputs and output of the OPAMP are connected as described in *Section 28.3.3: Signal routing*.

28.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. When the application does not need to have read or write access to those registers, the clock can be switched off using the peripheral clock enable register (see OPAMPEN bit in *Section 7.7.43: RCC APB1 Clock Register (RCC_APB1LENR)*).

The bit OPAEN enables and disables the OPAMP operation. The OPAMP registers configurations should be changed before enabling the OPAEN bit in order to avoid spurious effects on the output.

When the output of the operational amplifier is no more needed the operational amplifier can be disabled to save power. All the configurations previously set (including the calibration) are maintained while OPAMP is disabled.

28.3.2 Initial configuration

The default configuration of the operational amplifier is a functional mode where the three input/outputs are connected to external pins. In the default mode the operational amplifier uses the factory trimming values for its offset calibration. See electrical characteristics section of the datasheet for factory trimming conditions, usually the temperature is 30 °C and the voltage is 3 V. The trimming values can be adjusted, see *Section 28.3.5: Calibration* for changing the trimming values. The default configuration uses the normal mode, which provides the standard performance. The bit OPAHSM can be set in order to switch the operational amplifier to high-speed mode for a better slew rate. Both normal and high-speed mode characteristics are defined in *Section: Electrical characteristics* of the datasheet.

As soon as the OPAEN bit in OPAMPx_CSR register is set, the operational amplifier is functional. The two input pins and the output pin are connected as defined in *Section 28.3.3: Signal routing* and the default connection settings can be changed.

Note: The inputs and output pins must be configured in analog mode (default state) in the corresponding GPIOx_MODER register.

28.3.3 Signal routing

The routing for the operational amplifier pins is determined by OPAMPx_CSR register.

The connections of the two operational amplifiers (OPAMP1 and OPAMP2) are described in the table below.

Table 234. Operational amplifier possible connections

Signal	Pin	Internal	comment
OPAMP1_VINM	PC5(INM0) PA7(INM1)	ADC1_IN8 ADC2_IN8 OPAMP1_VOUT or PGA	controlled by bits PGA_GAIN and VM_SEL.
OPAMP1_VINP	PB0	dac_out1 ADC1_IN9 ADC2_IN9 COMP1_INP	controlled by bit VP_SEL.
OPAMP1_VOUT(1)	PC4	ADC1_IN4 ADC2_IN4 COMP1_INM7	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.
OPAMP2_VINM	PE8(INM0) PG1(INM1)	OPAMP2_VOUT or PGA	controlled by bits PGA_GAIN and VM_SEL.
OPAMP2_VINP	PE9	dac_out2 COMP2_INP	controlled by bit VP_SEL
OPAMP2_VOUT(1)	PE7	COMP2_INM7	-

1. Both OPAMP1_VOUT and OPAMP2_VOUT are not available on all packages. In this case, the unused I/O should not be activated since is internally set to a fixed configuration.

28.3.4 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifiers can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

Note: The amplifier output pin is directly connected to the output pad to minimize the output impedance. When the amplifier is enabled, it cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the internal channel.

The impedance of the signal must be maintained below a level which avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.

Standalone mode (external gain setting mode)

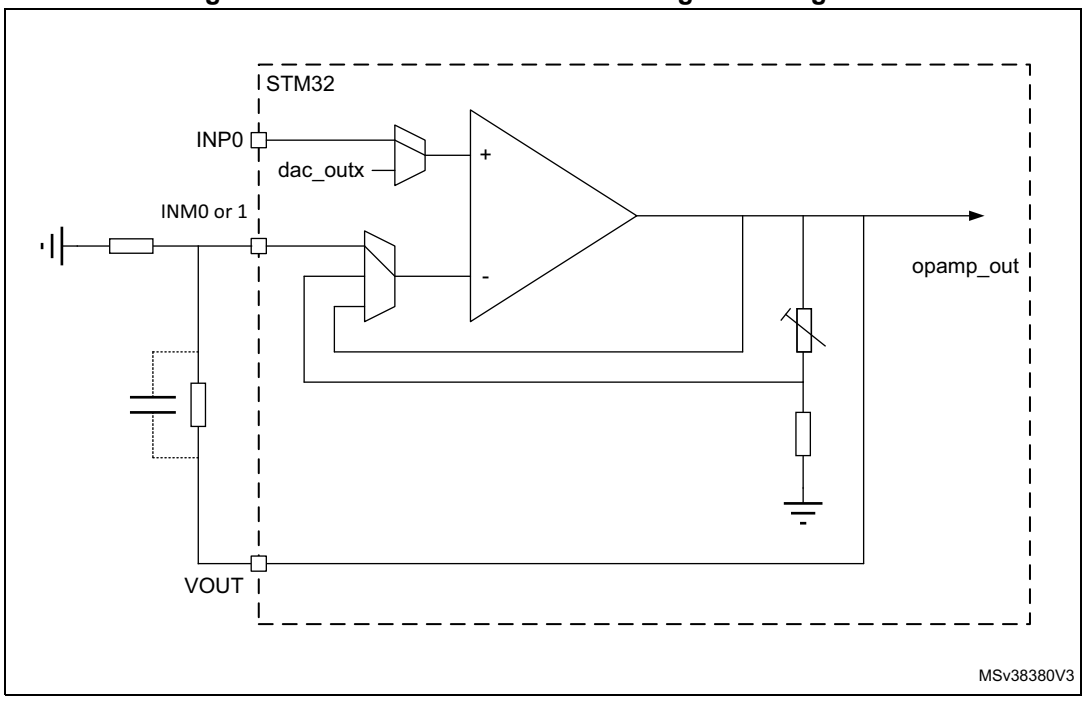
The procedure to use the OPAMP in standalone mode is presented hereafter.

Starting from the default value of OPAMPx_CSR, and the default state of GPIOx_MODER, as soon as the OPAEN bit is set, the two input pins and the output pin are connected to the operational amplifier.

This default configuration uses the factory trimming values and operates in normal mode (highest performance). The behavior of the OPAMP can be changed as follows:

- OPAHSM can be set to “operational amplifier high-speed” mode in order to have high slew rate.
- USERTRIM can be set to modify the trimming values for input offsets.

Figure 215. Standalone mode: external gain setting mode



Follower configuration mode

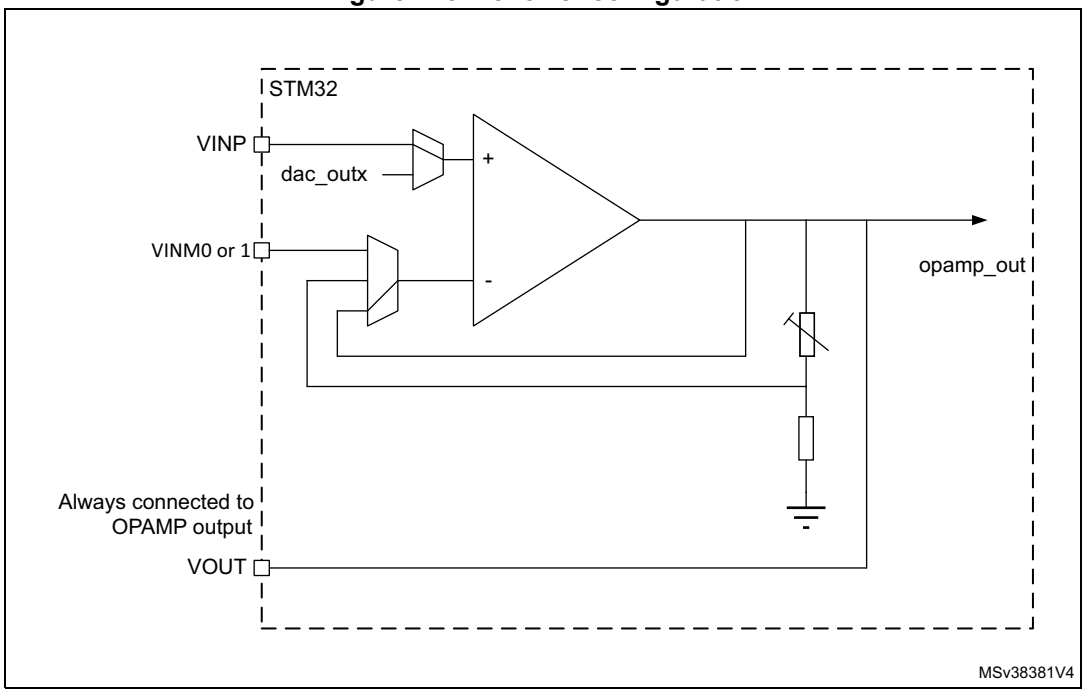
The procedure to use the OPAMP in follower mode is presented hereafter.

- configure VM_SEL bits as “opamp_out connected to OPAMPx_VINM input”, 11
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”, 00
- As soon as the OPAEN bit is set, the voltage on pin OPAMPx_VINP is buffered to pin OPAMPx_VOUT.

Note: The pin corresponding to OPAMPx_VINM is free for another usage.

The signal on the OPAMP1 output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming the input signal frequency is compatible with the operational amplifier gain bandwidth specification.

Figure 216. Follower configuration



Programmable gain amplifier mode

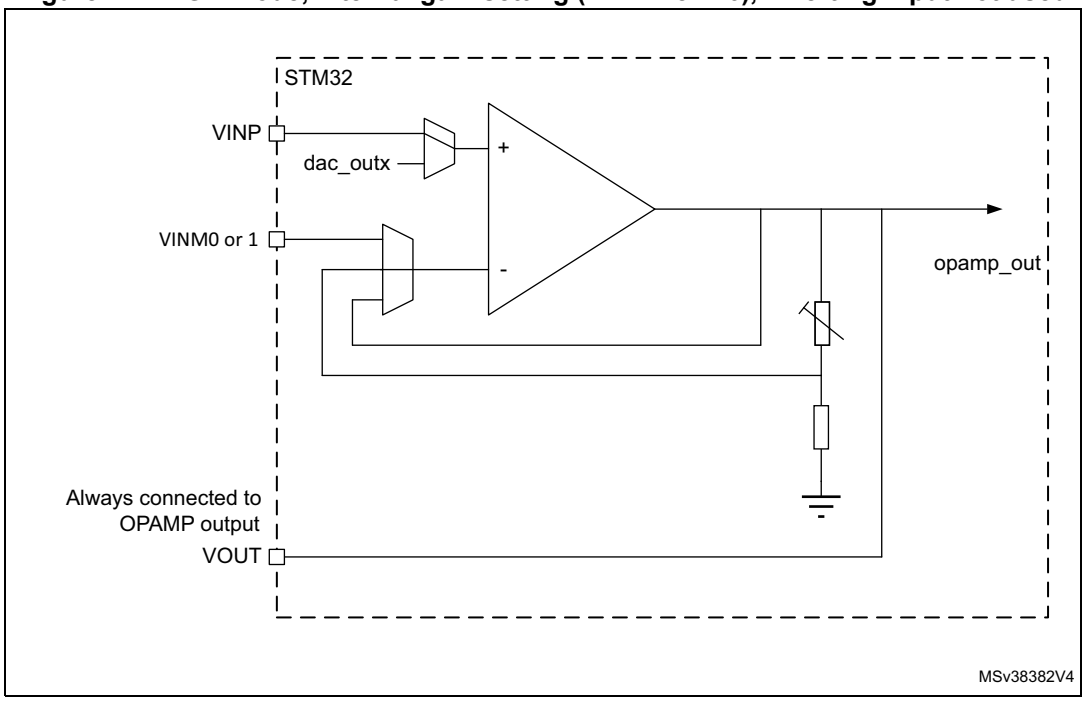
The procedure to use the OPAMP as programmable gain amplifier is presented hereafter.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “internal Gain 2, 4, 8 or 16”, 0000 to 0011
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”, 00

  As soon as the OPAEN bit is set, the voltage on pin OPAMPx_VINP is amplified by the selected gain and visible on pin OPAMPx_VOUT.

Note: To avoid saturation, the input voltage should stay below V_{DDA} divided by the selected gain.

Figure 217. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used



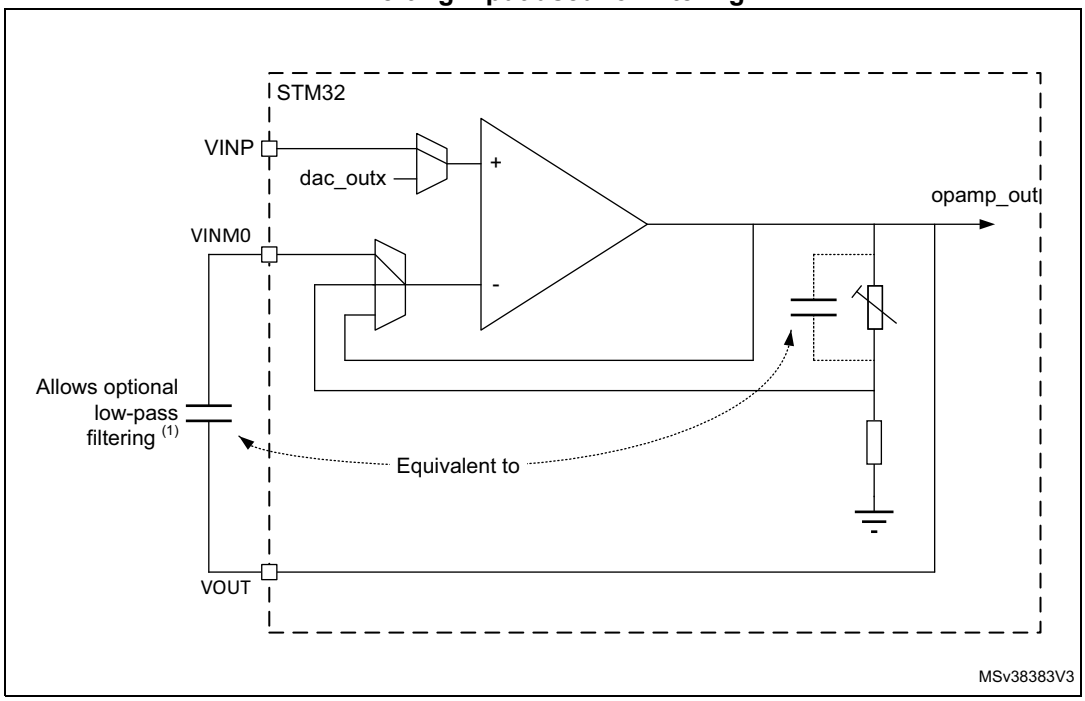
Programmable gain amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external filtering, is presented hereafter.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “internal Gain 2, 4, 8 or 16 with filtering on INM0”, 0100 to 0111
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

  Any external connection on INM can be used in parallel with the internal PGA, for example a capacitor can be connected between opamp_out and INM for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

Figure 218. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



1. The gain depends on the cut-off frequency.

Programmable gain amplifier, non-inverting with external bias or inverting mode

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “Inverting gain=-1,-3,-7,-15/ Non-inverting gain =2,4,8,16 with INM0”, 1000 to 1011
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

Figure 219. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15)

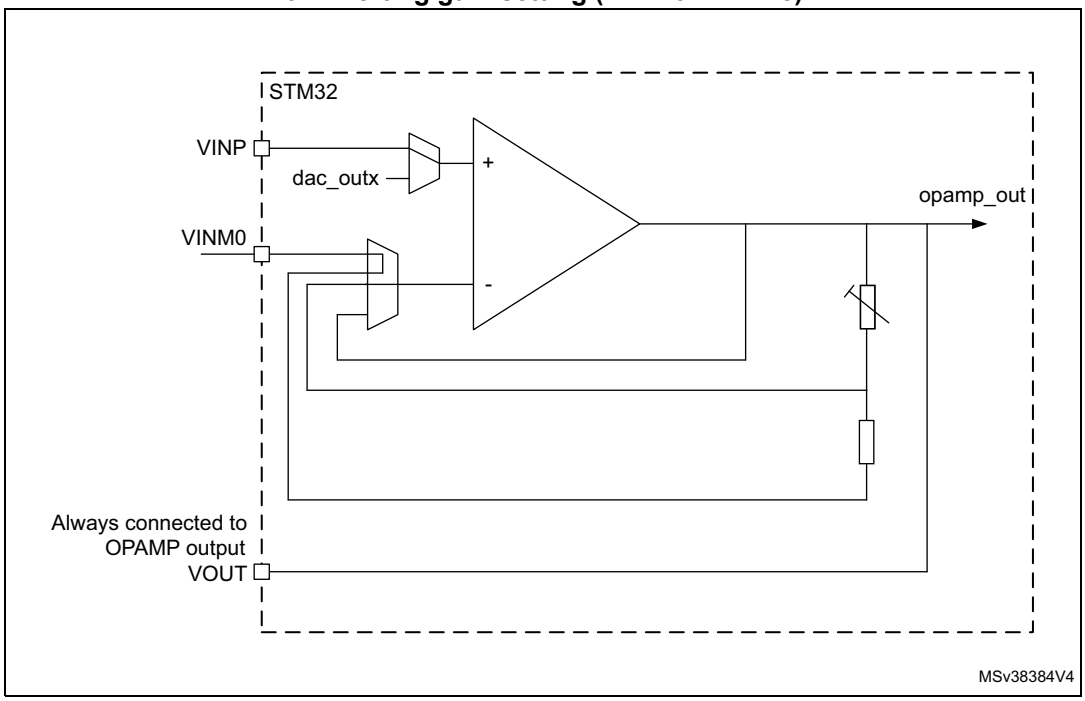
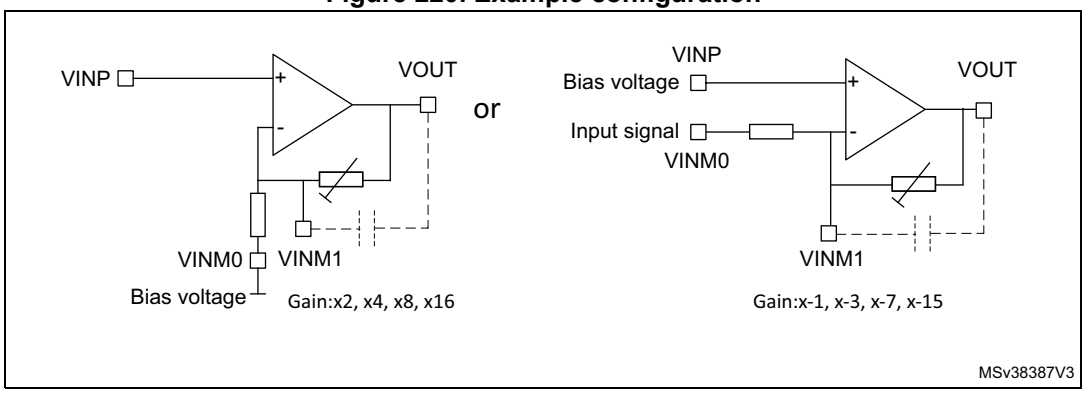


Figure 220. Example configuration



Programmable gain amplifier, non-inverting with external bias or inverting mode with filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode with filtering

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “Inverting gain=-1,-3,-7,-15/ Non-inverting gain =2,4,8,16 with INM0, INM1 node for filtering”, 1100 to 1111
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

  Any external connection on VM1 can be used in parallel with the internal PGA, for example a capacitor can be connected between opamp_out and VM1 for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

Figure 221. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15) with filtering

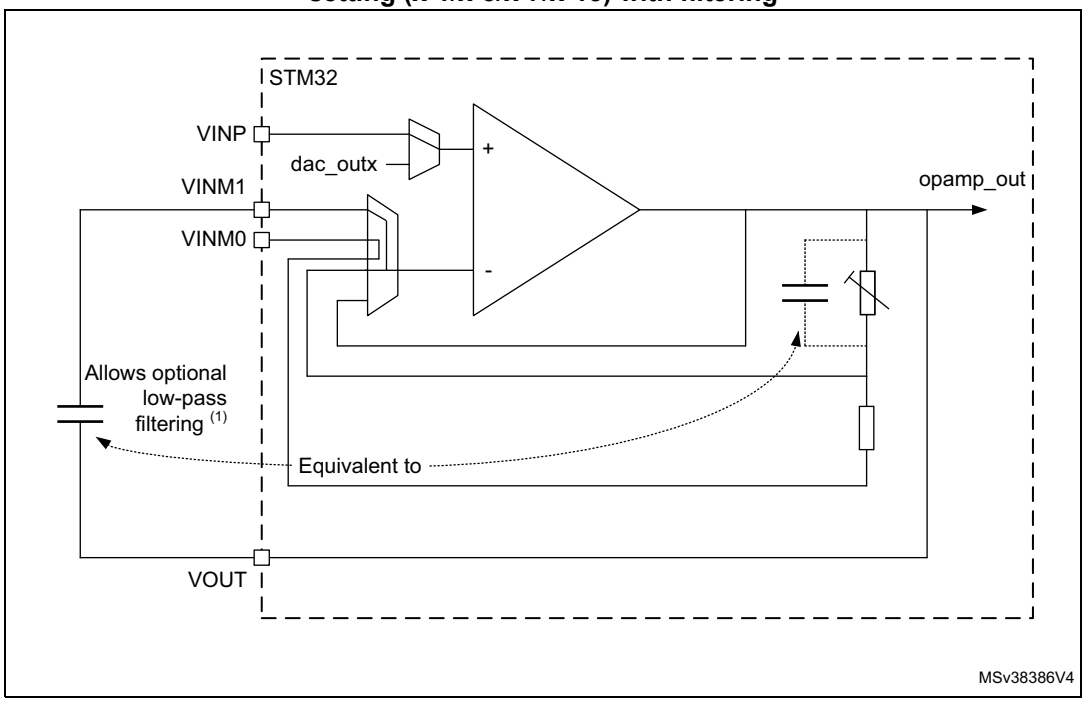
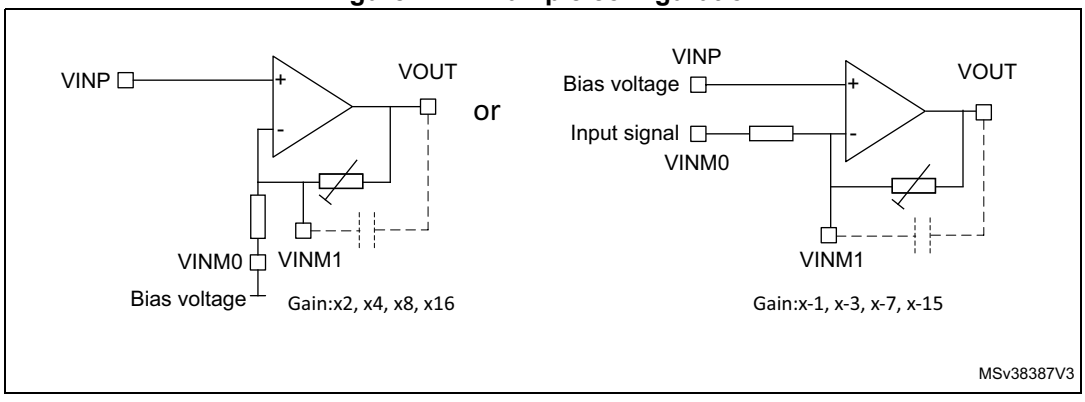


Figure 222. Example configuration



28.3.5 Calibration

The OPAMP interface continuously sends trimmed offset values to the operational amplifiers. At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

Each operational amplifier can be trimmed by the user. Specific registers allow to have different trimming values for normal mode and for high-speed mode.

The aim of the calibration is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry allows to reduce the input offset voltage to less than +/-1.5 mV within stable voltage and temperature conditions.

For each operational amplifier and each mode two trimming value needs to be trimmed, one for N differential pair and one for P differential pair.

There are two registers for trimming the offsets for each operational amplifiers, one for normal mode (OPAMPx_OTR) and one high-speed mode (OPAMPx_HSOTR). Each register is composed of five bits for P differential pair trimming and five bits for N differential pair trimming. These are the ‘user’ values.

The user is able to switch from ‘factory’ values to ‘user’ trimmed values using the USERTRIM bit in the OPAMPx_CSR register. This bit is reset at startup and so the ‘factory’ value are applied by default to the OPAMP option registers.

User is liable to change the trimming values in calibration or in functional mode.

The offset trimming registers are typically configured after the calibration operation is initialized by setting bit CALON to 1. When CALON = 1 the inputs of the operational amplifier are disconnected from the functional environment.

- Setting CALSEL to 01 initializes the offset calibration for the P differential pair (low voltage reference used).
- Resetting CALSEL to 11 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the bit CALOUT will reflect the influence of the trimming value selected by CALSEL and OPAHSM. The software should increment the TRIMOFFSETN bits in the OPAMP control register from 0x00 to the first value that causes the CALOUT bit to change from 1 to 0 in the OPAMP register. If the CALOUT bit is reset, the offset is calibrated correctly and the corresponding trimming value must be stored. The CALOUT flag needs up to 1 ms after the trimming value is changed to become steady (see $t_{OFFTRIM}$max delay specification in the electrical characteristics section of the datasheet).

Note: The closer the trimming value is to the optimum trimming value, the longer it takes to stabilize (with a maximum stabilization time remaining below 1 ms in any case).

Table 235. Operating modes and calibration

Mode	Control bits				Output	
	OPAEN	OPAHSM	CALON	CALSEL	V_{OUT}	CALOUT flag
Normal operating mode	1	0	0	X	analog	0
High-speed mode	1	1	0	X	analog	0
Power down	0	X	X	X	Z	0

Table 235. Operating modes and calibration (continued)

Mode	Control bits				Output	
	OPAEN	OPAHSM	CALON	CALSEL	V_{OUT}	CALOUT flag
Offset cal N diff for normal mode	1	0	1	11	analog	X
Offset cal P diff for normal mode	1	0	1	01	analog	X
Offset cal N diff for high-speed mode	1	1	1	11	analog	X
Offset cal P diff for high-speed mode	1	1	1	01	analog	X

Calibration procedure

Here are the steps to perform a full calibration of either one of the operational amplifiers:

1. Set the OPAEN bit in OPAMPx_CSR to 1 to enable the operational amplifier.
2. Set the USERTRIM bit in the OPAMPx_CSR register to 1.
3. Choose a calibration mode (refer to *Table 235: Operating modes and calibration*). The steps 3 to 4 will have to be repeated 4 times. For the first iteration select
   – Normal mode and N differential pair

   The above calibration mode correspond to OPAHSM=0 and CALSEL=11 in the OPAMPx_CSR register.
4. Increment TRIMOFFSETN[4:0] in OPAMPx_OTR starting from 00000b until CALOUT changes to 0 in OPAMPx_CSR.

Note: Between the write to the OPAMPx_OTR register and the read of the CALOUT value, make sure to wait for the $t_{OFFTRIM}$max delay specified in the electrical characteristics section of the datasheet, to get the correct CALOUT value.

The commutation means that the is correctly compensated and that the corresponding trim code must be saved in the OPAMPx_OTR register.

Repeat steps 3 to 4 for:

- Normal_mode and P differential pair, CALSEL=01
- High-speed mode and N differential pair
- High-speed mode and P differential pair

If a mode is not used, it is not necessary to perform the corresponding calibration.

All operational amplifier can be calibrated at the same time.

Note: During the whole calibration phase the external connection of the operational amplifier output must not pull up or down currents higher than 500 µA.

28.4 OPAMP low-power modes

Table 236. Effect of low-power modes on the OPAMP

Mode	Description
Sleep	No effect.
D2 Stop	No effect, OPAMP registers content is kept.
Standby	The OPAMP registers are powered down and must be re-initialized after exiting Standby.

28.5 OPAMP PGA gain

When OPAMP is configured as PGA mode, it can select the gain of x2,x4,x8,x16 for non-inverting mode and x-1, x-3, x-7, x-15 for inverting mode.

When OPAMP is configured as non-inverting mode, the Gain error can be refer to the product datasheet. When it is configured as inverting mode, Gain factor is defined not only the on chip feedback resistor but also the signal source output impedance. If signal source output impedance is not negligible compare to the input feedback resistance of PGA, it will create the gain error. Please refer to the PGA resistance value in the product datasheet.

28.6 OPAMP registers

28.6.1 OPAMP1 control/status register (OPAMP1_CSR)

Address: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CAL OUT	TST REF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USER TRIM	PGA_GAIN	
	r	rw											rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CALON	Res.	Res.	OPA HSM	Res.	VM_SEL		Res.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw		rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CALOUT:** Operational amplifier calibration output
OPAMP output status flag. During the calibration mode, OPAMP is used as comparator.
0: Non-inverting < inverting
1: Non-inverting > inverting

Bit 29 **TSTREF:** OPAMP calibration reference voltage output control (reserved for test)
0: INTVREF of OPAMP is not output
1: INTVREF of OPAMP is output

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **USERTRIM:** User trimming enable

This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: ‘factory’ trim code used

1: ‘user’ trim code used

Bits 17:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

0000: Non-inverting internal Gain 2, VREF- referenced

0001: Non-inverting internal Gain 4, VREF- referenced

0010: Non-inverting internal Gain 8, VREF- referenced

0011: Non-inverting internal Gain 16, VREF- referenced

0100: Non-inverting internal Gain 2 with filtering on INM0, VREF- referenced

0101: Non-inverting internal Gain 4 with filtering on INM0, VREF- referenced

0110: Non-inverting internal Gain 8 with filtering on INM0, VREF- referenced

0111: Non-inverting internal Gain 16 with filtering on INM0, VREF- referenced

1000: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias

1001: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias

1010: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias

1011: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias

1100: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias, INM1 node for filtering

1101: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias, INM1 node for filtering

1110: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias, INM1 node for filtering

1111: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias, INM1 node for filtering

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs

01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)

10: 0.5*VDDA applied on OPAMP inputs

11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode

1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: operational amplifier in normal mode

1: operational amplifier in high-speed mode

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **VM_SEL:** Inverting input selection
These bits are used only when OPAMODE = 00, 01, 10 or 11.
00: INM0 connected to OPAMP INM input
01: INM1 connected to OPAMP NM input
10: Feedback resistor is connected to OPAMP INM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting
11: opamp_out connected to OPAMP INM input (Follower mode)

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **VP_SEL:** Non inverted input selection
00: GPIO connected to OPAMPx_VINP
01: dac_outx connected to OPAMPx_VINP
10: Reserved
11: Reserved

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)
0: Normal operating mode. Non-inverting input connected to inputs.
1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable
0: operational amplifier disabled
1: operational amplifier enabled

28.6.2 OPAMP1 trimming register in normal mode (OPAMP1_OTR)

Address: 0x04

Reset value: 0x0000 XXXX (factory trimmed values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMOFFSETP					Res.	Res.	Res.	TRIMOFFSETN				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMOFFSETP[4:0]:** Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]:** Trim for NMOS differential pairs

28.6.3 OPAMP1 trimming register in high-speed mode (OPAMP1_HSOTR)

Address: 0x08

Reset value: 0x0000 XXXX (factory trimmed values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMHSOFFSETP					Res.	Res.	Res.	TRIMHSOFFSETN				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMHSOFFSETP[4:0]:** High-speed mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMHSOFFSETN[4:0]:** High-speed mode trim for NMOS differential pairs

28.6.4 OPAMP option register (OPAMP_OR)

Address: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

28.6.5 OPAMP2 control/status register (OPAMP2_CSR)

Address: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CAL OUT	TST REF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USER TRIM	PGA_GAIN	
	r	rw											rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		CALSEL		CALON	OPA MODE		OPA HSM	Res.	VM_SEL		Res.	VP_SEL		FORCE _VP	OPAEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CALOUT:** Operational amplifier calibration output

OPAMP output status flag. During the calibration mode, OPAMP is used as comparator.

0: Non-inverting < inverting

1: Non-inverting > inverting

Bit 29 **TSTREF:** OPAMP calibration reference voltage output control (reserved for test)

0: INTVREF of OPAMP is not output

1: INTVREF of OPAMP is output

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **USERTRIM:** User trimming enable

This bit allows to switch from 'factory' AOP offset trimmed values to 'user' AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: 'factory' trim code used

1: 'user' trim code used

Bits 17:14 **PGA_GAIN:** Operational amplifier Programmable amplifier gain value

0000: Non-inverting internal Gain 2, VREF- referenced

0001: Non-inverting internal Gain 4, VREF- referenced

0010: Non-inverting internal Gain 8, VREF- referenced

0011: Non-inverting internal Gain 16, VREF- referenced

0100: Non-inverting internal Gain 2 with filtering on INM0, VREF- referenced

0101: Non-inverting internal Gain 4 with filtering on INM0, VREF- referenced

0110: Non-inverting internal Gain 8 with filtering on INMINM0, VREF- referenced

0111: Non-inverting internal Gain 16 with filtering on INM0, VREF- referenced

1000: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias

1001: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias

1010: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias

1011: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias

1100: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias, INM1 node for filtering

1101: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias, INM1 node for filtering

1110: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias, INM1 node for filtering

1111: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias, INM1 node for filtering

Bits 13:12 **CALSEL:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs

01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)

10: 0.5*VDDA applied on OPAMP inputs

11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode

1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAHSM:** Operational amplifier high-speed mode
The operational amplifier must be disable to change this configuration.
0: operational amplifier in normal mode
1: operational amplifier in high-speed mode

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **VM_SEL:** Inverting input selection
These bits are used only when OPAMODE = 00, 01, 10 or 11.
00: INM0 connected to OPAMP INM input
01: INM1 connected to OPAMP INM input
10: Feedback resistor is connected to OPAMP INM input (PGA mode), Inverting input selection is depends on the PGA_GAIN setting
11: opamp_out connected to OPAMP INM input (Follower mode)

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **VP_SEL:** Non inverted input selection
00: GPIO connected to OPAMPx_VINP
01: DAC connected to OPAMPx_VINP
10: Reserved
11: Reserved

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)
0: Normal operating mode. Non-inverting input connected to inputs.
1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier Enable
0: operational amplifier disabled
1: operational amplifier enabled

28.6.6 OPAMP2 trimming register in normal mode (OPAMP2_OTR)

Address: 0x14

Reset value: 0x0000 XXXX (factory trimmed values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMOFFSETP					Res.	Res.	Res.	TRIMOFFSETN				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMOFFSETP[4:0]:** Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]:** Trim for NMOS differential pairs

28.6.7 OPAMP2 trimming register in high-speed mode (OPAMP2_HSOTR)

Address: 0x18

Reset value: 0x0000 XXXX (factory trimmed values)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMHSOFFSETP					Res.	Res.	Res.	TRIMHSOFFSETN				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMHSOFFSETP[4:0]:** High-speed mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMHSOFFSETN[4:0]:** High-speed mode trim for NMOS differential pairs

28.6.8 OPAMP register map

Table 237. OPAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**OPAMP1_CSR**	Res.	CALOUT	TSTREF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERTRIM	PGA_GAIN				CALSEL		CALON	Res.	Res.	OPAHSM	Res.	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAEN
	Reset value		0	0											0	0	0	0	0	0	0	0			0		0	0		0	0	0	0
0x04	**OPAMP1_OTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM OFFSETP[4:0]					Res.	Res.	Res.	TRIM OFFSETN[4:0]				
	Reset value																				(1)								(1)				
0x08	**OPAMP1_ HSOTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMHS OFFSETP[4:0]					Res.	Res.	Res.	TRIMHS OFFSETN[4:0]				
	Reset value																				(1)								(1)				
0x0C	**OPAMP_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x10	**OPAMP2_CSR**	Res.	CALOUT	TSTREF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERTRIM	PGA_GAIN				CALSEL		CALON	Res.	Res.	OPAHSM	Res.	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAEN
	Reset value		0	0											0	0	0	0	0	0	0	0			0		0	0		0	0	0	0
0x14	**OPAMP2_OTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM OFFSETP[4:0]					Res.	Res.	Res.	TRIM OFFSETN[4:0]				
	Reset value																				(1)								(1)				
0x18	**OPAMP2_ HSOTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMHS OFFSETP[4:0]					Res.	Res.	Res.	TRIMHS OFFSETN[4:0]				
	Reset value																				(1)								(1)				

1. Factory trimmed values.

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

29 Digital filter for sigma delta modulators (DFSDM)

29.1 Introduction

Digital filter for sigma delta modulators (DFSDM) is a high-performance module dedicated to interface external ΣΔ modulators to a microcontroller. It is featuring up to 8 external digital serial interfaces (channels) and up to 4 digital filters with flexible Sigma Delta stream digital processing options to offer up to 24-bit final ADC resolution. DFSDM also features optional parallel data stream input from internal ADC peripherals or from microcontroller memory.

An external ΣΔ modulator provides digital data stream of converted analog values from the external ΣΔ modulator analog input. This digital data stream is sent into a DFSDM input channel through a serial interface. DFSDM supports several standards to connect various ΣΔ modulator outputs: SPI interface and Manchester coded 1-wire interface (both with adjustable parameters). DFSDM module supports the connection of up to 8 multiplexed input digital serial channels which are shared with up to 4 DFSDM modules. DFSDM module also supports alternative parallel data inputs from up to 8 internal 16-bit data channels (from internal ADCs or from microcontrollers memory).

DFSDM is converting an input data stream into a final digital data word which represents an analog input value on a ΣΔ modulator analog input. The conversion is based on a configurable digital process: the digital filtering and decimation of the input serial data stream.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, length of filter, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion mode and continuous mode. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible timer triggering system can be used to control the start of conversion of DFSDM. This timing control is capable of triggering simultaneous conversions or inserting a programmable delay between conversions.

DFSDM features an analog watchdog function. Analog watchdog can be assigned to any of the input channel data stream or to final output data. Analog watchdog has its own digital filtering of input data stream to reach the required speed and resolution of watched data.

To detect short-circuit in control applications, there is a short-circuit detector. This block watches each input channel data stream for occurrence of stable data for a defined time duration (several 0’s or 1’s in an input data stream).

An extremes detector block watches final output data and stores maximum and minimum values from the output data values. The extremes values stored can be restarted by software.

Two power modes are supported: normal mode and stop mode.

29.2 DFSDM main features

- Up to 8 multiplexed input digital serial channels:
  - configurable SPI interface to connect various ΣΔ modulators
  - configurable Manchester coded 1 wire interface support
  - clock output for ΣΔ modulator(s)
- Alternative inputs from up to 8 internal digital parallel channels:
  - inputs with up to 16 bit resolution
  - internal sources: ADCs data or memory (CPU/DMA write) data streams
- Adjustable digital signal processing:
  - $Sinc^x$ filter: filter order/type (1..5), oversampling ratio (up to 1..1024)
  - integrator: oversampling ratio (1..256)
- Up to 24-bit output data resolution:
  - right bit-shifter on final data (0..31 bits)
- Signed output data format
- Automatic data offset correction (offset stored in register by user)
- Continuous or single conversion
- Start-of-conversion synchronization with:
  - software trigger
  - internal timers
  - external events
  - start-of-conversion synchronously with first DFSDM filter (DFSDM_FLT0)
- Analog watchdog feature:
  - low value and high value data threshold registers
  - own configurable $Sinc^x$ digital filter (order = 1..3, oversampling ratio = 1..32)
  - input from output data register or from one or more input digital serial channels
  - continuous monitoring independently from standard conversion
- Short-circuit detector to detect saturated analog input values (bottom and top ranges):
  - up to 8-bit counter to detect 1..256 consecutive 0's or 1's on input data stream
  - monitoring continuously each channel (8 serial channel transceiver outputs)
- Break generation on analog watchdog event or short-circuit detector event
- Extremes detector:
  - store minimum and maximum values of output data values
  - refreshed by software
- DMA may be used to read the conversion data
- Interrupts: end of conversion, overrun, analog watchdog, short-circuit, channel clock absence
- "regular" or "injected" conversions:
  - "regular" conversions can be requested at any time or even in continuous mode without having any impact on the timing of "injected" conversions

29.3 DFSDM implementation

This section describes the configuration implemented in DFSDMx.

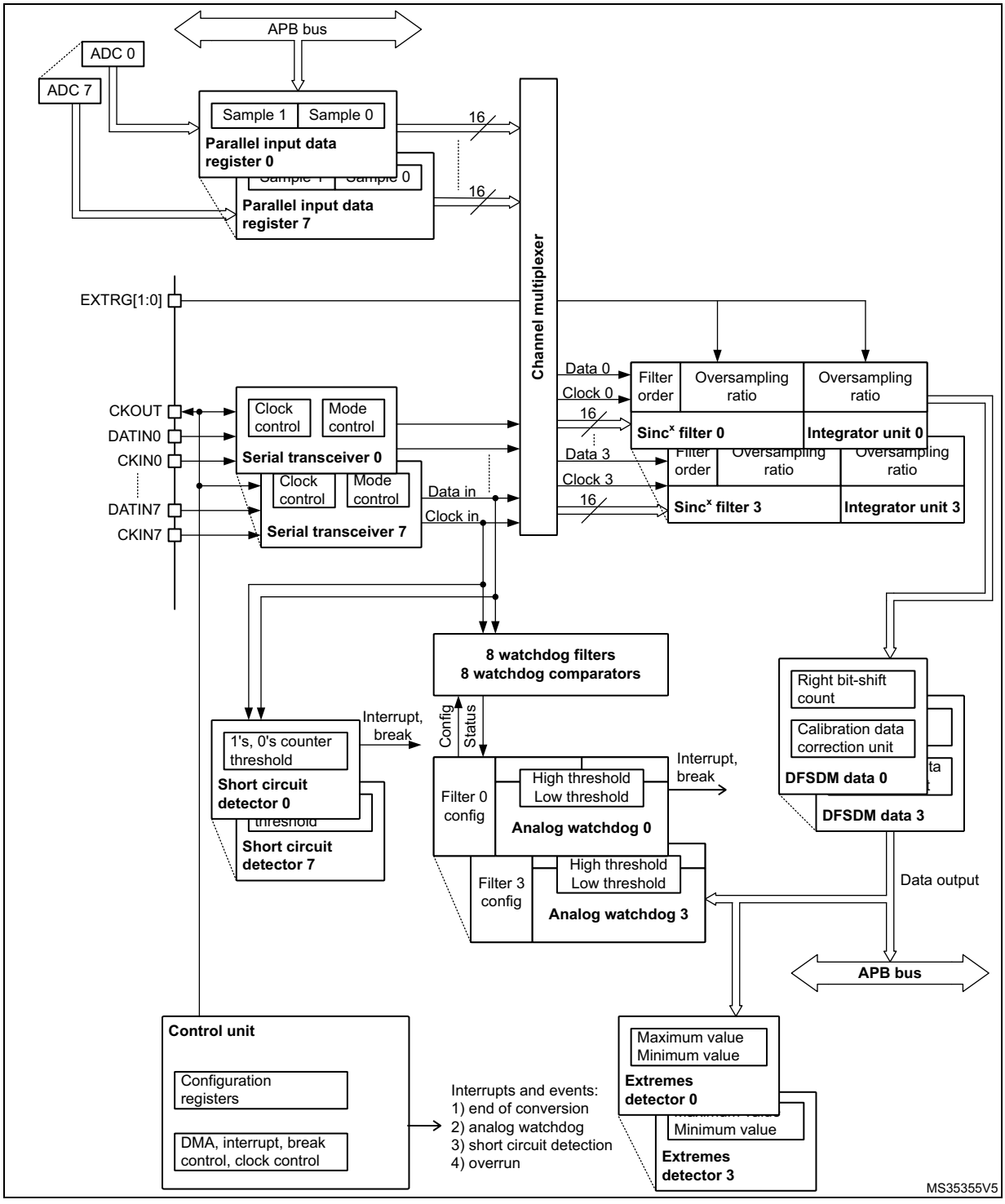
Table 238. DFSDM1 implementation

DFSDM features	DFSDM1
Number of channels	8
Number of filters	4
Input from internal ADC	X
Supported trigger sources	32
Pulses skipper	-
ID registers support	-

29.4 DFSDM functional description

29.4.1 DFSDM block diagram

Figure 223. Single DFSDM block diagram



1. This example shows 4 DFSDM filters and 8 input channels (max. configuration).

29.4.2 DFSDM pins and internal signals

Table 239. DFSDM external pins

Name	Signal Type	Remarks
VDD	Power supply	Digital power supply.
VSS	Power supply	Digital ground power supply.
CKIN[7:0]	Clock input	Clock signal provided from external ΣΔ modulator. FT input.
DATIN[7:0]	Data input	Data signal provided from external ΣΔ modulator. FT input.
CKOUT	Clock output	Clock output to provide clock signal into external ΣΔ modulator.
EXTRG[1:0]	External trigger signal	Input trigger from two EXTI signals to start analog conversion (from GPIOs: EXTI11, EXTI15).

Table 240. DFSDM internal signals

Name	Signal Type	Remarks
dfsdm_jtrg[31:0]	Internal/ external trigger signal	Input trigger from internal/external trigger sources in order to start analog conversion (from internal sources: synchronous input, from external sources: asynchronous input with synchronization). See *Table 241* for details.
dfsdm_break[3:0]	break signal output	Break signals event generation from Analog watchdog or short-circuit detector
dfsdm_dma[3:0]	DMA request signal	DMA request signal from each DFSDM_FLTx (x=0..3): end of injected conversion event.
dfsdm_it[3:0]	Interrupt request signal	Interrupt signal for each DFSDM_FLTx (x=0..3)
dfsdm_dat_adc[15:0]	ADC input data	Up to 4 internal ADC data buses as parallel inputs.

Table 241. DFSDM triggers connection

Trigger name	Trigger source
dfsdm_jtrg0	TIM1_TRGO
dfsdm_jtrg1	TIM1_TRGO2
dfsdm_jtrg2	TIM8_TRGO
dfsdm_jtrg3	TIM8_TRGO2
dfsdm_jtrg4	TIM3_TRGO
dfsdm_jtrg5	TIM4_TRGO
dfsdm_jtrg6	TIM16_OC1
dfsdm_jtrg7	TIM6_TRGO
dfsdm_jtrg8	TIM7_TRGO
dfsdm_jtrg9	HRTIM1_ADCTRG1

Table 241. DFSDM triggers connection (continued)

Trigger name	Trigger source
dfsdm_jtrg10	HRTIM1_ADCTRG3
dfsdm_jtrg[23:11]	Reserved
dfsdm_jtrg24	EXTI11
dfsdm_jtrg25	EXTI15
dfsdm_jtrg26	LPTIMER1
dfsdm_jtrg27	LPTIMER2
dfsdm_jtrg28	LPTIMER3
dfsdm_jtrg[31:29]	Reserved

Table 242. DFSDM break connection

Break name	Break destination
dfsdm_break[0]	TIM15 break
dfsdm_break[1]	TIM16 break2
dfsdm_break[2]	TIM1/TIM17/TIM8 break
dfsdm_break[3]	TIM1/TIM8 break2

29.4.3 DFSDM reset and clocks

DFSDM on-off control

The DFSDM interface is globally enabled by setting DFSDMEN=1 in the DFSDM_CH0CFGR1 register. Once DFSDM is globally enabled, all input channels (y=0..7) and digital filters DFSDM_FLTx (x=0..3) start to work if their enable bits are set (channel enable bit CHEN in DFSDM_CHyCFGR1 and DFSDM_FLTx enable bit DFEN in DFSDM_FLTxCR1).

Digital filter x DFSDM_FLTx (x=0..3) is enabled by setting DFEN=1 in the DFSDM_FLTxCR1 register. Once DFSDM_FLTx is enabled (DFEN=1), both $Sinc^x$ digital filter unit and integrator unit are reinitialized.

By clearing DFEN, any conversion which may be in progress is immediately stopped and DFSDM_FLTx is put into stop mode. All register settings remain unchanged except DFSDM_FLTxAWSR and DFSDM_FLTxISR (which are reset).

Channel y (y=0..7) is enabled by setting CHEN=1 in the DFSDM_CHyCFGR1 register. Once the channel is enabled, it receives serial data from the external ΣΔ modulator or parallel internal data sources (ADCs or CPU/DMA wire from memory).

DFSDM must be globally disabled (by DFSDMEN=0 in DFSDM_CH0CFGR1) before stopping the system clock to enter in the STOP mode of the device.

DFSDM clocks

The internal DFSDM clock $f_{DFSDMCLK}$, which is used to drive the channel transceivers, digital processing blocks (digital filter, integrator) and next additional blocks (analog watchdog, short-circuit detector, extremes detector, control block) is generated by the RCC block and is derived from the system clock SYSCLK or peripheral clock PCLK2 (see DFSDMSEL bit description in). The DFSDM clock is automatically stopped in stop mode (if DFEN = 0 for all DFSDM_FLTx, x=0..3).

The DFSDM serial channel transceivers can receive an external serial clock to sample an external serial data stream. The internal DFSDM clock must be at least 4 times faster than the external serial clock if standard SPI coding is used, and 6 times faster than the external serial clock if Manchester coding is used.

DFSDM can provide one external output clock signal to drive external ΣΔ modulator(s) clock input(s). It is provided on CKOUT pin. This output clock signal must be in the range specified in given device datasheet and is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) by programmable divider in the range 2 - 256 (CKOUTDIV in DFSDM_CH0CFGR1 register). Audio clock source is SAI1 clock selected by SAI1SEL[1:0] field in RCC configuration (see).

29.4.4 Serial channel transceivers

There are 8 multiplexed serial data channels which can be selected for conversion by each filter or Analog watchdog or Short-circuit detector. Those serial transceivers receive data stream from external ΣΔ modulator. Data stream can be sent in SPI format or Manchester coded format (see SITP[1:0] bits in DFSDM_CHyCFGR1 register).
The channel is enabled for operation by setting CHEN=1 in DFSDM_CHyCFGR1 register.

Channel inputs selection

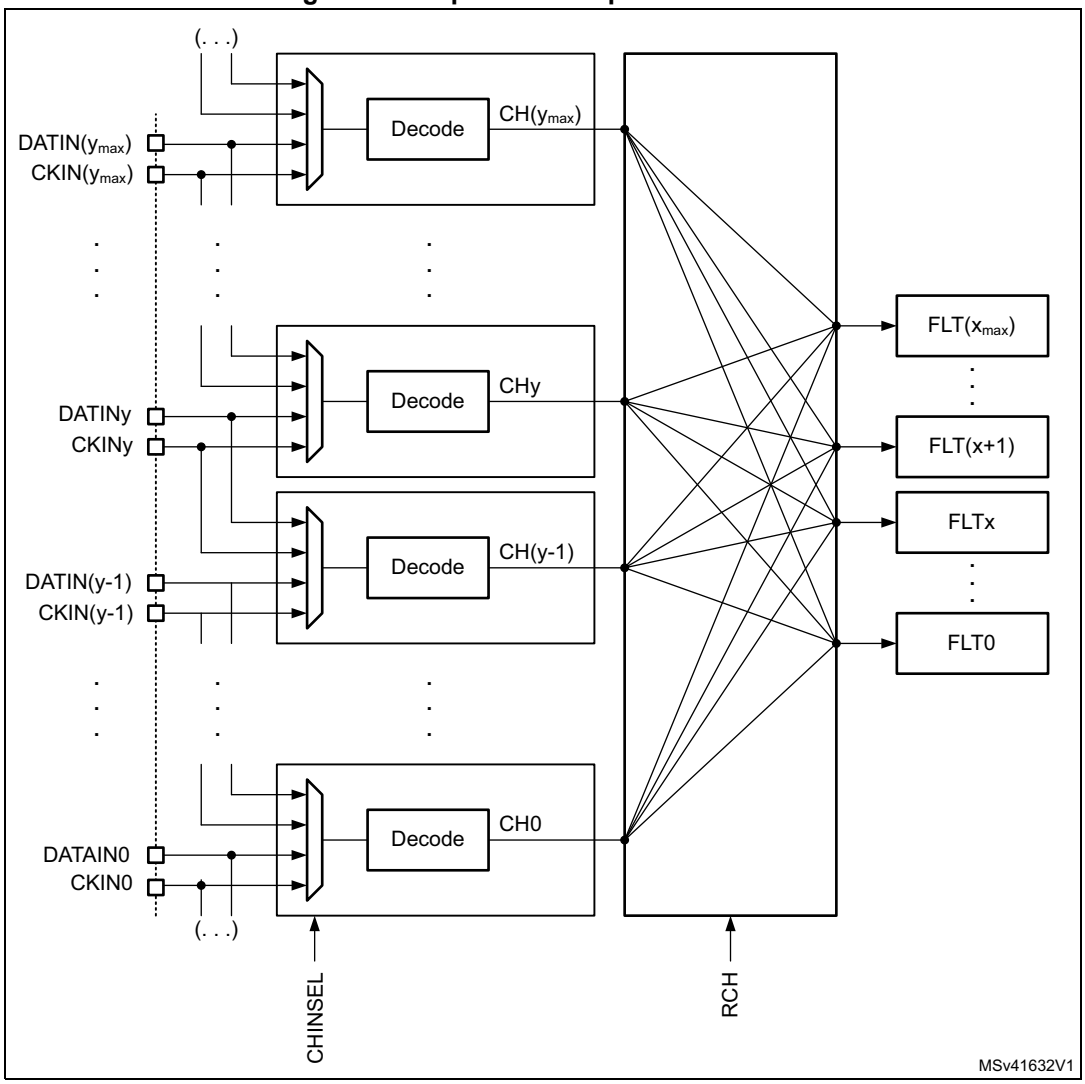
Serial inputs (data and clock signals) from DATINy and CKINy pins can be redirected from the following channel pins. This serial input channel redirection is set by CHINSEL bit in DFSDM_CHyCFGR1 register.

Channel redirection can be used to collect audio data from PDM (pulse density modulation) stereo microphone type. PDM stereo microphone has one data and one clock signal. Data signal provides information for both left and right audio channel (rising clock edge samples for left channel and falling clock edge samples for right channel).

Configuration of serial channels for PDM microphone input:

- PDM microphone signals (data, clock) will be connected to DFSDM input serial channel y (DATINy, CKOUT) pins.
- Channel y will be configured: CHINSEL = 0 (input from given channel pins: DATINy, CKINy).
- Channel (y-1) (modulo 8) will be configured: CHINSEL = 1 (input from the following channel ((y-1)+1) pins: DATINy, CKINy).
- Channel y: SITP[1:0] = 0 (rising edge to strobe data) => left audio channel on channel y.
- Channel (y-1): SITP[1:0] = 1 (falling edge to strobe data) => right audio channel on channel y-1.
- Two DFSDM filters will be assigned to channel y and channel (y-1) (to filter left and right channels from PDM microphone).

Figure 224. Input channel pins redirection



Output clock generation

A clock signal can be provided on CKOUT pin to drive external ΣΔ modulator clock inputs. The frequency of this CKOUT signal is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) divided by a predivider (see CKOUTDIV bits in DFSDM_CH0CFGR1 register). If the output clock is stopped, then CKOUT signal is set to low state (output clock can be stopped by CKOUTDIV=0 in DFSDM_CHyCFGR1 register or by DFSDMEN=0 in DFSDM_CH0CFGR1 register). The output clock stopping is performed:

- 4 system clocks after DFSDMEN is cleared (if CKOUTSRC=0)
- 1 system clock and 3 audio clocks after DFSDMEN is cleared (if CKOUTSRC=1)

Before changing CKOUTSRC the software has to wait for CKOUT being stopped to avoid glitch on CKOUT pin. The output clock signal frequency must be in the range 0 - 20 MHz.

SPI data input format operation

In SPI format, the data stream is sent in serial format through data and clock signals. Data signal is always provided from DATINy pin. A clock signal can be provided externally from CKINy pin or internally from a signal derived from the CKOUT signal source.

In case of external clock source selection (SPICKSEL[1:0]=0) data signal (on DATINy pin) is sampled on rising or falling clock edge (of CKINy pin) according SITP[1:0] bits setting (in DFSDM_CHyCFGR1 register).

Internal clock sources - see SPICKSEL[1:0] in DFSDM_CHyCFGR1 register:

- CKOUT signal:
  - For connection to external ΣΔ modulator which uses directly its clock input (from CKOUT) to generate its output serial communication clock.
  - Sampling point: on rising/falling edge according SITP[1:0] setting.
- CKOUT/2 signal (generated on CKOUT rising edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
  - Sampling point: on each second CKOUT falling edge.
- CKOUT/2 signal (generated on CKOUT falling edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).
  - Sampling point: on each second CKOUT rising edge.

Note: An internal clock source can only be used when the external ΣΔ modulator uses CKOUT signal as a clock input (to have synchronous clock and data operation).

Internal clock source usage can save CKINy pin connection (CKINy pins can be used for other purpose).

The clock source signal frequency must be in the range 0 - 20 MHz for SPI coding and less than $f_{DFSDMCLK}/4$.

Manchester coded data input format operation

In Manchester coded format, the data stream is sent in serial format through DATINy pin only. Decoded data and clock signal are recovered from serial stream after Manchester decoding. There are two possible settings of Manchester codings (see SITP[1:0] bits in DFSDM_CHyCFGR1 register):

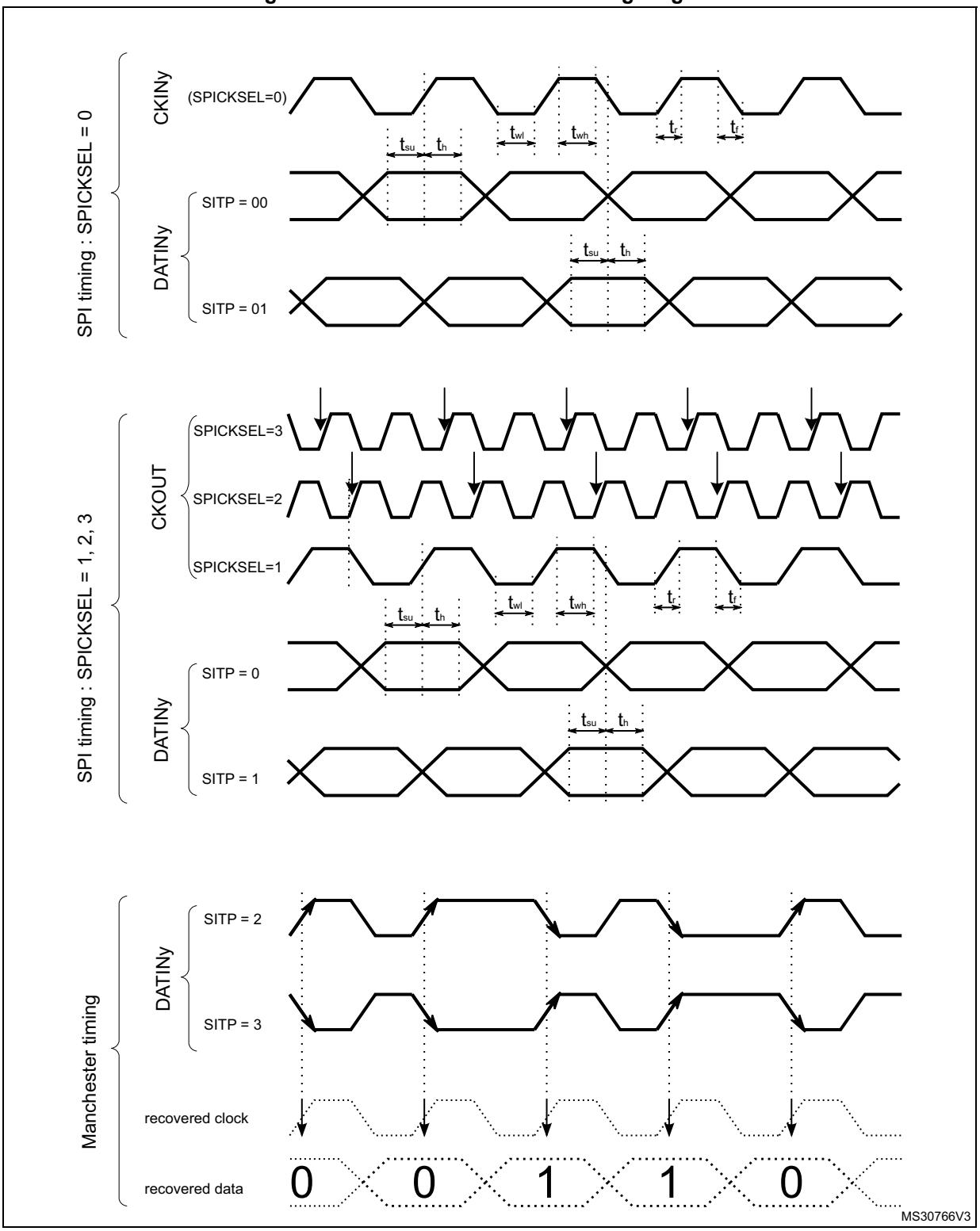
- signal rising edge = log 0; signal falling edge = log 1
- signal rising edge = log 1; signal falling edge = log 0

The recovered clock signal frequency for Manchester coding must be in the range 0 - 10 MHz and less than $f_{DFSDMCLK}/6$.

To correctly receive Manchester coded data, the CKOUTDIV divider (in DFSDM_CH0CFGR1 register) must be set with respect to expected Manchester data rate according formula:

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

Figure 225. Channel transceiver timing diagrams



Clock absence detection

Channels serial clock inputs can be checked for clock absence/presence to ensure the correct operation of conversion and error reporting. Clock absence detection can be enabled or disabled on each input channel y by bit CKABEN in DFSDM_CHyCFGR1 register. If enabled, then this clock absence detection is performed continuously on a given channel. A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock error (see CKABF[7:0] in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed. Clock absence status bit CKABF[y] is set also by hardware when corresponding channel y is disabled (if CHEN[y] = 0 then CKABF[y] is held in set state).

When a clock absence event has occurred, the data conversion (and/or analog watchdog and short-circuit detector) provides incorrect data. The user should manage this event and discard given data while a clock absence is reported.

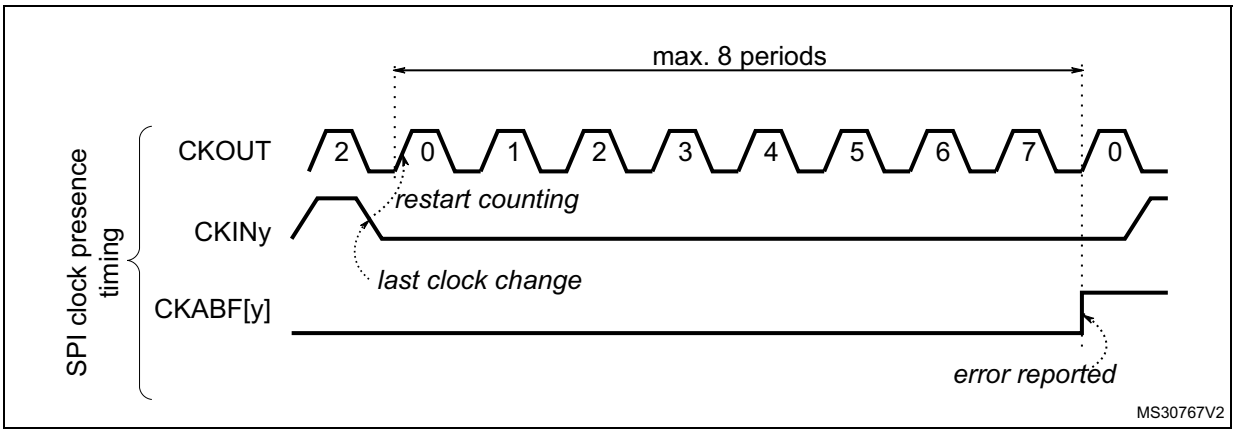
The clock absence feature is available only when the system clock is used for the CKOUT signal (CKOUTSRC=0 in DFSDM_CH0CFGR1 register).

When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM_FLT0ICR register). The software sequence concerning clock absence detection feature should be:

- Enable given channel by CHEN = 1
- Try to clear the clock absence flag (by CLRCKABF = 1) until the clock absence flag is really cleared (CKABF = 0). At this time, the transceiver is synchronized (signal clock is valid) and is able to receive data.
- Enable the clock absence feature CKABEN = 1 and the associated interrupt CKABIE = 1 to detect if the SPI clock is lost or Manchester data edges are missing.

If SPI data format is used, then the clock absence detection is based on the comparison of an external input clock with an output clock generation (CKOUT signal). The external input clock signal into the input channel must be changed at least once per 8 signal periods of CKOUT signal (which is controlled by CKOUTDIV field in DFSDM_CH0CFGR1 register).

Figure 226. Clock absence timing diagram for SPI



If Manchester data format is used, then the clock absence means that the clock recovery is unable to perform from Manchester coded signal. For a correct clock recovery, it is first necessary to receive data with 1 to 0 or 0 to 1 transition (see *Figure 228* for Manchester synchronization).

The detection of a clock absence in Manchester coding (after a first successful synchronization) is based on changes comparison of coded serial data input signal with output clock generation (CKOUT signal). There must be a voltage level change on DATINy pin during 2 periods of CKOUT signal (which is controlled by CKOUTDIV bits in DFSDM_CH0CFGR1 register). This condition also defines the minimum data rate to be able to correctly recover the Manchester coded data and clock signals.

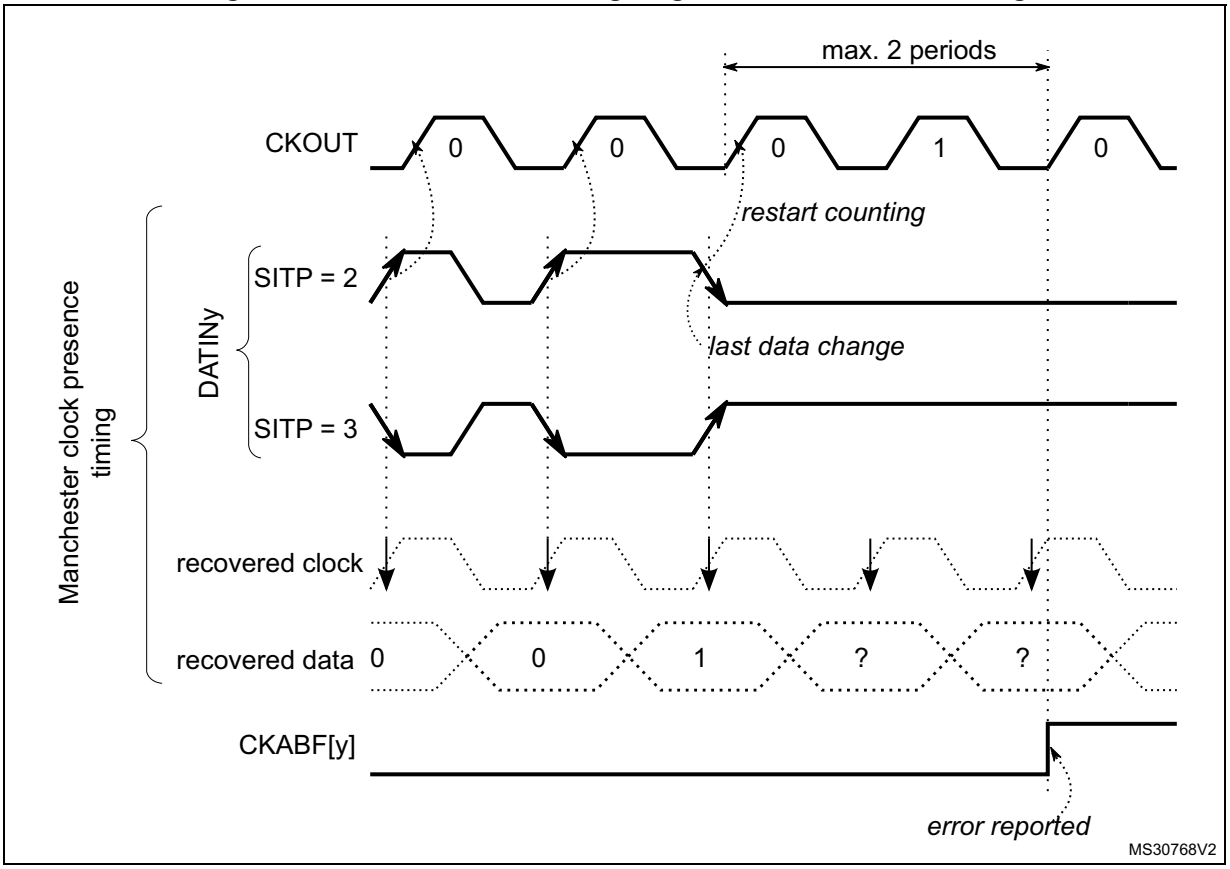
The maximum data rate of Manchester coded data must be less than the CKOUT signal.

So to correctly receive Manchester coded data, the CKOUTDIV divider must be set according the formula:

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock recovery error (see CKABF[7:0] in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed.

Figure 227. Clock absence timing diagram for Manchester coding



Manchester/SPI code synchronization

The Manchester coded stream must be synchronized the first time after enabling the channel (CHEN=1 in DFSDM_CHyCFGR1 register). The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (to be able to detect valid data edge) is received. The end of the synchronization can be checked by polling CKABF[y]=0 for a given channel after it has been cleared by CLRCKABF[y] in DFSDM_FLT0ICR, following the software sequence detailed hereafter:

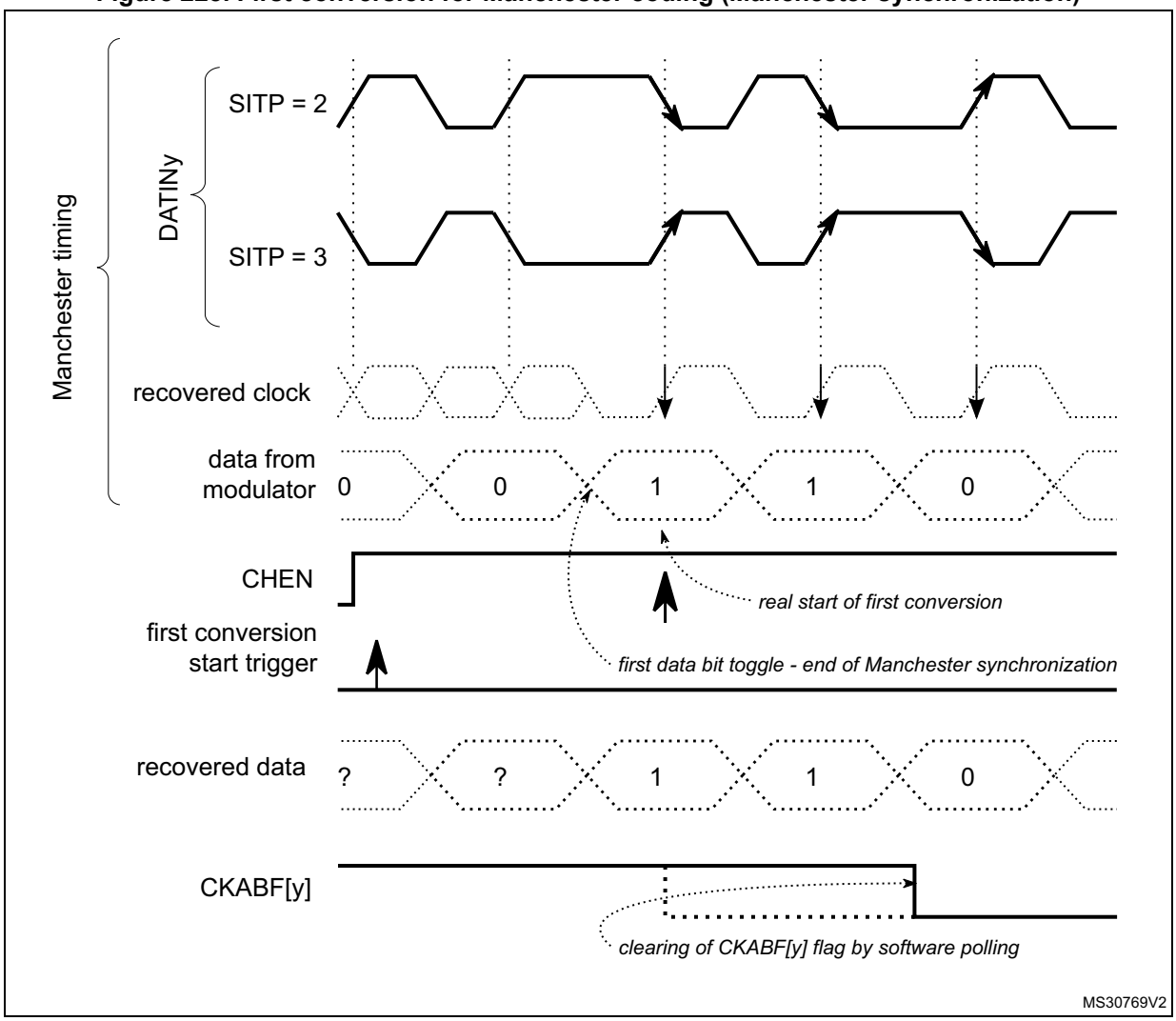
CKABF[y] flag is cleared by setting CLRCKABF[y] bit. If channel y is not yet synchronized the hardware immediately set the CKABF[y] flag. Software is then reading back the CKABF[y] flag and if it is set then perform again clearing of this flag by setting CLRCKABF[y] bit. This software sequence (polling of CKABF[y] flag) continues until CKABF[y] flag is set (signalizing that Manchester stream is synchronized). To be able to synchronize/receive Manchester coded data the CKOUTDIV divider (in DFSDM_CH0CFGR1 register) must be set with respect to expected Manchester data rate according the formula below.

$$((CKOUTDIV + 1) \times T_{SYSCLK}) < T_{Manchester\ clock} < (2 \times CKOUTDIV \times T_{SYSCLK})$$

SPI coded stream is synchronized after first detection of clock input signal (valid rising/falling edge).

Note: When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM_FLT0ICR register).

Figure 228. First conversion for Manchester coding (Manchester synchronization)



External serial clock frequency measurement

The measuring of a channel serial clock input frequency provides a real data rate from an external ΣΔ modulator, which is important for application purposes.

An external serial clock input frequency can be measured by a timer counting DFSDM clocks ($f_{DFSDMCLK}$) during one conversion duration. The counting starts at the first input data clock after a conversion trigger (regular or injected) and finishes by last input data clock before conversion ends (end of conversion flag is set). Each conversion duration (time between first serial sample and last serial sample) is updated in counter CNVCNT[27:0] in register DFSDM_FLTxCNVTIMR when the conversion finishes (JEOCF=1 or REOCF=1). The user can then compute the data rate according to the digital filter settings (FORD, FOSR, IOSR, FAST). The external serial frequency measurement is stopped only if the filter is bypassed (FOSR=0, only integrator is active, CNVCNT[27:0]=0 in DFSDM_FLTxCNVTIMR register).

In case of parallel data input (*Section 29.4.6: Parallel data inputs*) the measured frequency is the average input data rate during one conversion.

Note: When conversion is interrupted (e.g. by disabling/enabling the selected channel) the interruption time is also counted in CNVCNT[27:0]. Therefore it is recommended to not interrupt the conversion for correct conversion duration result.

Conversion times:

injected conversion or regular conversion with FAST = 0 (or first conversion if FAST=1):

for $Sinc^x$ filters (x=1..5):

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$

for FastSinc filter:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$

regular conversion with FAST = 1 (except first conversion):

for $Sinc^x$ and FastSinc filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

where:

- f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate (in case of parallel data input)
- F_{OSR} is the filter oversampling ratio: F_{OSR} = FOSR[9:0]+1 (see DFSDM_FLTxFCR register)
- I_{OSR} is the integrator oversampling ratio: I_{OSR} = IOSR[7:0]+1 (see DFSDM_FLTxFCR register)
- F_{ORD} is the filter order: F_{ORD} = FORD[2:0] (see DFSDM_FLTxFCR register)

Channel offset setting

Each channel has its own offset setting (in register) which is finally subtracted from each conversion result (injected or regular) from a given channel. Offset correction is performed after the data right bit shift. The offset is stored as a 24-bit signed value in OFFSET[23:0] field in DFSDM_CHyCFGR2 register.

Data right bit shift

To have the result aligned to a 24-bit value, each channel defines a number of right bit shifts which will be applied on each conversion result (injected or regular) from a given channel. The data bit shift number is stored in DTRBS[4:0] bits in DFSDM_CHyCFGR2 register.

The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained, in order to have valid 24-bit signed format of result data.

29.4.5 Configuring the input serial interface

The following parameters must be configured for the input serial interface:

- **Output clock predivider**. There is a programmable predivider to generate the output clock from DFSDM clock (2 - 256). It is defined by CKOUTDIV[7:0] bits in DFSDM_CH0CFGR1 register.
- **Serial interface type and input clock phase.** Selection of SPI or Manchester coding and sampling edge of input clock. It is defined by SITP [1:0] bits in DFSDM_CHyCFGR1 register.
- **Input clock source.** External source from CKINy pin or internal from CKOUT pin. It is defined by SPICKSEL[1:0] field in DFSDM_CHyCFGR1 register.
- **Final data right bit-shift.** Defines the final data right bit shift to have the result aligned to a 24-bit value. It is defined by DTRBS[4:0] in DFSDM_CHyCFGR2 register.
- **Channel offset per channel.** Defines the analog offset of a given serial channel (offset of connected external ΣΔ modulator). It is defined by OFFSET[23:0] bits in DFSDM_CHyCFGR2 register.
- **short-circuit detector and clock absence per channel enable.** To enable or disable the short-circuit detector (by SCDEN bit) and the clock absence monitoring (by CKABEN bit) on a given serial channel in register DFSDM_CHyCFGR1.
- **Analog watchdog filter and short-circuit detector threshold settings.** To configure channel analog watchdog filter parameters and channel short-circuit detector parameters. Configurations are defined in DFSDM_CHyAWSCDR register.

29.4.6 Parallel data inputs

Each input channel provides a register for 16-bit parallel data input (besides serial data input). Each 16-bit parallel input can be sourced from internal data sources only:

- internal ADC results[(a)]
- direct CPU/DMA writing.

The selection for using serial or parallel data input for a given channel is done by field DATMPX[1:0] of DFSDM_CHyCFGR1 register. In DATMPX[1:0] is also defined the parallel data source: internal ADC[(a)] or direct write by CPU/DMA.

Each channel contains a 32-bit data input register DFSDM_CHyDATINR in which it can be written a 16-bit data. Data are in 16-bit signed format. Those data can be used as input to the digital filter which is accepting 16-bit parallel data.

If serial data input is selected (DATMPX[1:0] = 0), the DFSDM_CHyDATINR register is write protected.

Input from internal ADC[(a)]

In case of ADC data parallel input (DATMPX[1:0]=1) the ADC[y+1] result is assigned to channel y input (ADC1 is filling DFSDM_CHDATIN0R register, ADC2 is filling DFSDM_CHDATIN1R register, ... , ADC8 is filling DFSDM_CHDATIN7R register). End of conversion event from ADC[y+1] causes update of channel y data (parallel data from ADC[y+1] are put as next sample to digital filter). Data from ADC[y+1] is written into DFSDM_CHyDATINR register (field INDAT0[15:0]) when end of conversion event occurred.

a. ADC1 and ADC2 only.

The setting of data packing mode (DATPACK[1:0] in the DFSDM_CHyCFGR1 register) has no effect in case of ADC data input.

Note: Extension of ADC specification: in case the internal ADC is configured in interleaved mode (e.g. ADC1 together with ADC2 - see ADC specification) then each result from ADC1 or from ADC2 will come to the same 16-bit bus - to the bus of ADC1 - which is coming into DFSDM channel 0 (fixed connection). So there will be double input data rate into DFSDM channel 0 (even samples come from ADC1 and odd samples from ADC2). Channel 1 associated with ADC2 will be free.

Input from memory (direct CPU/DMA write)

The direct data write into DFSDM_CHyDATINR register by CPU or DMA (DATMPX[1:0]=2) can be used as data input in order to process digital data streams from memory or peripherals.

Data can be written by CPU or DMA into DFSDM_CHyDATINR register:

1. **CPU data write:**

   Input data are written directly by CPU into DFSDM_CHyDATINR register.

2. **DMA data write:**

   The DMA should be configured in memory-to-memory transfer mode to transfer data from memory buffer into DFSDM_CHyDATINR register. The destination memory address is the address of DFSDM_CHyDATINR register. Data are transferred at DMA transfer speed from memory to DFSDM parallel input.

   This DMA transfer is different from DMA used to read DFSDM conversion results. Both DMA can be used at the same time - first DMA (configured as memory-to-memory transfer) for input data writings and second DMA (configured as peripheral-to-memory transfer) for data results reading.

The accesses to DFSDM_CHyDATINR can be either 16-bit or 32-bit wide, allowing to load respectively one or two samples in one write operation. 32-bit input data register (DFSDM_CHyDATINR) can be filled with one or two 16-bit data samples, depending on the data packing operation mode defined in field DATPACK[1:0] of DFSDM_CHyCFGR1 register:

1. **Standard mode (DATPACK[1:0]=0):**

   Only one sample is stored in field INDAT0[15:0] of DFSDM_CHyDATINR register which is used as input data for channel y. The upper 16 bits (INDAT1[15:0]) are ignored and write protected. The digital filter must perform one input sampling (from INDAT0[15:0]) to empty data register after it has been filled by CPU/DMA. This mode is used together with 16-bit CPU/DMA access to DFSDM_CHyDATINR register to load one sample per write operation.

2. **Interleaved mode (DATPACK[1:0]=1):**

   DFSDM_CHyDATINR register is used as a two sample buffer. The first sample is stored in INDAT0[15:0] and the second sample is stored in INDAT1[15:0]. The digital filter must perform two input samplings from channel y to empty DFSDM_CHyDATINR register. This mode is used together with 32-bit CPU/DMA access to DFSDM_CHyDATINR register to load two samples per write operation.

3. **Dual mode (DATPACK[1:0]=2):**

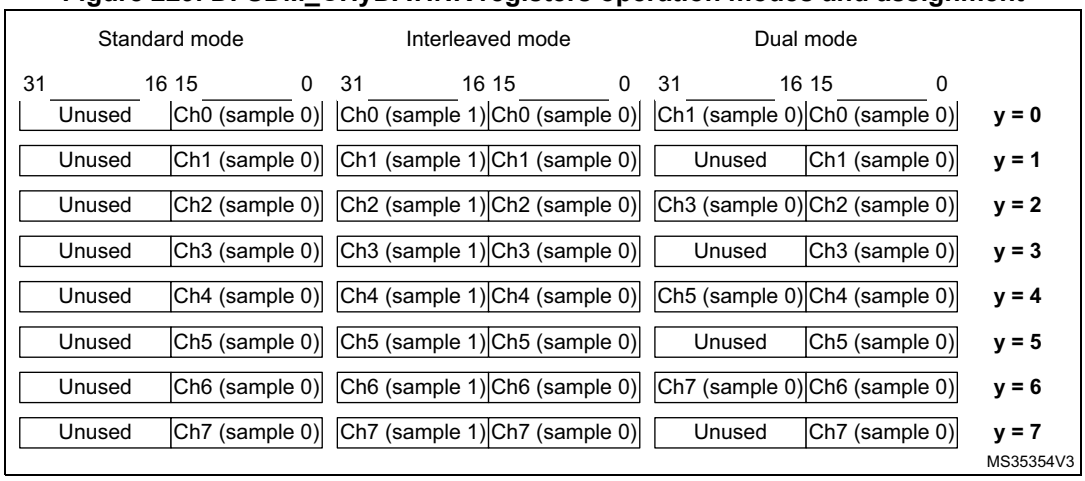
   Two samples are written into DFSDM_CHyDATINR register. The data INDAT0[15:0] is for channel y, the data in INDAT1[15:0] is for channel y+1. The data in INDAT1[15:0] is automatically copied INDAT0[15:0] of the following (y+1) channel data register

DFSDM_CH[y+1]DATINR). The digital filters must perform two samplings - one from channel y and one from channel (y+1) - in order to empty DFSDM_CHyDATINR registers.

Dual mode setting (DATPACK[1:0]=2) is available only on even channel numbers (y = 0, 2, 4, 6). If odd channel (y = 1, 3, 5, 7) is set to Dual mode then both INDAT0[15:0] and INDAT1[15:0] parts are write protected for this channel. If even channel is set to Dual mode then the following odd channel must be set into Standard mode (DATPACK[1:0]=0) for correct cooperation with even channels.

See *Figure 229* for DFSDM_CHyDATINR registers data modes and assignments of data samples to channels.

Figure 229. DFSDM_CHyDATINR registers operation modes and assignment



The write into DFSDM_CHyDATINR register to load one or two samples must be performed after the selected input channel (channel y) is enabled for data collection (starting conversion for channel y). Otherwise written data are lost for next processing.

For example: for single conversion and interleaved mode, do not start writing pair of data samples into DFSDM_CHyDATINR before the single conversion is started (any data present in the DFSDM_CHyDATINR before starting a conversion is discarded).

29.4.7 Channel selection

There are 8 multiplexed channels which can be selected for conversion using the injected channel group and/or using the regular channel.

The **injected channel group** is a selection of any or all of the 8 channels. JCHG[7:0] in the DFSDM_FLTxJCHGR register selects the channels of the injected group, where JCHG[y]=1 means that channel y is selected.

Injected conversions can operate in scan mode (JSCAN=1) or single mode (JSCAN=0). In scan mode, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first, followed immediately by the next higher channel until all the channels selected by JCHG[7:0] have been converted. In single mode (JSCAN=0), only one channel from the selected channels is converted, and the channel selection is moved to the next channel. Writing to JCHG[7:0] if JSCAN=0 resets the channel selection to the lowest selected channel.

Injected conversions can be launched by software or by a trigger. They are never interrupted by regular conversions.

The **regular channel** is a selection of just one of the 8 channels. RCH[2:0] in the DFSDM_FLTxCR1 register indicates the selected channel.

Regular conversions can be launched only by software (not by a trigger). A sequence of continuous regular conversions is temporarily interrupted when an injected conversion is requested.

Performing a conversion on a disabled channel (CHEN=0 in DFSDM_CHyCFGR1 register) causes that the conversion will never end - because no input data is provided (with no clock signal). In this case, it is necessary to enable a given channel (CHEN=1 in DFSDM_CHyCFGR1 register) or to stop the conversion by DFEN=0 in DFSDM_FLTxCR1 register.

29.4.8 Digital filter configuration

DFSDM contains a $Sinc^x$ type digital filter implementation. This $Sinc^x$ filter performs an input digital data stream filtering, which results in decreasing the output data rate (decimation) and increasing the output data resolution. The $Sinc^x$ digital filter is configurable in order to reach the required output data rates and required output data resolution. The configurable parameters are:

- Filter order/type: (see FORD[2:0] bits in DFSDM_FLTxFCR register):
  - FastSinc
  - $Sinc^1$
  - $Sinc^2$
  - $Sinc^3$
  - $Sinc^4$
  - $Sinc^5$
- Filter oversampling/decimation ratio (see FOSR[9:0] bits in DFSDM_FLTxFCR register):
  - FOSR = 1-1024 - for FastSinc filter and $Sinc^x$ filter x = F_{ORD} = 1..3
  - FOSR = 1-215 - for $Sinc^x$ filter x = F_{ORD} = 4
  - FOSR = 1-73 - for $Sinc^x$ filter x = F_{ORD} = 5

The filter has the following transfer function (impulse response in H domain):

- $Sinc^x$ filter type: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^x$$
- FastSinc filter type: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

Figure 230. Example: Sinc^3 filter response

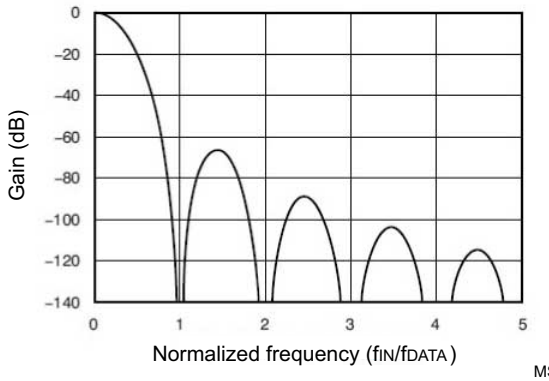


Table 243. Filter maximum output resolution (peak data values from filter output) for some FOSR values

FOSR	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
x	+/- x	+/- x^2	+/- 2x^2	+/- x^3	+/- x^4	+/- x^5
4	+/- 4	+/- 16	+/- 32	+/- 64	+/- 256	+/- 1024
8	+/- 8	+/- 64	+/- 128	+/- 512	+/- 4096	-
32	+/- 32	+/- 1024	+/- 2048	+/- 32768	+/- 1048576	+/- 33554432
64	+/- 64	+/- 4096	+/- 8192	+/- 262144	+/- 16777216	+/- 1073741824
128	+/- 128	+/- 16384	+/- 32768	+/- 2097152	+/- 268435456	Result can overflow on full scale input (> 32-bit signed integer)
256	+/- 256	+/- 65536	+/- 131072	+/- 16777216		
1024	+/- 1024	+/- 1048576	+/- 2097152	+/- 1073741824		

For more information about Sinc filter type properties and usage, it is recommended to study the theory about digital filters (more resources can be downloaded from internet).

29.4.9 Integrator unit

The integrator performs additional decimation and a resolution increase of data coming from the digital filter. The integrator simply performs the sum of data from a digital filter for a given number of data samples from a filter.

The integrator oversampling ratio parameter defines how many data counts will be summed to one data output from the integrator. IOSR can be set in the range 1-256 (see IOSR[7:0] bits description in DFSDM_FLTxFCR register).

Table 244. Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and $Sinc^3$ filter type (largest data)

IOSR	$Sinc^1$	$Sinc^2$	FastSinc	$Sinc^3$	$Sinc^4$	$Sinc^5$
x	+/- FOSR. x	+/- $FOSR^2$. x	+/- 2.$FOSR^2$. x	+/- $FOSR^3$. x	+/- $FOSR^4$. x	+/- $FOSR^5$. x
4	-	-	-	+/- 67 108 864	-	-
32	-	-	-	+/- 536 870 912	-	-
128	-	-	-	+/- 2 147 483 648	-	-
256	-	-	-	+/- 2^{32}	-	-

29.4.10 Analog watchdog

The analog watchdog purpose is to trigger an external signal (break or interrupt) when an analog signal reaches or crosses given maximum and minimum threshold values. An interrupt/event/break generation can then be invoked.

Each analog watchdog will supervise serial data receiver outputs (after the analog watchdog filter on each channel) or data output register (current injected or regular conversion result) according to AWFSEL bit setting (in DFSDM_FLTxCR1 register). The input channels to be monitored or not by the analog watchdog x will be selected by AWDCH[7:0] in DFSDM_FLTxCR2 register.

Analog watchdog conversions on input channels are independent from standard conversions. In this case, the analog watchdog uses its own filters and signal processing on each input channel independently from the main injected or regular conversions. Analog watchdog conversions are performed in a continuous mode on the selected input channels in order to watch channels also when main injected or regular conversions are paused (RCIP = 0, JCIP = 0).

There are high and low threshold registers which are compared with given data values (set by AWHT[23:0] bits in DFSDM_FLTxAWHTR register and by AWLT[23:0] bits in DFSDM_FLTxAWLTR register).

There are 2 options for comparing the threshold registers with the data values

- Option1: in this case, the input data are taken from final output data register (AWFSEL=0). This option is characterized by:
  - high input data resolution (up to 24-bits)
  - slow response time - inappropriate for fast response applications like overcurrent detection
  - for the comparison the final data are taken after bit shifting and offset data correction
  - final data are available only after main regular or injected conversions are performed
  - can be used in case of parallel input data source (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register)
- Option2: in this case, the input data are taken from any serial data receivers output (AWFSEL=1). This option is characterized by:
  - input serial data are processed by dedicated analog watchdog $Sinc^x$ channel filters with configurable oversampling ratio (1..32) and filter order (1..3) (see AWFOSR[4:0] and AWFORD[1:0] bits setting in DFSDM_CHyAWSCDR register)
  - lower resolution (up to 16-bit)
  - fast response time - appropriate for applications which require a fast response like overcurrent/overvoltage detection)
  - data are available in continuous mode independently from main regular or injected conversions activity

In case of input channels monitoring (AWFSEL=1), the data for comparison to threshold is taken from channels selected by AWDCH[7:0] field (DFSDM_FLTxCR2 register). Each of the selected channels filter result is compared to one threshold value pair (AWHT[23:0] / AWLT[23:0]). In this case, only higher 16 bits (AWHT[23:8] / AWLT[23:8]) define the 16-bit threshold compared with the analog watchdog filter output because data coming from the analog watchdog filter is up to a 16-bit resolution. Bits AWHT[7:0] / AWLT[7:0] are not taken into comparison in this case (AWFSEL=1).

Parameters of the analog watchdog filter configuration for each input channel are set in DFSDM_CHyAWSCDR register (filter order AWFORD[1:0] and filter oversampling ratio AWFOSR[4:0]).

Each input channel has its own comparator which compares the analog watchdog data (from analog watchdog filter) with analog watchdog threshold values (AWHT/AWLT). When several channels are selected (field AWDCH[7:0] field of DFSDM_FLTxCR2 register), several comparison requests may be received simultaneously. In this case, the channel request with the lowest number is managed first and then continuing to higher selected channels. For each channel, the result can be recorded in a separate flag (fields AWHTF[7:0], AWLTF[7:0] of DFSDM_FLTxAWSR register). Each channel request is executed in 8 DFSDM clock cycles. So, the bandwidth from each channel is limited to 8 DFSDM clock cycles (if AWDCH[7:0] = 0xFF). Because the maximum input channel sampling clock frequency is the DFSDM clock frequency divided by 4, the configuration AWFOSR = 0 (analog watchdog filter is bypassed) cannot be used for analog watchdog feature at this input clock speed. Therefore user must properly configure the number of watched channels and analog watchdog filter parameters with respect to input sampling clock speed and DFSDM frequency.

Analog watchdog filter data for given channel y is available for reading by firmware on field WDATA[15:0] in DFSDM_CHyWDATR register. That analog watchdog filter data is converted continuously (if CHEN=1 in DFSDM_CHyCFGR1 register) with the data rate given by the analog watchdog filter setting and the channel input clock frequency.

The analog watchdog filter conversion works like a regular Fast Continuous Conversion without the intergator. The number of serial samples needed for one result from analog watchdog filter output (at channel input clock frequency f_{CKIN}):

first conversion:

for $Sinc^x$ filters (x=1..5): number of samples = [F_{OSR} * F_{ORD} + F_{ORD} + 1]

for FastSinc filter: number of samples = [F_{OSR} * 4 + 2 + 1]

next conversions:

for $Sinc^x$ and FastSinc filters: number of samples = [FOSR * IOSR]

where:

F_{OSR} ..... filter oversampling ratio: F_{OSR} = AWFOSR[4:0]+1 (see DFSDM_CHyAWSCDR register)

F_{ORD} ..... the filter order: F_{ORD} = AWFORD[1:0] (see DFSDM_CHyAWSCDR register)

In case of output data register monitoring (AWFSEL=0), the comparison is done after a right bit shift and an offset correction of final data (see OFFSET[23:0] and DTRBS[4:0] fields in DFSDM_CHyCFGR2 register). A comparison is performed after each injected or regular end of conversion for the channels selected by AWDCH[7:0] field (in DFSDM_FLTxCR2 register).

The status of an analog watchdog event is signalized in DFSDM_FLTxAWSR register where a given event is latched. AWHTF[y]=1 flag signalizes crossing AWHT[23:0] value on channel y. AWLTF[y]=1 flag signalizes crossing AWLT[23:0] value on channel y. Latched events in DFSDM_FLTxAWSR register are cleared by writing ‘1’ into the corresponding clearing bit CLRAWHTF[y] or CLRAWLTF[y] in DFSDM_FLTxAWCFR register.

The global status of an analog watchdog is signalized by the AWDF flag bit in DFSDM_FLTxISR register (it is used for the fast detection of an interrupt source). AWDF=1 signalizes that at least one watchdog occurred (AWHTF[y]=1 or AWLTF[y]=1 for at least one channel). AWDF bit is cleared when all AWHTF[7:0] and AWLTF[7:0] are cleared.

An analog watchdog event can be assigned to break output signal. There are four break outputs to be assigned to a high or low threshold crossing event (dfsdm_break[3:0]). The break signal assignment to a given analog watchdog event is done by BKAWH[3:0] and BKAWL[3:0] fields in DFSDM_FLTxAWHTR and DFSDM_FLTxAWLTR register.

29.4.11 Short-circuit detector

The purpose of a short-circuit detector is to signalize with a very fast response time if an analog signal reached saturated values (out of full scale ranges) and remained on this value given time. This behavior can detect short-circuit or open circuit errors (e.g. overcurrent or overvoltage). An interrupt/event/break generation can be invoked.

Input data into a short-circuit detector is taken from channel transceiver outputs.

There is an upcounting counter on each input channel which is counting consecutive 0’s or 1’s on serial data receiver outputs. A counter is restarted if there is a change in the data stream received - 1 to 0 or 0 to 1 change of data signal. If this counter reaches a short-circuit threshold register value (SCDT[7:0] bits in DFSDM_CHyAWSCDR register), then a short-

circuit event is invoked. Each input channel has its short-circuit detector. Any channel can be selected to be continuously monitored by setting the SCDEN bit (in DFSDM_CHyCFGR1 register) and it has its own short-circuit detector settings (threshold value in SCDT[7:0] bits, status bit SCDF[7:0], status clearing bits CLRSCDF[7:0]). Status flag SCDF[y] is cleared also by hardware when corresponding channel y is disabled (CHEN[y] = 0).

On each channel, a short-circuit detector event can be assigned to break output signal dfsdm_break[3:0]. There are four break outputs to be assigned to a short-circuit detector event. The break signal assignment to a given channel short-circuit detector event is done by BKSCD[3:0] field in DFSDM_CHyAWSCDR register.

Short circuit detector cannot be used in case of parallel input data channel selection (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register).

Four break outputs are totally available (shared with the analog watchdog function).

29.4.12 Extreme detector

The purpose of an extremes detector is to collect the minimum and maximum values of final output data words (peak to peak values).

If the output data word is higher than the value stored in the extremes detector maximum register (EXMAX[23:0] bits in DFSDM_FLTxEXMAX register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMAXCH[2:0] bits (in DFSDM_FLTxEXMAX register) .

If the output data word is lower than the value stored in the extremes detector minimum register (EXMIN[23:0] bits in DFSDM_FLTxEXMIN register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMINCH[2:0] bits (in DFSDM_FLTxEXMIN register).

The minimum and maximum register values can be refreshed by software (by reading given DFSDM_FLTxEXMAX or DFSDM_FLTxEXMIN register). After refresh, the extremes detector minimum data register DFSDM_FLTxEXMIN is filled with 0x7FFFFF (maximum positive value) and the extremes detector maximum register DFSDM_FLTxEXMAX is filled with 0x800000 (minimum negative value).

The extremes detector performs a comparison after a right bit shift and an offset data correction. For each extremes detector, the input channels to be considered into computing the extremes value are selected in EXCH[7:0] bits (in DFSDM_FLTxCR2 register).

29.4.13 Data unit block

The data unit block is the last block of the whole processing path: External ΣΔ modulators - Serial transceivers - Sinc filter - Integrator - Data unit block.

The output data rate depends on the serial data stream rate, and filter and integrator settings. The maximum output data rate is:

$$Datarate[samples/s] = \frac{f_{CKIN}}{F_{OSR} \cdot (I_{OSR} - 1 + F_{ORD}) + (F_{ORD} + 1)} \quad \text{...FAST = 0, Sincx filter}$$

$$Datarate[samples/s] = \frac{f_{CKIN}}{F_{OSR} \cdot (I_{OSR} - 1 + 4) + (2 + 1)} \quad \text{...FAST = 0, FastSinc filter}$$

or

$$Datarate[samples/s] = \frac{f_{CKIN}}{F_{OSR} \cdot I_{OSR}} \quad \text{...FAST = 1}$$

Maximum output data rate in case of parallel data input:

$$Datarate[samples/s] = \frac{f_{DATAIN_RATE}}{F_{OSR} \cdot (I_{OSR} - 1 + F_{ORD}) + (F_{ORD} + 1)} \quad \text{...FAST = 0, Sincx filter}$$

or

$$Datarate[samples/s] = \frac{f_{DATAIN_RATE}}{F_{OSR} \cdot (I_{OSR} - 1 + 4) + (2 + 1)} \quad \text{...FAST = 0, FastSinc filter}$$

or

$$Datarate[samples/s] = \frac{f_{DATAIN_RATE}}{F_{OSR} \cdot I_{OSR}} \quad \text{...FAST=1 or any filter bypass case } (F_{OSR} = 1)$$

where: f_{DATAIN_RATE}...input data rate from ADC or from CPU/DMA

The right bit-shift of final data is performed in this module because the final data width is 24-bit and data coming from the processing path can be up to 32 bits. This right bit-shift is configurable in the range 0-31 bits for each selected input channel (see DTRBS[4:0] bits in DFSDM_CHyCFGR2 register). The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained - to have valid 24-bit signed format of result data.

In the next step, an offset correction of the result is performed. The offset correction value (OFFSET[23:0] stored in register DFSDM_CHyCFGR2) is subtracted from the output data for a given channel. Data in the OFFSET[23:0] field is set by software by the appropriate calibration routine.

Due to the fact that all operations in digital processing are performed on 32-bit signed registers, the following conditions must be fulfilled not to overflow the result:

$FOSR^{FORD} \cdot IOSR <= 2^{31}$ ... for $Sinc^x$ filters, x = 1..5)
$2 \cdot FOSR^2 \cdot IOSR <= 2^{31}$ ... for FastSinc filter)

Note: In case of filter and integrator bypass (IOSR[7:0]=0, FOSR[9:0]=0), the input data rate (f_{DATAIN_RATE}) must be limited to be able to read all output data:
$f_{DATAIN_RATE} \leq f_{APB}$
where f_{APB} is the bus frequency to which the DFSDM peripheral is connected.

29.4.14 Signed data format

Each DFSDM input serial channel can be connected to one external ΣΔ modulator. An external ΣΔ modulator can have 2 differential inputs (positive and negative) which can be used for a differential or single-ended signal measurement.

A ΣΔ modulator output is always assumed in a signed format (a data stream of zeros and ones from a ΣΔ modulator represents values -1 and +1).

Signed data format in registers: Data is in a signed format in registers for final output data, analog watchdog, extremes detector, offset correction. The msb of output data word represents the sign of value (two's complement format).

29.4.15 Launching conversions

Injected conversions can be launched using the following methods:

- Software: writing '1' to JSWSTART in the DFSDM_FLTxCR1 register.
- Trigger: JEXTSEL[4:0] selects the trigger signal while JEXTEN activates the trigger and selects the active edge at the same time (see the DFSDM_FLTxCR1 register).
- Synchronous with DFSDM_FLT0 if JSYNC=1: for DFSDM_FLTx (x>0), an injected conversion is automatically launched when in DFSDM_FLT0; the injected conversion is started by software (JSWSTART=1 in DFSDM_FLT0CR2 register). Each injected conversion in DFSDM_FLTx (x>0) is always executed according to its local configuration settings (JSCAN, JCHG, etc.).

If the scan conversion is enabled (bit JSCAN=1) then, each time an injected conversion is triggered, all of the selected channels in the injected group (JCHG[7:0] bits in DFSDM_FLTxJCHGR register) are converted sequentially, starting with the lowest channel (channel 0, if selected).

If the scan conversion is disabled (bit JSCAN=0) then, each time an injected conversion is triggered, only one of the selected channels in the injected group (JCHG[7:0] bits in DFSDM_FLTxJCHGR register) is converted and the channel selection is then moved to the next selected channel. Writing to the JCHG[7:0] bits when JSCAN=0 sets the channel selection to the lowest selected injected channel.

Only one injected conversion can be ongoing at a given time. Thus, any request to launch an injected conversion is ignored if another request for an injected conversion has already been issued but not yet completed.

Regular conversions can be launched using the following methods:

- Software: by writing '1' to RSWSTART in the DFSDM_FLTxCR1 register.
- Synchronous with DFSDM_FLT0 if RSYNC=1: for DFSDM_FLTx (x>0), a regular conversion is automatically launched when in DFSDM_FLT0; a regular conversion is started by software (RSWSTART=1 in DFSDM_FLT0CR2 register). Each regular conversion in DFSDM_FLTx (x>0) is always executed according to its local configuration settings (RCONT, RCH, etc.).

Only one regular conversion can be pending or ongoing at a given time. Thus, any request to launch a regular conversion is ignored if another request for a regular conversion has already been issued but not yet completed. A regular conversion can be pending if it was interrupted by an injected conversion or if it was started while an injected conversion was in progress. This pending regular conversion is then delayed and is performed when all injected conversion are finished. Any delayed regular conversion is signalized by RPEND bit in DFSDM_FLTxRDATAR register.

29.4.16 Continuous and fast continuous modes

Setting RCONT in the DFSDM_FLTxCR1 register causes regular conversions to execute in continuous mode. RCONT=1 means that the channel selected by RCH[2:0] is converted repeatedly after '1' is written to RSWSTART.

The regular conversions executing in continuous mode can be stopped by writing ‘0’ to RCONT. After clearing RCONT, the on-going conversion is stopped immediately.

In continuous mode, the data rate can be increased by setting the FAST bit in the DFSDM_FLTxCR1 register. In this case, the filter does not need to be refilled by new fresh data if converting continuously from one channel because data inside the filter is valid from previously sampled continuous data. The speed increase depends on the chosen filter order. The first conversion in fast mode (FAST=1) after starting a continuous conversion by RSWSTART=1 takes still full time (as when FAST=0), then each subsequent conversion is finished in shorter intervals.

Conversion time in continuous mode:

if FAST = 0 (or first conversion if FAST=1):

for $Sinc^x$ filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}\text{-}1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$

for FastSinc filter:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}\text{-}1 + 4) + 2] / f_{CKIN}$

if FAST = 1 (except first conversion):

for $Sinc^x$ and FastSinc filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, only integrator active):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

Continuous mode is not available for injected conversions. Injected conversions can be started by timer trigger to emulate the continuous mode with precise timing.

If a regular continuous conversion is in progress (RCONT=1) and if a write access to DFSDM_FLTxCR1 register requesting regular continuous conversion (RCONT=1) is performed, then regular continuous conversion is restarted from the next conversion cycle (like new regular continuous conversion is applied for new channel selection - even if there is no change in DFSDM_FLTxCR1 register).

29.4.17 Request precedence

An injected conversion has a higher precedence than a regular conversion. A regular conversion which is already in progress is immediately interrupted by the request of an injected conversion; this regular conversion is restarted after the injected conversion finishes.

An injected conversion cannot be launched if another injected conversion is pending or already in progress: any request to launch an injected conversion (either by JSWSTART or by a trigger) is ignored as long as bit JCIP is ‘1’ (in the DFSDM_FLTxISR register).

Similarly, a regular conversion cannot be launched if another regular conversion is pending or already in progress: any request to launch a regular conversion (using RSWSTART) is ignored as long as bit RCIP is ‘1’ (in the DFSDM_FLTxISR register).

However, if an injected conversion is requested while a regular conversion is already in progress, the regular conversion is immediately stopped and an injected conversion is launched. The regular conversion is then restarted and this delayed restart is signalized in bit RPEND.

Injected conversions have precedence over regular conversions in that a injected conversion can temporarily interrupt a sequence of continuous regular conversions. When

the sequence of injected conversions finishes, the continuous regular conversions start again if RCONT is still set (and RPEND bit will signalize the delayed start on the first regular conversion result).

Precedence also matters when actions are initiated by the same write to DFSDM, or if multiple actions are pending at the end of another action. For example, suppose that, while an injected conversion is in process (JCIP=1), a single write operation to DFSDM_FLTxCR1 writes ‘1’ to RSWSTART, requesting a regular conversion. When the injected sequence finishes, the precedence dictates that the regular conversion is performed next and its delayed start is signalized in RPEND bit.

29.4.18 Power optimization in run mode

In order to reduce the consumption, the DFSDM filter and integrator are automatically put into idle when not used by conversions (RCIP=0, JCIP=0).

29.5 DFSDM interrupts

In order to increase the CPU performance, a set of interrupts related to the CPU event occurrence has been implemented:

- End of injected conversion interrupt:
  - enabled by JEOCIE bit in DFSDM_FLTxCR2 register
  - indicated in JEOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxJDATAR register (injected data)
  - indication of which channel end of conversion occurred, reported in JDATACH[2:0] bits in DFSDM_FLTxJDATAR register
- End of regular conversion interrupt:
  - enabled by REOCIE bit in DFSDM_FLTxCR2 register
  - indicated in REOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxRDATAR register (regular data)
  - indication of which channel end of conversion occurred, reported in RDATACH[2:0] bits in DFSDM_FLTxRDATAR register
- Data overrun interrupt for injected conversions:
  - occurred when injected converted data were not read from DFSDM_FLTxJDATAR register (by CPU or DMA) and were overwritten by a new injected conversion
  - enabled by JOVRIE bit in DFSDM_FLTxCR2 register
  - indicated in JOVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRJOVRF bit in DFSDM_FLTxICR register
- Data overrun interrupt for regular conversions:
  - occurred when regular converted data were not read from DFSDM_FLTxRDATAR register (by CPU or DMA) and were overwritten by a new regular conversion
  - enabled by ROVRIE bit in DFSDM_FLTxCR2 register
  - indicated in ROVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRROVRF bit in DFSDM_FLTxICR register
- Analog watchdog interrupt:

  - occurred when converted data (output data or data from analog watchdog filter - according to AWFSEL bit setting in DFSDM_FLTxCR1 register) crosses over/under high/low thresholds in DFSDM_FLTxAWHTR / DFSDM_FLTxAWLTR registers
  - enabled by AWDIE bit in DFSDM_FLTxCR2 register (on selected channels AWDCH[7:0])
  - indicated in AWDF bit in DFSDM_FLTxISR register
  - separate indication of high or low analog watchdog threshold error by AWHTF[7:0] and AWLTF[7:0] fields in DFSDM_FLTxAWSR register
  - cleared by writing ‘1’ into corresponding CLRAWHTF[7:0] or CLRAWLTF[7:0] bits in DFSDM_FLTxAWCFR register
- Short-circuit detector interrupt:
  - occurred when the number of stable data crosses over thresholds in DFSDM_CHyAWSCDR register
  - enabled by SCDIE bit in DFSDM_FLTxCR2 register (on channel selected by SCDEN bi tin DFSDM_CHyCFGR1 register)
  - indicated in SCDF[7:0] bits in DFSDM_FLTxISR register (which also reports the channel on which the short-circuit detector event occurred)
  - cleared by writing ‘1’ into the corresponding CLRSCDF[7:0] bit in DFSDM_FLTxICR register
- Channel clock absence interrupt:
  - occurred when there is clock absence on CKINy pin (see *Clock absence detection* in *Section 29.4.4: Serial channel transceivers*)
  - enabled by CKABIE bit in DFSDM_FLTxCR2 register (on channels selected by CKABEN bit in DFSDM_CHyCFGR1 register)
  - indicated in CKABF[y] bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRCKABF[y] bit in DFSDM_FLTxICR register

Table 245. DFSDM interrupt requests

Interrupt event	Event flag	Event/Interrupt clearing method	Interrupt enable control bit
End of injected conversion	JEOCF	reading DFSDM_FLTxJDATAR	JEOCIE
End of regular conversion	REOCF	reading DFSDM_FLTxRDATAR	REOCIE
Injected data overrun	JOVRF	writing CLRJOVRF = 1	JOVRIE
Regular data overrun	ROVRF	writing CLRROVRF = 1	ROVRIE
Analog watchdog	AWDF, AWHTF[7:0], AWLTF[7:0]	writing CLRAWHTF[7:0] = 1 writing CLRAWLTF[7:0] = 1	AWDIE, (AWDCH[7:0])
short-circuit detector	SCDF[7:0]	writing CLRSCDF[7:0] = 1	SCDIE, (SCDEN)
Channel clock absence	CKABF[7:0]	writing CLRCKABF[7:0] = 1	CKABIE, (CKABEN)

29.6 DFSDM DMA transfer

To decrease the CPU intervention, conversions can be transferred into memory using a DMA transfer. A DMA transfer for injected conversions is enabled by setting bit JDMAEN=1 in DFSDM_FLTxCR1 register. A DMA transfer for regular conversions is enabled by setting bit RDMAEN=1 in DFSDM_FLTxCR1 register.

Note: With a DMA transfer, the interrupt flag is automatically cleared at the end of the injected or regular conversion (JEOCF or REOCF bit in DFSDM_FLTxISR register) because DMA is reading DFSDM_FLTxJDATAR or DFSDM_FLTxRDATAR register.

29.7 DFSDM channel y registers (y=0..7)

29.7.1 DFSDM channel y configuration register (DFSDM_CHyCFGR1)

This register specifies the parameters used by channel y.

Address offset: 0x00 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFSDMEN	CKOUTSRC	Res.	Res.	Res.	Res.	Res.	Res.	CKOUTDIV[7:0]							
rw	rw							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
rw	rw	rw	rw				rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **DFSDMEN**: Global enable for DFSDM interface
0: DFSDM interface disabled
1: DFSDM interface enabled
If DFSDM interface is enabled, then it is started to operate according to enabled y channels and enabled x filters settings (CHEN bit in DFSDM_CHyCFGR1 and DFEN bit in DFSDM_FLTxCR1).
Data cleared by setting DFSDMEN=0:
–all registers DFSDM_FLTxISR are set to reset state (x = 0..3)
–all registers DFSDM_FLTxAWSR are set to reset state (x = 0..3)
Note: DFSDMEN is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bit 30 **CKOUTSRC**: Output serial clock source selection
0: Source for output clock is from system clock
1: Source for output clock is from audio clock
This value can be modified only when DFSDMEN=0 (in DFSDM_CH0CFGR1 register).
Note: CKOUTSRC is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **CKOUTDIV[7:0]**: Output serial clock divider

0: Output clock generation is disabled (CKOUT signal is set to low state)

1- 255: Defines the division of system clock for the serial clock output for CKOUT signal in range 2 - 256 (Divider = CKOUTDIV+1).

CKOUTDIV also defines the threshold for a clock absence detection.

This value can only be modified when DFSDMEN=0 (in DFSDM_CH0CFGR1 register).

If DFSDMEN=0 (in DFSDM_CH0CFGR1 register) then CKOUT signal is set to low state (setting is performed one DFSDM clock cycle after DFSDMEN=0).

Note: CKOUTDIV is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bits 15:14 **DATPACK[1:0]**: Data packing mode in DFSDM_CHyDATINR register.

0: Standard: input data in DFSDM_CHyDATINR register are stored only in INDAT0[15:0]. To empty DFSDM_CHyDATINR register one sample must be read by the DFSDM filter from channel y.

1: Interleaved: input data in DFSDM_CHyDATINR register are stored as two samples:

–first sample in INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y)

To empty DFSDM_CHyDATINR register, two samples must be read by the digital filter from channel y (INDAT0[15:0] part is read as first sample and then INDAT1[15:0] part is read as next sample).

2: Dual: input data in DFSDM_CHyDATINR register are stored as two samples:

–first sample INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y+1)

To empty DFSDM_CHyDATINR register first sample must be read by the digital filter from channel y and second sample must be read by another digital filter from channel y+1. Dual mode is available only on even channel numbers (y = 0, 2, 4, 6), for odd channel numbers (y = 1, 3, 5, 7) DFSDM_CHyDATINR is write protected. If an even channel is set to dual mode then the following odd channel must be set into standard mode (DATPACK[1:0]=0) for correct cooperation with even channel.

3: Reserved

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 13:12 **DATMPX[1:0]**: Input data multiplexer for channel y

0: Data to channel y are taken from external serial inputs as 1-bit values. DFSDM_CHyDATINR register is write protected.

1: Data to channel y are taken from internal analog to digital converter ADC_{y+1} output register update as 16-bit values (if ADC_{y+1} is available). Data from ADCs are written into INDAT0[15:0] part of DFSDM_CHyDATINR register.

2: Data to channel y are taken from internal DFSDM_CHyDATINR register by direct CPU/DMA write. There can be written one or two 16-bit data samples according DATPACK[1:0] bit field setting.

3: Reserved

Note: This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Note: DATMPX[1:0] = 1 is supported only by ADC1 and ADC2.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **CHINSEL**: Channel inputs selection

0: Channel inputs are taken from pins of the same channel y.

1: Channel inputs are taken from pins of the following channel (channel (y+1) modulo 8).

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 7 **CHEN**: Channel y enable

0: Channel y disabled

1: Channel y enabled

If channel y is enabled, then serial data receiving is started according to the given channel setting.

Bit 6 **CKABEN**: Clock absence detector enable on channel y
- 0: Clock absence detector disabled on channel y
- 1: Clock absence detector enabled on channel y

Bit 5 **SCDEN**: Short-circuit detector enable on channel y
- 0: Input channel y will not be guarded by the short-circuit detector
- 1: Input channel y will be continuously guarded by the short-circuit detector

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **SPICKSEL[1:0]**: SPI clock select for channel y
- 0: clock coming from external CKINy input - sampling point according SITP[1:0]
- 1: clock coming from internal CKOUT output - sampling point according SITP[1:0]
- 2: clock coming from internal CKOUT - sampling point on each second CKOUT falling edge. For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
- 3: clock coming from internal CKOUT output - sampling point on each second CKOUT rising edge. For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 1:0 **SITP[1:0]**: Serial interface type for channel y
- 00: SPI with rising edge to strobe data
- 01: SPI with falling edge to strobe data
- 10: Manchester coded input on DATINy pin: rising edge = logic 0, falling edge = logic 1
- 11: Manchester coded input on DATINy pin: rising edge = logic 1, falling edge = logic 0

This value can only be modified when CHEN=0 (in DFSDM_CHyCFGR1 register).

29.7.2 DFSDM channel y configuration register (DFSDM_CHyCFGR2)

This register specifies the parameters used by channel y.

Address offset: 0x04 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[7:0]								DTRBS[4:0]					Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:8 **OFFSET[23:0]**: 24-bit calibration offset for channel y
For channel y, OFFSET is applied to the results of each conversion from this channel.
This value is set by software.

Bits 7:3 **DTRBS[4:0]**: Data right bit-shift for channel y
0-31: Defines the shift of the data result coming from the integrator - how many bit shifts to the right will be performed to have final results. Bit-shift is performed before offset correction. The data shift is rounding the result to nearest integer value. The sign of shifted result is maintained (to have valid 24-bit signed format of result data).
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 2:0 Reserved, must be kept at reset value.

29.7.3 DFSDM channel y analog watchdog and short-circuit detector register (DFSDM_CHyAWSCDR)

Short-circuit detector and analog watchdog settings for channel y.

Address offset: 0x08 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]				
								rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **AWFORD[1:0]**: Analog watchdog Sinc filter order on channel y
0: FastSinc filter type
1: $Sinc^1$ filter type
2: $Sinc^2$ filter type
3: $Sinc^3$ filter type
$Sinc^x$ filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^x$$

FastSinc filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **AWFOSR[4:0]**: Analog watchdog filter oversampling ratio (decimation rate) on channel y
0 - 31: Defines the length of the Sinc type filter in the range 1 - 32 (AWFOSR + 1). This number is also the decimation ratio of the analog data rate.
This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).
Note: If AWFOSR = 0 then the filter has no effect (filter bypass).

Bits 15:12 **BKSCD[3:0]**: Break signal assignment for short-circuit detector on channel y
BKSCD[i] = 0: Break i signal not assigned to short-circuit detector on channel y
BKSCD[i] = 1: Break i signal assigned to short-circuit detector on channel y

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:0 **SCDT[7:0]**: short-circuit detector threshold for channel y
These bits are written by software to define the threshold counter for the short-circuit detector. If this value is reached, then a short-circuit detector event occurs on a given channel.

29.7.4 DFSDM channel y watchdog filter data register (DFSDM_CHyWDATR)

This register contains the data resulting from the analog watchdog filter associated to the input channel y.

Address offset: 0x0C + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WDATA[15:0]**: Input channel y watchdog data
Data converted by the analog watchdog filter for input channel y. This data is continuously converted (no trigger) for this channel, with a limited resolution (OSR=1..32/sinc order = 1..3).

29.7.5 DFSDM channel y data input register (DFSDM_CHyDATINR)

This register contains 16-bit input data to be processed by DFSDM filter module.

Address offset: 0x10 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INDAT1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INDAT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INDAT1[15:0]**: Input data for channel y or channel y+1

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2.
Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).
If DATPACK[1:0]=0 (standard mode)
INDAT0[15:0] is write protected (not used for input sample).
If DATPACK[1:0]=1 (interleaved mode)
Second channel y data sample is stored into INDAT1[15:0]. First channel y data sample is stored into INDAT0[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.
If DATPACK[1:0]=2 (dual mode).
For even y channels: sample in INDAT1[15:0] is automatically copied into INDAT0[15:0] of channel (y+1).
For odd y channels: INDAT1[15:0] is write protected.
See *Section 29.4.6: Parallel data inputs* for more details.
INDAT0[15:1] is in the16-bit signed format.

Bits 15:0 **INDAT0[15:0]**: Input data for channel y

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2.
Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).
If DATPACK[1:0]=0 (standard mode)
Channel y data sample is stored into INDAT0[15:0].
If DATPACK[1:0]=1 (interleaved mode)
First channel y data sample is stored into INDAT0[15:0]. Second channel y data sample is stored into INDAT1[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.
If DATPACK[1:0]=2 (dual mode).
For even y channels: Channel y data sample is stored into INDAT0[15:0].
For odd y channels: INDAT0[15:0] is write protected.
See *Section 29.4.6: Parallel data inputs* for more details.
INDAT0[15:0] is in the16-bit signed format.

29.8 DFSDM filter x module registers (x=0..3)

29.8.1 DFSDM filter x control register 1 (DFSDM_FLTxCR1)

Address offset: 0x100 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSWSTART	Res.
	rw	rw			rw	rw	rw			rw		rw	rw	rt_w1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSWSTART	DFEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw		rt_w1	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **AWFSEL**: Analog watchdog fast mode select

0: Analog watchdog on data output value (after the digital filter). The comparison is done after offset correction and shift

1: Analog watchdog on channel transceivers value (after watchdog filter)

Bit 29 **FAST**: Fast conversion mode selection for regular conversions

0: Fast conversion mode disabled

1: Fast conversion mode enabled

When converting a regular conversion in continuous mode, having enabled the fast mode causes each conversion (except the first) to execute faster than in standard mode. This bit has no effect on conversions which are not continuous.

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

if FAST=0 (or first conversion in continuous mode if FAST=1):

$t = [F_{OSR} * (I_{OSR}\text{-}1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$..... for $Sinc^x$ filters

$t = [F_{OSR} * (I_{OSR}\text{-}1 + 4) + 2] / f_{CKIN}$..... for FastSinc filter

if FAST=1 in continuous mode (except first conversion):

$t = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if $F_{OSR} = F_{OSR}[9:0]+1 = 1$ (filter bypassed, active only integrator):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

where: f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input.

Bits 28:27 Reserved, must be kept at reset value.

Bits 26:24 **RCH[2:0]**: Regular channel selection

0: Channel 0 is selected as the regular channel

1: Channel 1 is selected as the regular channel

...

7: Channel 7 is selected as the regular channel

Writing these bits when RCIP=1 takes effect when the next regular conversion begins. This is especially useful in continuous mode (when RCONT=1). It also affects regular conversions which are pending (due to ongoing injected conversion).

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RDMAEN**: DMA channel enabled to read data for the regular conversion

0: The DMA channel is not enabled to read regular data

1: The DMA channel is enabled to read regular data

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RSYNC**: Launch regular conversion synchronously with DFSDM_FLT0

0: Do not launch a regular conversion synchronously with DFSDM_FLT0

1: Launch a regular conversion in this DFSDM_FLTx at the very moment when a regular conversion is launched in DFSDM_FLT0

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 18 **RCONT**: Continuous mode selection for regular conversions

0: The regular channel is converted just once for each conversion request

1: The regular channel is converted repeatedly after each conversion request

Writing ‘0’ to this bit while a continuous regular conversion is already in progress stops the continuous mode immediately.

Bit 17 **RSWSTART**: Software start of a conversion on the regular channel
0: Writing ‘0’ has no effect
1: Writing ‘1’ makes a request to start a conversion on the regular channel and causes RCIP to become ‘1’. If RCIP=1 already, writing to RSWSTART has no effect. Writing ‘1’ has no effect if RSYNC=1.
This bit is always read as ‘0’.

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:13 **JEXTEN[1:0]**: Trigger enable and trigger edge selection for injected conversions
00: Trigger detection is disabled
01: Each rising edge on the selected trigger makes a request to launch an injected conversion
10: Each falling edge on the selected trigger makes a request to launch an injected conversion
11: Both rising edges and falling edges on the selected trigger make requests to launch injected conversions
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bits 12:8 **JEXTSEL[4:0]**: Trigger signal selection for launching injected conversions
0x0-0x1F: Trigger inputs selected by the following table (internal or external trigger).
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Note: synchronous trigger has latency up to one $f_{DFSDMCLK}$ clock cycle (with deterministic jitter), asynchronous trigger has latency 2-3 $f_{DFSDMCLK}$ clock cycles (with jitter up to 1 cycle).

	DFSDM_FLTx
0x00	dfsdm_jtrg0
0x01	dfsdm_jtrg1
...	
0x1E	dfsdm_jtrg30
0x1F	dfsdm_jtrg31

Refer to *Table 241: DFSDM triggers connection*.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JDMAEN**: DMA channel enabled to read data for the injected channel group
0: The DMA channel is not enabled to read injected data
1: The DMA channel is enabled to read injected data
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 4 **JSCAN**: Scanning conversion mode for injected conversions
0: One channel conversion is performed from the injected channel group and next the selected channel from this group is selected.
1: The series of conversions for the injected group channels is executed, starting over with the lowest selected channel.
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).
Writing JCHG if JSCAN=0 resets the channel selection to the lowest selected channel.

Bit 3 **JSYNC**: Launch an injected conversion synchronously with the DFSDM_FLT0 JSWSTART trigger
0: Do not launch an injected conversion synchronously with DFSDM_FLT0
1: Launch an injected conversion in this DFSDM_FLTx at the very moment when an injected conversion is launched in DFSDM_FLT0 by its JSWSTART trigger
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **JSWSTART**: Start a conversion of the injected group of channels
0: Writing ‘0’ has no effect.
1: Writing ‘1’ makes a request to convert the channels in the injected conversion group, causing JCIP to become ‘1’ at the same time. If JCIP=1 already, then writing to JSWSTART has no effect. Writing ‘1’ has no effect if JSYNC=1.
This bit is always read as ‘0’.

Bit 0 **DFEN**: DFSDM_FLTx enable
0: DFSDM_FLTx is disabled. All conversions of given DFSDM_FLTx are stopped immediately and all DFSDM_FLTx functions are stopped.
1: DFSDM_FLTx is enabled. If DFSDM_FLTx is enabled, then DFSDM_FLTx starts operating according to its setting.
Data which are cleared by setting DFEN=0:
–register DFSDM_FLTxISR is set to the reset state
–register DFSDM_FLTxAWSR is set to the reset state

29.8.2 DFSDM filter x control register 2 (DFSDM_FLTxCR2)

Address offset: 0x104 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXCH[7:0]								Res.	CKABIE	SCDIE	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **AWDCH[7:0]**: Analog watchdog channel selection
These bits select the input channel to be guarded continuously by the analog watchdog.
AWDCH[y] = 0: Analog watchdog is disabled on channel y
AWDCH[y] = 1: Analog watchdog is enabled on channel y

Bits 15:8 **EXCH[7:0]**: Extremes detector channel selection
These bits select the input channels to be taken by the Extremes detector.
EXCH[y] = 0: Extremes detector does not accept data from channel y
EXCH[y] = 1: Extremes detector accepts data from channel y

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CKABIE**: Clock absence interrupt enable
0: Detection of channel input clock absence interrupt is disabled
1: Detection of channel input clock absence interrupt is enabled
Please see the explanation of CKABF[7:0] in DFSDM_FLTxISR.
Note: CKABIE is present only in DFSDM_FLT0CR2 register (filter x=0)

Bit 5 **SCDIE**: Short-circuit detector interrupt enable
0: short-circuit detector interrupt is disabled
1: short-circuit detector interrupt is enabled
Please see the explanation of SCDF[7:0] in DFSDM_FLTxISR.
Note: SCDIE is present only in DFSDM_FLT0CR2 register (filter x=0)

Bit 4 **AWDIE**: Analog watchdog interrupt enable
0: Analog watchdog interrupt is disabled
1: Analog watchdog interrupt is enabled
Please see the explanation of AWDF in DFSDM_FLTxISR.

Bit 3 **ROVRIE**: Regular data overrun interrupt enable
0: Regular data overrun interrupt is disabled
1: Regular data overrun interrupt is enabled
Please see the explanation of ROVRF in DFSDM_FLTxISR.

Bit 2 **JOVRIE**: Injected data overrun interrupt enable
0: Injected data overrun interrupt is disabled
1: Injected data overrun interrupt is enabled
Please see the explanation of JOVRF in DFSDM_FLTxISR.

Bit 1 **REOCIE**: Regular end of conversion interrupt enable
0: Regular end of conversion interrupt is disabled
1: Regular end of conversion interrupt is enabled
Please see the explanation of REOCF in DFSDM_FLTxISR.

Bit 0 **JEOCIE**: Injected end of conversion interrupt enable
0: Injected end of conversion interrupt is disabled
1: Injected end of conversion interrupt is enabled
Please see the explanation of JEOCF in DFSDM_FLTxISR.

29.8.3 DFSDM filter x interrupt and status register (DFSDM_FLTxISR)

Address offset: 0x108 + 0x80 * x, (x = 0 to 3)

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCDF[7:0]								CKABF[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	r	r									r	r	r	r	r

Bits 31:24 **SCDF[7:0]**: short-circuit detector flag

SDCF[y]=0: No short-circuit detector event occurred on channel y

SDCF[y]=1: The short-circuit detector counter reaches, on channel y, the value programmed in the DFSDM_CHyAWSCDR registers

This bit is set by hardware. It can be cleared by software using the corresponding CLRSCDF[y] bit in the DFSDM_FLTxICR register. SCDF[y] is cleared also by hardware when CHEN[y] = 0 (given channel is disabled).

Note: SCDF[7:0] is present only in DFSDM_FLT0ISR register (filter x=0)

Bits 23:16 **CKABF[7:0]**: Clock absence flag

CKABF[y]=0: Clock signal on channel y is present.

CKABF[y]=1: Clock signal on channel y is not present.

Given y bit is set by hardware when clock absence is detected on channel y. It is held at CKABF[y]=1 state by hardware when CHEN=0 (see DFSDM_CHyCFGR1 register). It is held at CKABF[y]=1 state by hardware when the transceiver is not yet synchronized.It can be cleared by software using the corresponding CLRCKABF[y] bit in the DFSDM_FLTxICR register.

Note: CKABF[7:0] is present only in DFSDM_FLT0ISR register (filter x=0)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RCIP**: Regular conversion in progress status

0: No request to convert the regular channel has been issued

1: The conversion of the regular channel is in progress or a request for a regular conversion is pending

A request to start a regular conversion is ignored when RCIP=1.

Bit 13 **JCIP**: Injected conversion in progress status

0: No request to convert the injected channel group (neither by software nor by trigger) has been issued

1: The conversion of the injected channel group is in progress or a request for a injected conversion is pending, due either to ‘1’ being written to JSWSTART or to a trigger detection

A request to start an injected conversion is ignored when JCIP=1.

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **AWDF**: Analog watchdog

0: No Analog watchdog event occurred

1: The analog watchdog block detected voltage which crosses the value programmed in the DFSDM_FLTxAWLTR or DFSDM_FLTxAWHTR registers.

This bit is set by hardware. It is cleared by software by clearing all source flag bits AWHTF[7:0] and AWLTF[7:0] in DFSDM_FLTxAWSR register (by writing ‘1’ into the clear bits in DFSDM_FLTxAWCFR register).

Bit 3 **ROVRF**: Regular conversion overrun flag

0: No regular conversion overrun has occurred

1: A regular conversion overrun has occurred, which means that a regular conversion finished while REOCF was already ‘1’. RDATAR is not affected by overruns

This bit is set by hardware. It can be cleared by software using the CLRROVRF bit in the DFSDM_FLTxICR register.

Bit 2 **JOVRF**: Injected conversion overrun flag
0: No injected conversion overrun has occurred
1: An injected conversion overrun has occurred, which means that an injected conversion finished while JEOCF was already ‘1’. JDATAR is not affected by overruns
This bit is set by hardware. It can be cleared by software using the CLRJOVRF bit in the DFSDM_FLTxICR register.

Bit 1 **REOCF**: End of regular conversion flag
0: No regular conversion has completed
1: A regular conversion has completed and its data may be read
This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxRDATAR.

Bit 0 **JEOCF**: End of injected conversion flag
0: No injected conversion has completed
1: An injected conversion has completed and its data may be read
This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxJDATAR.

Note: For each of the flag bits, an interrupt can be enabled by setting the corresponding bit in DFSDM_FLTxCR2. If an interrupt is called, the flag must be cleared before exiting the interrupt service routine.

All the bits of DFSDM_FLTxISR are automatically reset when DFEN=0.

29.8.4 DFSDM filter x interrupt flag clear register (DFSDM_FLTxICR)

Address offset: 0x10C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRSCDF[7:0]								CLRCKABF[7:0]							
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRROVRF	CLRJOVRF	Res.	Res.
												rc_w1	rc_w1		

Bits 31:24 **CLRSCDF[7:0]**: Clear the short-circuit detector flag
CLRSCDF[y]=0: Writing ‘0’ has no effect
CLRSCDF[y]=1: Writing ‘1’ to position y clears the corresponding SCDF[y] bit in the DFSDM_FLTxISR register
Note: CLRSCDF[7:0] is present only in DFSDM_FLT0ICR register (filter x=0)

Bits 23:16 **CLRCKABF[7:0]**: Clear the clock absence flag
CLRCKABF[y]=0: Writing ‘0’ has no effect
CLRCKABF[y]=1: Writing ‘1’ to position y clears the corresponding CKABF[y] bit in the DFSDM_FLTxISR register. When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y].
Note: CLRCKABF[7:0] is present only in DFSDM_FLT0ICR register (filter x=0)

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CLRROVRF**: Clear the regular conversion overrun flag
0: Writing ‘0’ has no effect
1: Writing ‘1’ clears the ROVRF bit in the DFSDM_FLTxISR register

Bit 2 **CLRJOVRF**: Clear the injected conversion overrun flag
0: Writing ‘0’ has no effect
1: Writing ‘1’ clears the JOVRF bit in the DFSDM_FLTxISR register

Bits 1:0 Reserved, must be kept at reset value.

Note: The bits of DFSDM_FLTxICR are always read as ‘0’.

29.8.5 DFSDM filter x injected channel group selection register (DFSDM_FLTxJCHGR)

Address offset: 0x110 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **JCHG[7:0]**: Injected channel group selection
JCHG[y]=0: channel y is not part of the injected group
JCHG[y]=1: channel y is part of the injected group
If JSCAN=1, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first and the sequence ends at the highest selected channel.
If JSCAN=0, then only one channel is converted from the selected channels, and the channel selection is moved to the next channel. Writing JCHG, if JSCAN=0, resets the channel selection to the lowest selected channel.
At least one channel must always be selected for the injected group. Writes causing all JCHG bits to be zero are ignored.

29.8.6 DFSDM filter x control register (DFSDM_FLTxFCR)

Address offset: 0x114 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]									
rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **FORD[2:0]**: Sinc filter order

0: FastSinc filter type
1: $Sinc^1$ filter type
2: $Sinc^2$ filter type
3: $Sinc^3$ filter type
4: $Sinc^4$ filter type
5: $Sinc^5$ filter type
6-7: Reserved

$Sinc^x$ filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^x$$

FastSinc filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}} \right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1).

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:16 **FOSR[9:0]**: Sinc filter oversampling ratio (decimation rate)

0 - 1023: Defines the length of the Sinc type filter in the range 1 - 1024 (F_{OSR} = FOSR[9:0] +1). This number is also the decimation ratio of the output data rate from filter.
This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1)

Note: If FOSR = 0, then the filter has no effect (filter bypass).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IOSR[7:0]**: Integrator oversampling ratio (averaging length)

0- 255: The length of the Integrator in the range 1 - 256 (IOSR + 1). Defines how many samples from Sinc filter will be summed into one output data sample from the integrator. The output data rate from the integrator will be decreased by this number (additional data decimation ratio).
This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1)

Note: If IOSR = 0, then the Integrator has no effect (Integrator bypass).

29.8.7 DFSDM filter x data register for injected group (DFSDM_FLTxJDATAR)

Address offset: 0x118 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[7:0]								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:8 **JDATA[23:0]**: Injected group conversion data
When each conversion of a channel in the injected group finishes, its resulting data is stored in this field. The data is valid when JEOCF=1. Reading this register clears the corresponding JEOCF.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **JDATACH[2:0]**: Injected channel most recently converted
When each conversion of a channel in the injected group finishes, JDATACH[2:0] is updated to indicate which channel was converted. Thus, JDATA[23:0] holds the data that corresponds to the channel indicated by JDATACH[2:0].

Note: DMA may be used to read the data from this register. Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears JEOCF in DFSDM_FLTxISR. Thus, the firmware must not read this register if DMA is activated to read data from this register.

29.8.8 DFSDM filter x data register for the regular channel (DFSDM_FLTxRDATAR)

Address offset: 0x11C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[7:0]								Res.	Res.	Res.	RPEND	Res.	RDATACH[2:0]		
r	r	r	r	r	r	r	r				r		r	r	r

Bits 31:8 **RDATA[23:0]**: Regular channel conversion data
When each regular conversion finishes, its data is stored in this register. The data is valid when REOCF=1. Reading this register clears the corresponding REOCF.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **RPEND**: Regular channel pending data
Regular data in RDATA[23:0] was delayed due to an injected channel trigger during the conversion

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **RDATACH[2:0]**: Regular channel most recently converted
When each regular conversion finishes, RDATACH[2:0] is updated to indicate which channel was converted (because regular channel selection RCH[2:0] in DFSDM_FLTxCR1 register can be updated during regular conversion). Thus RDATA[23:0] holds the data that corresponds to the channel indicated by RDATACH[2:0].

Note: Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears REOCF in DFSDM_FLTxISR.

29.8.9 DFSDM filter x analog watchdog high threshold register (DFSDM_FLTxAWHTR)

Address offset: 0x120 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWHT[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWHT[7:0]								Res.	Res.	Res.	Res.	BKAWH[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:8 **AWHT[23:0]**: Analog watchdog high threshold

These bits are written by software to define the high threshold for the analog watchdog.

Note: In case channel transceivers monitor (AWFSEL=1), the higher 16 bits (AWHT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWHT[7:0] are not taken into comparison in this case.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **BKAWH[3:0]**: Break signal assignment to analog watchdog high threshold event

BKAWH[i] = 0: Break i signal is not assigned to an analog watchdog high threshold event

BKAWH[i] = 1: Break i signal is assigned to an analog watchdog high threshold event

29.8.10 DFSDM filter x analog watchdog low threshold register (DFSDM_FLTxAWLTR)

Address offset: 0x124 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWLT[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWLT[7:0]								Res.	Res.	Res.	Res.	BKAWL[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:8 **AWLT[23:0]**: Analog watchdog low threshold

These bits are written by software to define the low threshold for the analog watchdog.

Note: In case channel transceivers monitor (AWFSEL=1), only the higher 16 bits (AWLT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWLT[7:0] are not taken into comparison in this case.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **BKAWL[3:0]**: Break signal assignment to analog watchdog low threshold event

BKAWL[i] = 0: Break i signal is not assigned to an analog watchdog low threshold event

BKAWL[i] = 1: Break i signal is assigned to an analog watchdog low threshold event

29.8.11 DFSDM filter x analog watchdog status register (DFSDM_FLTxAWSR)

Address offset: 0x128 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWHTF[7:0]								AWLTF[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **AWHTF[7:0]**: Analog watchdog high threshold flag

AWHTF[y]=1 indicates a high threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWHTF[y] bit in the DFSDM_FLTxAWCFR register.

Bits 7:0 **AWLTF[7:0]**: Analog watchdog low threshold flag

AWLTF[y]=1 indicates a low threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWLTF[y] bit in the DFSDM_FLTxAWCFR register.

Note: All the bits of DFSDM_FLTxAWSR are automatically reset when DFEN=0.

29.8.12 DFSDM filter x analog watchdog clear flag register (DFSDM_FLTxAWCFR)

Address offset: 0x12C + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRAWHTF[7:0]								CLRAWLTF[7:0]							
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **CLRAWHTF[7:0]**: Clear the analog watchdog high threshold flag

CLRAWHTF[y]=0: Writing ‘0’ has no effect

CLRAWHTF[y]=1: Writing ‘1’ to position y clears the corresponding AWHTF[y] bit in the DFSDM_FLTxAWSR register

Bits 7:0 **CLRAWLTF[7:0]**: Clear the analog watchdog low threshold flag

CLRAWLTF[y]=0: Writing ‘0’ has no effect

CLRAWLTF[y]=1: Writing ‘1’ to position y clears the corresponding AWLTF[y] bit in the DFSDM_FLTxAWSR register

29.8.13 DFSDM filter x extremes detector maximum register (DFSDM_FLTxEXMAX)

Address offset: 0x130 + 0x80 * x, (x = 0 to 3)

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXMAX[23:8]															
rs_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXMAX[7:0]								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r						r	r	r

Bits 31:8 **EXMAX[23:0]**: Extremes detector maximum value

These bits are set by hardware and indicate the highest value converted by DFSDM_FLTx. EXMAX[23:0] bits are reset to value (0x800000) by reading of this register.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EXMAXCH[2:0]**: Extremes detector maximum data channel.

These bits contains information about the channel on which the data is stored into EXMAX[23:0]. Bits are cleared by reading of this register.

29.8.14 DFSDM filter x extremes detector minimum register (DFSDM_FLTxEXMIN)

Address offset: 0x134 + 0x80 * x, (x = 0 to 3)

Reset value: 0x7FFF FF00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXMIN[23:8]															
rc_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXMIN[7:0]								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r						r	r	r

Bits 31:8 **EXMIN[23:0]**: Extremes detector minimum value

These bits are set by hardware and indicate the lowest value converted by DFSDM_FLTx. EXMIN[23:0] bits are reset to value (0x7FFFFF) by reading of this register.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EXMINCH[2:0]**: Extremes detector minimum data channel

These bits contain information about the channel on which the data is stored into EXMIN[23:0]. Bits are cleared by reading of this register.

29.8.15 DFSDM filter x conversion timer register (DFSDM_FLTxCNVTIMR)

Address offset: 0x138 + 0x80 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNVCNT[27:12]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNVCNT[11:0]												Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r				

Bits 31:4 **CNVCNT[27:0]**: 28-bit timer counting conversion time t = CNVCNT[27:0] / $f_{DFSDMCLK}$

The timer has an input clock from DFSDM clock (system clock $f_{DFSDMCLK}$). Conversion time measurement is started on each conversion start and stopped when conversion finishes (interval between first and last serial sample). Only in case of filter bypass (FOSR[9:0] = 0) is the conversion time measurement stopped and CNVCNT[27:0] = 0. The counted time is:

if FAST=0 (or first conversion in continuous mode if FAST=1):

$t = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$..... for $Sinc^x$ filters

$t = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$..... for FastSinc filter

if FAST=1 in continuous mode (except first conversion):

$t = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):

CNVCNT = 0 (counting is stopped, conversion time: $t = I_{OSR} / f_{CKIN}$)

where: f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input (from internal ADC or from CPU/DMA write)

Note: When conversion is interrupted (e.g. by disable/enable selected channel) the timer counts also this interruption time.

Bits 3:0 Reserved, must be kept at reset value.

29.8.16 DFSDM register map

The following table summarizes the DFSDM registers.

Table 246. DFSDM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**DFSDM_CH0CFGR1**	DFSDMEN	CKOUTSRC	Res.	Res.	Res.	Res.	Res.	Res.	CKOUTDIV[7:0]								DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value	0	0							0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0		0	0	0	0
0x04	**DFSDM_CH0CFGR2**	OFFSET[23:0]																								DTRBS[4:0]						Res.	Res.	Res.
	reset value	0																								0							
0x08	**DFSDM_CH0AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x0C	**DFSDM_CH0WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**DFSDM_CH0DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14 - 0x1C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x20	**DFSDM_CH1CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**DFSDM_CH1CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x28	**DFSDM_CH1AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x2C	**DFSDM_CH1WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**DFSDM_CH1DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34 - 0x3C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x40	**DFSDM_CH2CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x44	**DFSDM_CH2CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x48	**DFSDM_CH2AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x4C	**DFSDM_CH2WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**DFSDM_CH2DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54 - 0x5C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x60	**DFSDM_CH3CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x64	**DFSDM_CH3CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x68	**DFSDM_ CH3AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x6C	**DFSDM_ CH3WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	**DFSDM_ CH3DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74 - 0x7C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x80	**DFSDM_ CH4CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x84	**DFSDM_ CH4CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x88	**DFSDM_ CH4AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x8C	**DFSDM_ CH4WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	**DFSDM_ CH4DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94 - 0x9C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xA0	**DFSDM_ CH5CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0xA4	**DFSDM_ CH5CFGR2**	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xA8	**DFSDM_ CH5AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xAC	**DFSDM_CH5WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB0	**DFSDM_CH5DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4 - 0xBC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xC0	**DFSDM_CH6CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0xC4	**DFSDM_CH6CFGR2**	OFFSET[23:0]																								DTRBS[4:0]						Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xC8	**DFSDM_CH6AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value								0	0		0	0	0	0	0		0	0	0	0					0	0	0	0	0	0	0	0
0xCC	**DFSDM_CH6WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0	**DFSDM_CH6DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD4 - 0xDC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xE0	**DFSDM_CH7CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0xE4	**DFSDM_CH7CFGR2**	OFFSET[23:0]																								DTRBS[4:0]						Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xE8	**DFSDM_CH7AWSCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value								0	0		0	0	0	0	0		0	0	0	0					0	0	0	0	0	0	0	0
0xEC	**DFSDM_CH7WDATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF0	**DFSDM_CH7DATINR**	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF4 - 0xFC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	**DFSDM_FLT0CR1**	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x104	**DFSDM_FLT0CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	CKABIE	SCDIE	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x108	**DFSDM_FLT0ISR**	SCDF[7:0]								CKABF[7:0]								Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		0	0									0	0	0	0	0
0x10C	**DFSDM_FLT0ICR**	CLRSCDF[7:0]								CLRCKABF[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													0	0		
0x110	**DFSDM_FLT0JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x114	**DFSDM_FLT0FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x118	**DFSDM_FLT0JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH [2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x11C	**DFSDM_FLT0RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x120	**DFSDM_FLT0AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x124	**DFSDM_FLT0AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x128	**DFSDM_FLT0AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C	**DFSDM_FLT0AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x130	**DFSDM_FLT0EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x134	**DFSDM_FLT0EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x138	**DFSDM_FLT0CNVTIMR**	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x13C - 0x17C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x180	**DFSDM_FLT1CR1**	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x184	**DFSDM_FLT1CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x188	**DFSDM_FLT1ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x18C	**DFSDM_FLT1ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x190	**DFSDM_FLT1JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x194	**DFSDM_FLT1FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x198	**DFSDM_FLT1JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x19C	**DFSDM_ FLT1RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x1A0	**DFSDM_ FLT1AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x1A4	**DFSDM_ FLT1AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x1A8	**DFSDM_ FLT1AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1AC	**DFSDM_ FLT1AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1B0	**DFSDM_ FLT1EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x1B4	**DFSDM_ FLT1EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x1B8	**DFSDM_ FLT1CNVTIMR**	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x1BC - 0x1FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x200	**DFSDM_ FLT2CR1**	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x204	**DFSDM_ FLT2CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x208	**DFSDM_ FLT2ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x20C	**DFSDM_ FLT2ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x210	**DFSDM_FLT2JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x214	**DFSDM_FLT2FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x218	**DFSDM_FLT2JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x21C	**DFSDM_FLT2RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x220	**DFSDM_FLT2AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x224	**DFSDM_FLT2AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x228	**DFSDM_FLT2AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x22C	**DFSDM_FLT2AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x230	**DFSDM_FLT2EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x234	**DFSDM_FLT2EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x238	**DFSDM_FLT2CNVTIMR**	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x23C - 0x27C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x280	**DFSDM_FLT3CR1**	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x284	**DFSDM_ FLT3CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x288	**DFSDM_ FLT3ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x28C	**DFSDM_ FLT3ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x290	**DFSDM_ FLT3JCHGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x294	**DFSDM_ FLT3FCR**	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x298	**DFSDM_ FLT3JDATAR**	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x29C	**DFSDM_ FLT3RDATAR**	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x2A0	**DFSDM_ FLT3AWHTR**	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x2A4	**DFSDM_ FLT3AWLTR**	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x2A8	**DFSDM_ FLT3AWSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2AC	**DFSDM_ FLT3AWCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2B0	**DFSDM_ FLT3EXMAX**	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0

Table 246. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2B4	**DFSDM_ FLT3EXMIN**	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x2B8	**DFSDM_ FLT3CNVTIMR**	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x2BC - 0x3FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

30 Digital camera interface (DCMI)

30.1 DCMI introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

This interface is for use with black & white cameras, X24 and X5 cameras, and it is assumed that all preprocessing like resizing is performed in the camera module.

30.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

30.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

30.4 DCMI functional overview

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (D13-D0) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

30.4.1 DCMI block diagram

Figure 231 shows the DCMI block diagram.

Figure 231. DCMI block diagram

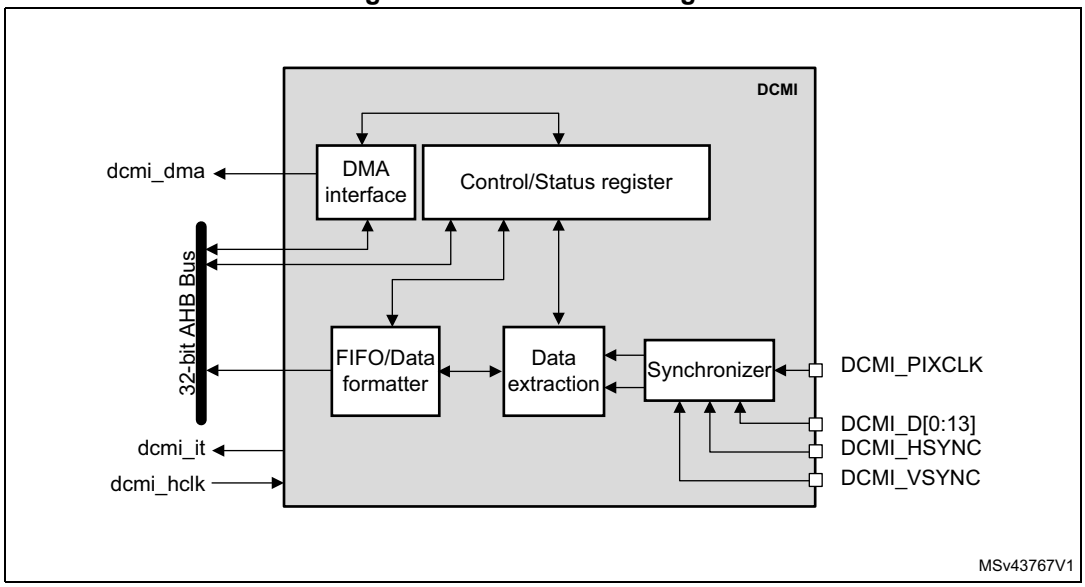
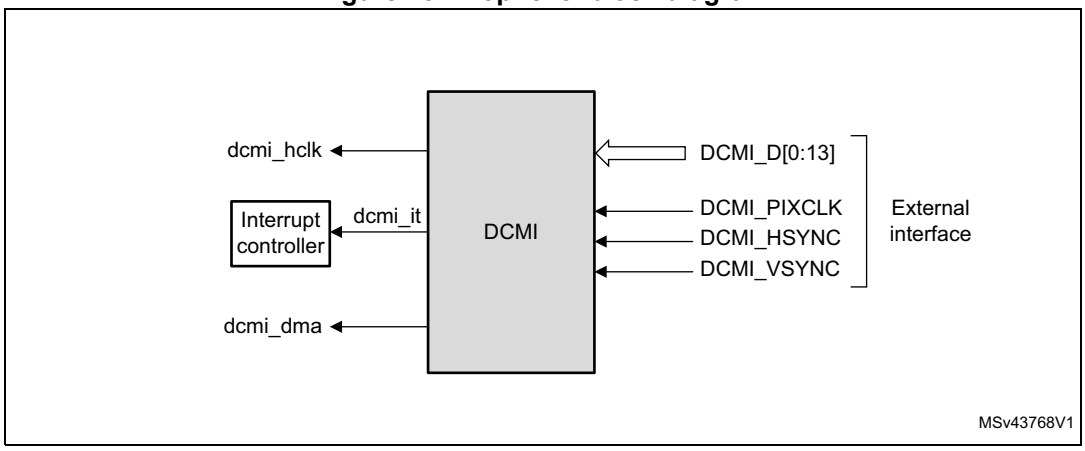


Figure 232. Top-level block diagram



30.4.2 DCMI internal signals

Table 247 shows the DCMI internal signals.

Table 247. DCMI internal signals

Name	Signal type	Description
dcmi_dma	Digital output	DCMI DMA request
dcmi_it	Digital output	DCMI interrupt request
dcmi_hclk	Digital input	DCMI interface clock

30.4.3 DMA interface

The DMA interface is active when the CAPTURE bit in the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

30.4.4 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits in the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

Table 248 shows the DCMI pins.

Table 248. DCMI external signals

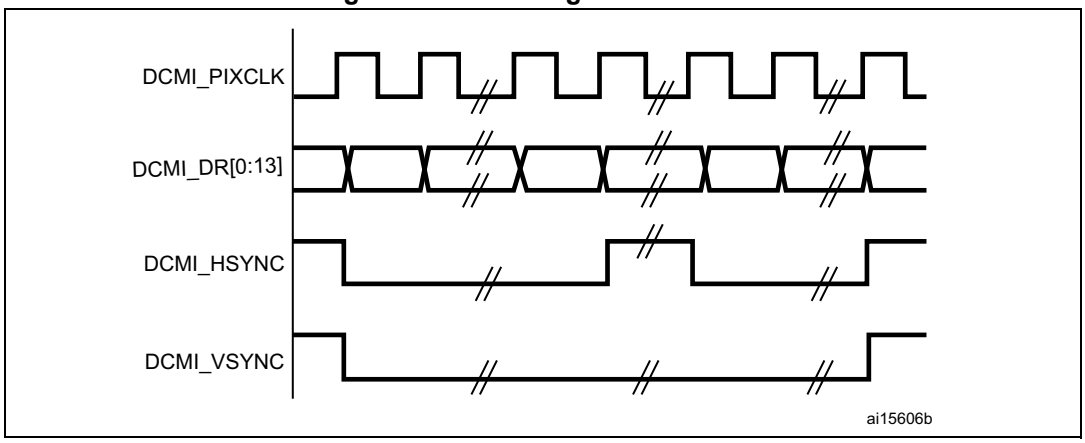
Signal name		Signal type	Signal description
8 bits 10 bits 12 bits 14 bits	DCMI_D[0..7] DCMI_D[0..9] DCMI_D[0..11] DCMI_D[0..13]	Digital inputs	DCMI data
DCMI_PIXCLK		Digital input	Pixel clock
DCMI_HSYNC		Digital input	Horizontal synchronization / Data valid
DCMI_VSYNC		Digital input	Vertical synchronization

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame

Figure 233. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] in DCMI_CR are programmed to “00” the interface captures 8 LSBs at its input (DCMI_D[0:7]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. *Table 249* gives an example of the positioning of captured data bytes in two 32-bit words.

Table 249. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10-bit data

When EDM[1:0] in DCMI_CR are programmed to “01”, the camera interface captures 10-bit data at its input DCMI_D[0..9] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits in the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in *Table 250*.

Table 250. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-bit data

When EDM[1:0] in DCMI_CR are programmed to “10”, the camera interface captures the 12-bit data at its input DCMI_D[0..11] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in *Table 251*.

Table 251. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-bit data

When EDM[1:0] in DCMI_CR are programmed to “11”, the camera interface captures the 14-bit data at its input DCMI_D[0..13] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in *Table 252*.

Table 252. Positioning of captured data bytes in 32-bit words (14-bit width)

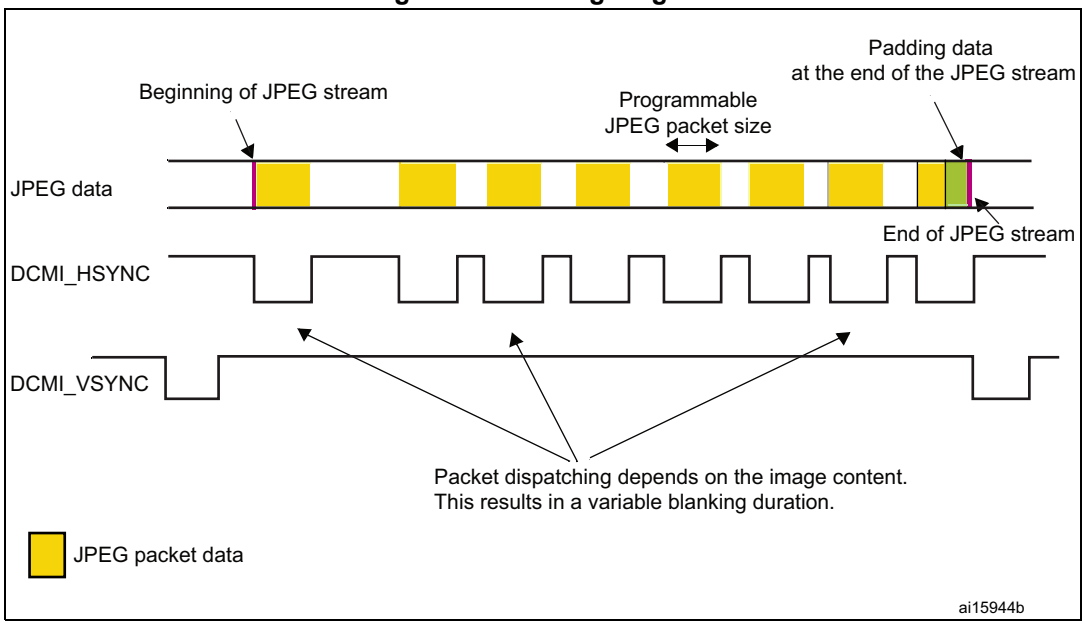
Byte address	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

30.4.5 Synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits should be cleared to “00”).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. *Figure 234* shows the corresponding timing diagram.

Figure 234. Timing diagram



Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits should be programmed to “00”). For other data widths, this mode generates unpredictable results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

  Four embedded codes signal the following events
  - Frame start (FS)
  - Frame end (FE)
  - Line start (LS)
  - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see *Section 30.7.7: DCMI embedded synchronization code register (DCMI_ESCR)*).

  A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.
- Mode 1

  An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS ≤ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. You can therefore select a bit to compare in the embedded code and detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

30.4.6 Capture modes

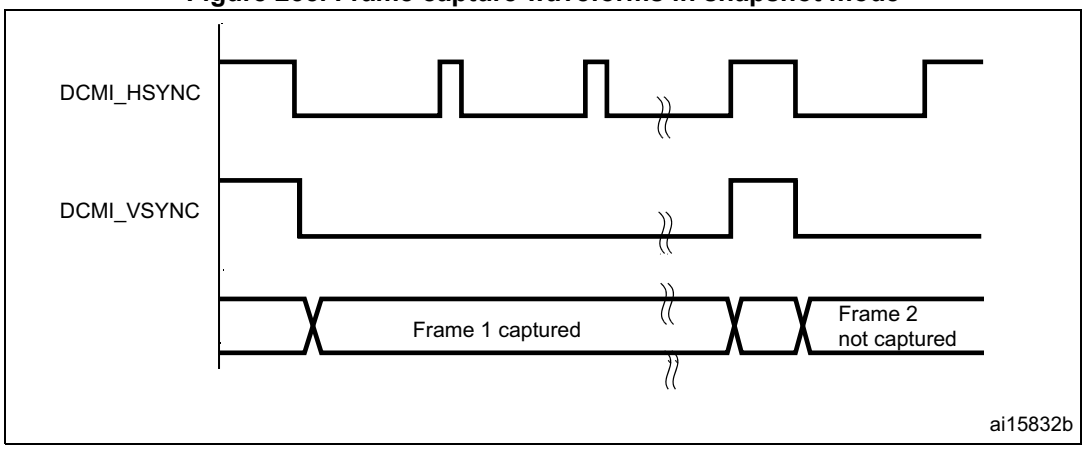
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = ‘1’ in the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 235. Frame capture waveforms in snapshot mode

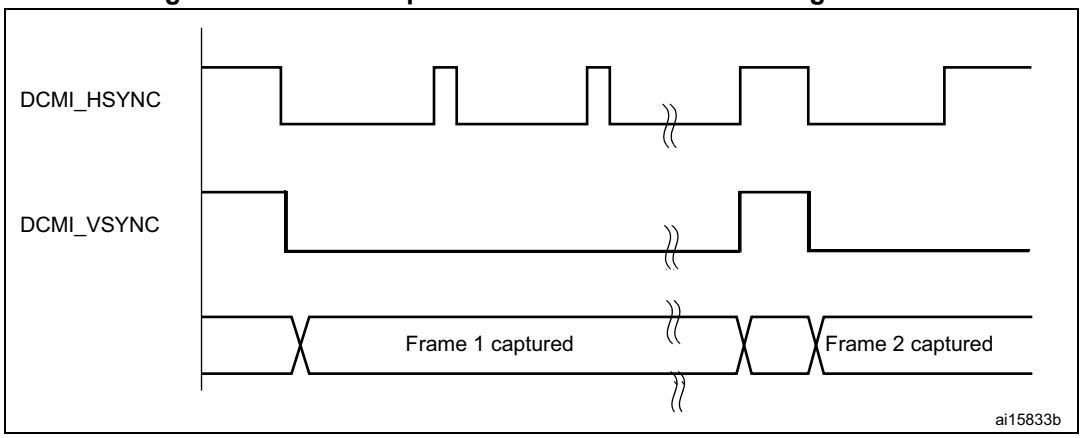


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = ‘0’ in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 236. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

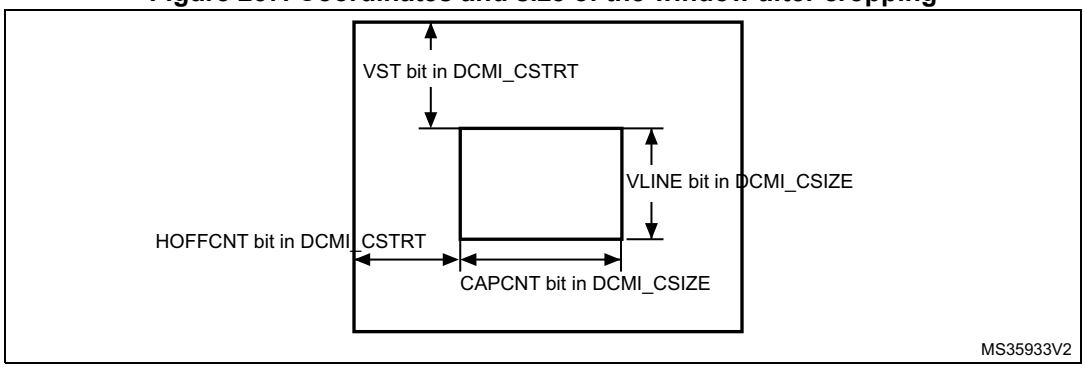
In continuous grab mode, you can configure the FCRC bits in DCMI_CR to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

Note: In the hardware synchronization mode (ESS = '0' in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = '0' in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

30.4.7 Crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

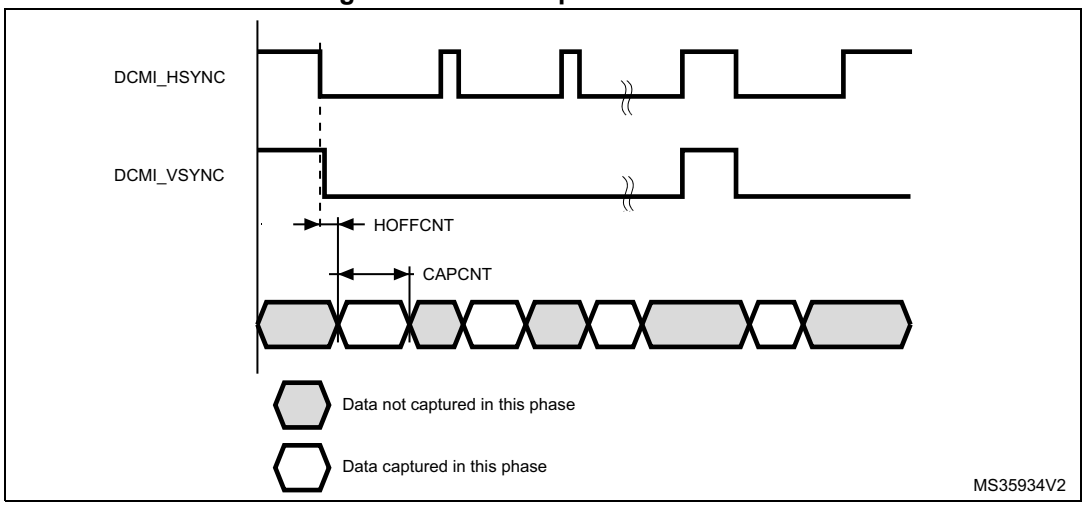
Figure 237. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 238. Data capture waveforms



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

30.4.8 JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit in the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4, you should therefore be careful when handling this case since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with ‘0s’ and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in the JPEG format.

30.4.9 FIFO

Input mode

A four-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

30.5 Data format description

30.5.1 Data formats

Three types of data are supported:

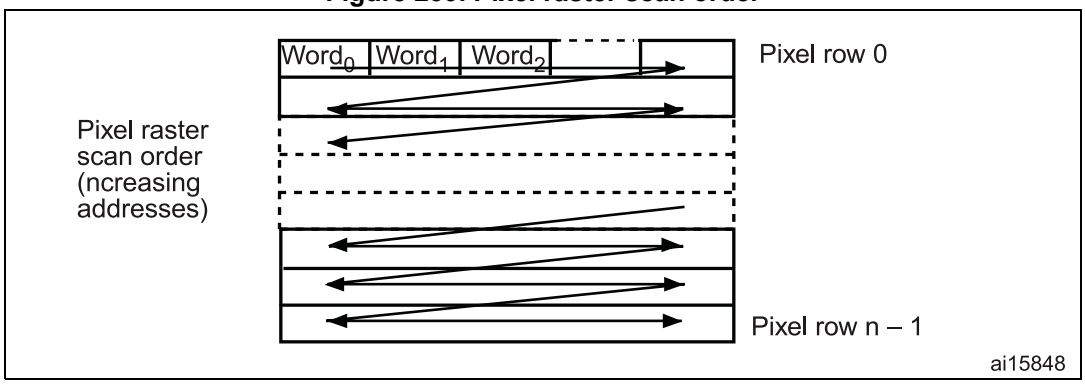
- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred.

Compressed data: JPEG

For B&W, YCbCr or RGB data, the maximum input size is 2048 × 2048 pixels. No limit in JPEG compressed mode.

For monochrome, RGB & YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little endian format is supported.

Figure 239. Pixel raster scan order



30.5.2 Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

Table 253 shows how the data are stored.

Table 253. Data storage in monochrome progressive video format

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

30.5.3 RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G & B interleaved: BRGBRGBRG, etc.
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats.
Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.
The 24 BPP (palletized format) and grayscale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

Table 254 shows how the data are stored.

Table 254. Data storage in RGB progressive video format

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

30.5.4 YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one Buffer: Y, Cb & Cr interleaved: CbYCrYCbYCr, etc.

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in *Table 255*.

Table 255. Data storage in YCbCr progressive video format

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

30.5.5 YCbCr format - Y only

Characteristics:

- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only Luma component of each

pixel , encoded in 8 bits, is stored as shown in *Table 256*.

The result is a monochrome image having the same resolution as the original YCbCr data.

Table 256. Data storage in YCbCr progressive video format - Y extraction mode

Byte address	31:24	23:16	15:8	7:0
0	Y n + 3	Y n + 2	Y n + 1	Y n
4	Y n + 7	Y n + 6	Y n + 5	Y n + 4

30.5.6 Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode allows to only store a half resolution image. It is selected through OELS and LSM control bits.

30.6 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. *Table 257* gives the list of all interrupts.

Table 257. DCMI interrupts

Interrupt name	Interrupt event
IT_LINE	Indicates the end of line
IT_FRAME	Indicates the end of frame capture
IT_OVR	indicates the overrun of data reception
IT_VSYNC	Indicates the synchronization frame
IT_ERR	Indicates the detection of an error in the embedded synchronization frame detection
dcmi_it	Logic OR of the previous interrupts

30.7 DCMI register description

All DCMI registers have to be accessed as 32-bit words, otherwise a bus error occurs.

30.7.1 DCMI control register (DCMI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]	
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OELS:** Odd/Even Line Select (Line Select Start)

This bit works in conjunction with LSM field (LSM = 1)

0: Interface captures first line after the frame start, second one being dropped

1: Interface captures second line from the frame start, first one being dropped

Bit 19 **LSM:** Line Select mode

0: Interface captures all received lines

1: Interface captures one line out of two.

Bit 18 **OEBS:** Odd/Even Byte Select (Byte Select Start)

This bit works in conjunction with BSM field (BSM <> 00)

0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped

1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped

Bits 17:16 **BSM[1:0]:** Byte Select mode

00: Interface captures all received data

01: Interface captures every other byte from the received data

10: Interface captures one byte out of four

11: Interface captures two bytes out of four

Note: This mode only work for EDM[1:0]=00. For all other EDM values, this bit field must be programmed to the reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** DCMI enable

0: DCMI disabled

1: DCMI enabled

Note: The DCMI configuration registers should be programmed correctly before enabling this Bit

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode

00: Interface captures 8-bit data on every pixel clock

01: Interface captures 10-bit data on every pixel clock

10: Interface captures 12-bit data on every pixel clock

11: Interface captures 14-bit data on every pixel clock

Bits 9:8 **FCRC[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.

00: All frames are captured

01: Every alternate frame captured (50% bandwidth reduction)

10: One frame in 4 frames captured (75% bandwidth reduction)

11: reserved

Bit 7 **VSPOL:** Vertical synchronization polarity

This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.

0: DCMI_VSYNC active low

1: DCMI_VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity

This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.

0: DCMI_HSYNC active low

1: DCMI_HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity

This bit configures the capture edge of the pixel clock

0: Falling edge active.

1: Rising edge active.

Bit 4 **ESS:** Embedded synchronization select

0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.

1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.

Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.

This bit is disabled in JPEG mode.

Bit 3 **JPEG:** JPEG format

0: Uncompressed video format

1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2 **CROP:** Crop feature

0: The full image is captured. In this case the total number of bytes in an image frame should be a multiple of 4

1: Only the data inside the window specified by the crop register will be captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE:** Capture enable

0: Capture disabled.

1: Capture enabled.

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the 1st frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit will be effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers should be programmed correctly before enabling this bit.

30.7.2 DCMI status register (DCMI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FNE**: FIFO not empty

This bit gives the status of the FIFO

1: FIFO contains valid data

0: FIFO empty

Bit 1 **VSYNC**:

This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity.

When embedded synchronization codes are used, the meaning of this bit is the following:

0: active frame

1: synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**:

This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity.

When embedded synchronization codes are used, the meaning of this bit is the following:

0: active line

1: synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

30.7.3 DCMI raw interrupt status register (DCMI_RIS)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
											r	r	r	r	r

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_RIS**: Line raw interrupt status

This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by writing a ‘1’ to the LINE_ISC bit in DCMI_ICR.

Bit 3 **VSYNC_RIS**: DCMI_VSYNC raw interrupt status

This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by writing a ‘1’ to the VSYNC_ISC bit in DCMI_ICR.

Bit 2 **ERR_RIS**: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing a ‘1’ to the ERR_ISC bit in DCMI_ICR.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_RIS**:Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

This bit is cleared by writing a ‘1’ to the OVR_ISC bit in DCMI_ICR.

Bit 0 **FRAME_RIS**: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (e.g. window cropped outside the frame).

This bit is cleared by writing a ‘1’ to the FRAME_ISC bit in DCMI_ICR.

30.7.4 DCMI interrupt enable register (DCMI_IER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
											rw	rw	rw	rw	rw

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_IE**: Line interrupt enable
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received

Bit 3 **VSYNC_IE**: DCMI_VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE**: Synchronization error interrupt enable
0: No interrupt generation
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE**: Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE**: Capture complete interrupt enable
0: No interrupt generation
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

30.7.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_MIS**: Line masked interrupt status
This bit gives the status of the masked line interrupt
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS**: VSYNC masked interrupt status
This bit gives the status of the masked VSYNC interrupt
0: No interrupt is generated on DCMI_VSYNC transitions
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS**: Synchronization error masked interrupt status
This bit gives the status of the masked synchronization error interrupt
0: No interrupt is generated on a synchronization error
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS**: Overrun masked interrupt status
This bit gives the status of the masked overflow interrupt
0: No interrupt is generated on overrun
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS**: Capture complete masked interrupt status
This bit gives the status of the masked capture complete interrupt
0: No interrupt is generated after a complete capture
1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

30.7.6 DCMI interrupt clear register (DCMI_ICR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
											w	w	w	w	w

The DCMI_ICR register is write-only. Writing a ‘1’ into a bit of this register clears the corresponding bit in the DCMI_RIS and DCMI_MIS registers. Writing a ‘0’ has no effect.

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_ISC**: line interrupt status clear
Writing a ‘1’ into this bit clears LINE_RIS in the DCMI_RIS register

Bit 3 **VSYNC_ISC**: Vertical Synchronization interrupt status clear
Writing a ‘1’ into this bit clears the VSYNC_RIS bit in DCMI_RIS

Bit 2 **ERR_ISC**: Synchronization error interrupt status clear
Writing a ‘1’ into this bit clears the ERR_RIS bit in DCMI_RIS
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC**: Overrun interrupt status clear
Writing a ‘1’ into this bit clears the OVR_RIS bit in DCMI_RIS

Bit 0 **FRAME_ISC**: Capture complete interrupt status clear
Writing a ‘1’ into this bit clears the FRAME_RIS bit in DCMI_RIS

30.7.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEC[7:0]]								LEC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSC[7:0]								FSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEC[7:0]**: Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the 1st occurrence of LSC after an FEC code will be interpreted as a start of frame delimiter.

30.7.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEU[7:0]								LEU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSU[7:0]								FSU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.

0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.

1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.

0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data

1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.

0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data

1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.

0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data

1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data

30.7.9 DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	VST[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HOFFCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **VST[12:0]**: Vertical start line count

The image capture starts with this line number. Previous line data are ignored.
0x0000 => line 1
0x0001 => line 2
0x0002 => line 3
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **HOFFCNT[13:0]**: Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

30.7.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	VLINE[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CAPCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **VLINE[13:0]**: Vertical line count

This value gives the number of lines to be captured from the starting point.
0x0000 => 1 line
0x0001 => 2 lines
0x0002 => 3 lines
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CAPCNT[13:0]**: Capture count

This value gives the number of pixel clocks to be captured from the starting point on the same line. It value should corresponds to word-aligned data for different widths of parallel interfaces.

0x0000 => 1 pixel

0x0001 => 2 pixels

0x0002 => 3 pixels

....

30.7.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Byte3[7:0]								Byte2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte1[7:0]								Byte0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **Byte3[7:0]**: Data byte 3

Bits 23:16 **Byte2[7:0]**: Data byte 2

Bits 15:8 **Byte1[7:0]**: Data byte 1

Bits 7:0 **Byte0[7:0]**: Data byte 0

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 4-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

30.7.12 DCMI register map

Table 258 summarizes the DCMI registers.

Table 258. DCMI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM		Res.	ENABLE	Res.	Res.	EDM		FCRC		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value												0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC								LEC								LSC								FSC							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DCMI_ESUR	FEU								LEU								LSU								FSU							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DCMI_CWSTRT	Res.	Res.	Res.	VST[12:0													Res.	Res.	HOFFCNT[13:0]													
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DCMI_CWSIZE	Res.	Res.	VLINE13:0]														Res.	Res.	CAPCNT[13:0]													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DCMI_DR	Byte3								Byte2								Byte1								Byte0							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

31 LCD-TFT display controller (LTDC)

31.1 Introduction

The LCD-TFT (liquid crystal display - thin film transistor) display controller provides a parallel digital RGB (red, green, blue) and signals for horizontal, vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD and TFT panels.

31.2 LTDC main features

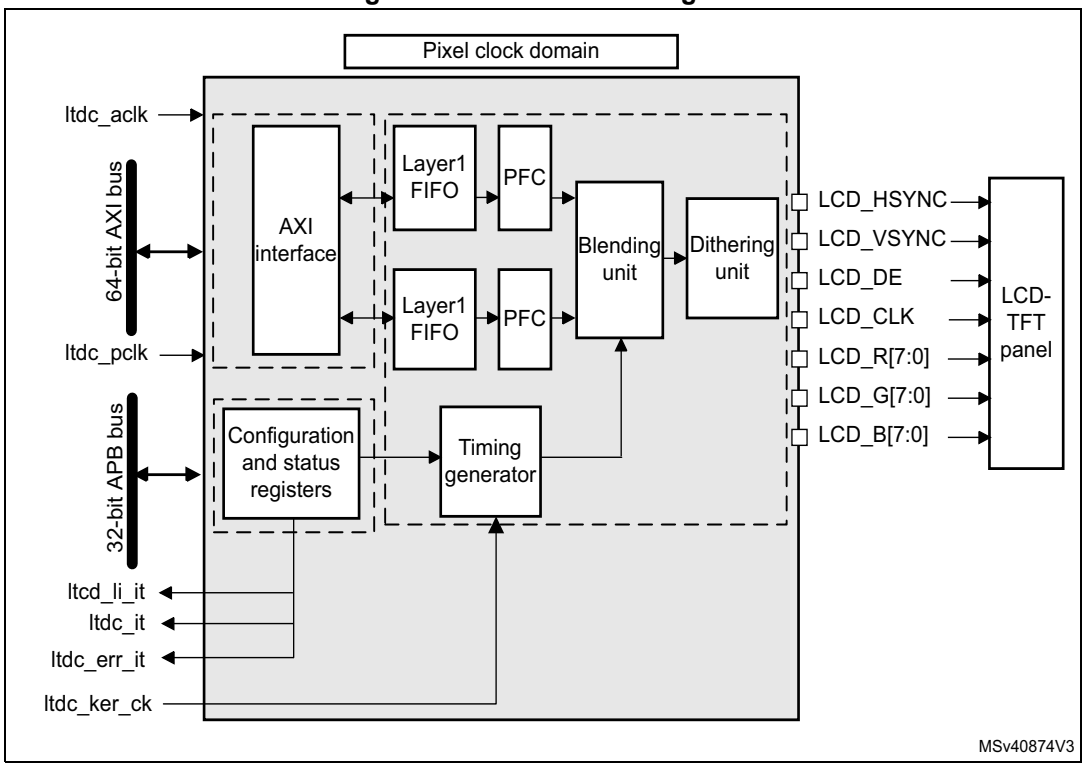
- 24-bit RGB parallel pixel output; 8 bits-per-pixel (RGB888)
- 2 display layers with dedicated FIFO (64x64-bit)
- Color look-up table (CLUT) up to 256 color (256x24-bit) per layer
- Programmable timings for different display panels
- Programmable background color
- Programmable polarity for HSYNC, VSYNC and data enable
- Up to 8 input color formats selectable per layer
  - ARGB8888
  - RGB888
  - RGB565
  - ARGB1555
  - ARGB4444
  - L8 (8-bit luminance or CLUT)
  - AL44 (4-bit alpha + 4-bit luminance)
  - AL88 (8-bit alpha + 8-bit luminance)
- Pseudo-random dithering output for low bits per channel
  - Dither width 2-bits for red, green, blue
- Flexible blending between two layers using alpha value (per pixel or constant)
- Color keying (transparency color)
- Programmable window position and size
- Supports thin film transistor (TFT) color displays
- AXI master interface with burst of 16 double-words
- Up to 4 programmable interrupt events

31.3 LTDC functional description

31.3.1 LTDC block diagram

The block diagram of the LTDC is shown in *Figure 240: LTDC block diagram*.

Figure 240. LTDC block diagram



Layer FIFO: One FIFO 64x64-bit per layer.

PFC: pixel format converter, performing the pixel format conversion from the selected input pixel format of a layer to words.

AXI interface: for data transfer from memories to the FIFO.

Blending, dithering unit and timings generator: Refer to *Section 31.4.1* and *Section 31.4.2*.

31.3.2 LCD-TFT internal signals

Table 259 gives the list of LCD-TFT internal signals.

Table 259. LCD-TFT internal signals

Names	Signal type	Description
ltdc_aclk	Digital input	Clock for LCD-TFT registers in AXI clock domain
ltdc_pclk	Digital input	LCD-TFT register interface clock
ltdc_ker_ck	Digital input	LCD-TFT kernel clock used for LCD_CLK (pixel clock) generation
ltdc_li_it	Digital output	LCD-TFT line interrupt trigger for MDMA

Table 259. LCD-TFT internal signals (continued)

Names	Signal type	Description
ltdc_it	Digital output	LCD-TFT global interrupt request
ltdc_err_it	Digital output	LCD-TFT global error interrupt request

31.3.3 LTDC pins and external signal interface

Table 260 summarizes the LTDC signal interface.

Table 260. LTDC pins and signal interface

LCD-TFT signals	I/O	Description
LCD_CLK	O	Clock output
LCD_HSYNC	O	Horizontal synchronization
LCD_VSYNC	O	Vertical synchronization
LCD_DE	O	Not data enable
LCD_R[7:0]	O	Data: 8-bit red data
LCD_G[7:0]	O	Data: 8-bit green data
LCD_B[7:0]	O	Data: 8-bit blue data

The LTDC-TFT controller pins must be configured by the user application. The unused pins can be used for other purposes.

For LTDC outputs up to 24-bit (RGB888), if less than 8 bpp are used to output for example RGB565 or RGB666 to interface on 16- or 18-bit displays, the RGB display data lines must be connected to the MSB of the LCD-TFT controller RGB data lines. As an example, in the case of an LCD-TFT controller interfacing with a RGB565 16-bit display, the LCD display R[4:0], G[5:0] and B[4:0] data lines pins must be connected to LCD-TFT controller LCD_R[7:3], LCD_G[7:2] and LCD_B[7:3].

31.3.4 LTDC reset and clocks

The LCD-TFT controller peripheral uses 3 clock domains:

- AXI clock domain (ltdc_aclk)

  This domain contains the LCD-TFT AXI master interface for data transfer from the memories to the Layer FIFO and the frame buffer configuration register
- APB clock domain (ltdc_pclk):

  This domain contains the global configuration registers and the interrupt register.
- Pixel clock domain (LCD_CLK)

  This domain contains the pixel data generation, the layer configuration register as well as the LCD-TFT interface signal generator. The LCD_CLK output should be configured following the panel requirements. The LCD_CLK is generated from a specific PLL output (refer to the reset and clock control section).

Table 261 summarizes the clock domain for each register.

Table 261. Clock domain for each register

LTDC register	Clock domain
LTDC_LxCR	ltdc_aclk
LTDC_LxCFBAR	
LTDC_LxCFBLR	
LTDC_LxCFBLNR	
LTDC_SRCR	ltdc_pclk
LTDC_IER	
LTDC_ISR	
LTDC_ICR	
LTDC_SSCR	Pixel clock (LCD_CLK)
LTDC_BPCR	
LTDC_AWCR	
LTDC_TWCR	
LTDC_GCR	
LTDC_BCCR	
LTDC_LIPCR	
LTDC_CPSR	
LTDC_CDSR	
LTDC_LxWHPCR	
LTDC_LxWVPCR	
LTDC_LxCKCR	
LTDC_LxPFCR	
LTDC_LxCACR	
LTDC_LxDCCR	
LTDC_LxBFCR	
LTDC_LxCLUTWR	

Care must be taken while accessing the LTDC registers, the APB bus is stalled during the access for a given time period (refer to *Table 262*).

Table 262. LTDC register access and update durations

-	Register clock domain		
-	**AXI domain**	**APB domain**	**Pixel clock domain**
Register read access duration	7 x ltdc_pclk + 5 x ltdc_aclk	7 x ltdc_pclk	7 x ltdc_pclk + 5 x ltdc_ker_clk
Register write access duration	6 x ltdc_pclk + 5 x ltdc_aclk	6 x ltdc_pclk	6 x ltdc_pclk + 5 x ltdc_ker_clk

The LCD controller can be reset by setting the corresponding bit in the RCC_APBRSTR register. It resets the three clock domains.

31.4 LTDC programmable parameters

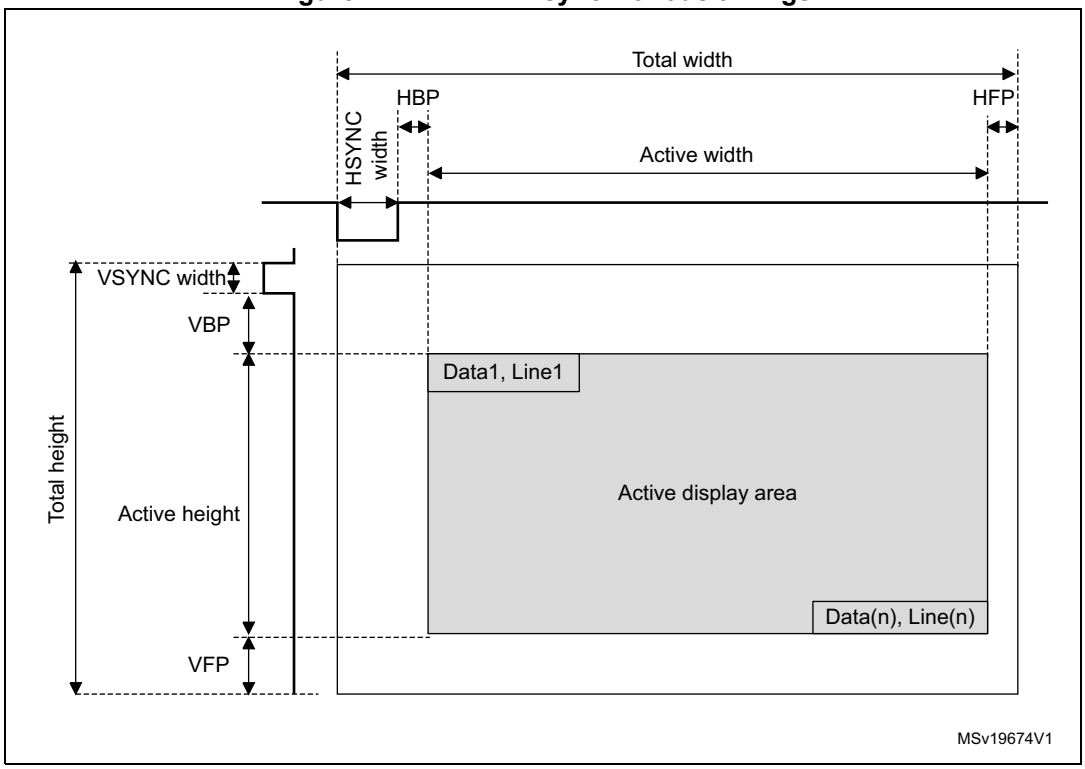
The LCD-TFT controller provides flexible configurable parameters. It can be enabled or disabled through the LTDC_GCR register.

31.4.1 LTDC global configuration parameters

Synchronous timings

Figure 241 presents the configurable timing parameters generated by the synchronous timings generator block presented in the block diagram *Figure 240*. It generates the horizontal and vertical synchronization timings panel signals, the pixel clock and the data enable signals.

Figure 241. LCD-TFT synchronous timings



Note: The HBP and HFP are respectively the horizontal back porch and front porch period.

The VBP and the VFP are respectively the vertical back porch and front porch period.

The LCD-TFT programmable synchronous timings are:

- HSYNC and VSYNC width: horizontal and vertical synchronization width, configured by programming a value of **HSYNC width - 1** and **VSYNC width - 1** in the LTDC_SSCR register
- HBP and VBP: horizontal and vertical synchronization back porch width, configured by programming the accumulated value **HSYNC width + HBP - 1** and the accumulated value **VSYNC width + VBP - 1** in the LTDC_BPCR register.
- Active width and active height: the active width and active height are configured by programming the accumulated value **HSYNC width + HBP + active width - 1** and the accumulated value **VSYNC width + VBP + active height - 1** in the LTDC_AWCR register (only up to 1024x768 is supported).
- Total width: the total width is configured by programming the accumulated value **HSYNC width + HBP + active width + HFP - 1** in the LTDC_TWCR register. The HFP is the horizontal front porch period.
- Total height: the total height is configured by programming the accumulated value **VSYNC height + VBP + active height + VFP - 1** in the LTDC_TWCR register. The VFP is the vertical front porch period.

Note: When the LTDC is enabled, the timings generated start with X/Y=0/0 position as the first horizontal synchronization pixel in the vertical synchronization area and following the back porch, active data display area and the front porch.

When the LTDC is disabled, the timing generator block is reset to X = *total width - 1*, Y = *total height - 1* and held the last pixel before the vertical synchronization phase and the FIFO are flushed. Therefore only blanking data is output continuously.

Example of synchronous timings configuration

TFT-LCD timings (must be extracted from panel datasheet):

- horizontal and vertical synchronization width: 0xA pixels and 0x2 lines
- horizontal and vertical back porch: 0x14 pixels and 0x2 lines
- active width and active height: 0x140 pixels, 0xF0 lines (320x240)
- horizontal front porch: 0xA pixels
- vertical front porch: 0x4 lines

The programmed values in the LTDC timings registers are:

- LTDC_SSCR register: to be programmed to 0x00090001. (HSW[11:0] is 0x9 and VSH[10:0] is 0x1)
- LTDC_BPCR register: to be programmed to 0x001D0003 (AHBP[11:0] is 0x1D(0xA+ 0x13) and AVBP[10:0]A is 0x3(0x2 + 0x1)).
- LTDC_AWCR register: to be programmed to 0x015D00F3 (AAW[11:0] is 0x15D(0xA +0x14 +0x13F) and AAH[10:0] is 0xF3(0x2 +0x2 + 0xEF).
- LTDC_TWCR register: to be programmed to 0x00000167 (TOTALW[11:0] is 0x167(0xA +0x14 +0x140 + 0x9).
- LTDC_THCR register: to be programmed to 0x000000F7 (TOTALH[10:0]is 0xF7(0x2 +0x2 + 0xF0 + 3)

Programmable polarity

The horizontal and vertical synchronization, data enable and pixel clock output signals polarity can be programmed to active high or active low through the LTDC_GCR register.

Background color

A constant background color (RGB888) can programmed through the LTDC_BCCR register. It is used for blending with the bottom layer.

Dithering

The dithering pseudo-random technique using an LFSR is used to add a small random value (threshold) to each pixel color channel (R, G or B) value, thus rounding up the MSB in some cases when displaying a 24-bit data on 18-bit display. Thus the Dithering technique is used to round data which is different from one frame to the other.

The dithering pseudo-random technique is the same as comparing LSBs against a threshold value and adding a 1 to the MSB part only, if the LSB part is >= the threshold. The LSBs are typically dropped once dithering was applied.

The width of the added pseudo-random value is 2 bits for each color channel: 2 bits for red, 2 bits for green and 2 bits for blue.

Once the LCD-TFT controller is enabled, the LFSR starts running with the first active pixel and it is kept running even during blanking periods and when dithering is switched off. If the LTDC is disabled, the LFSR is reset.

The dithering can be switched On and Off on the fly through the LTDC_GCR register.

Reload shadow registers

Some configuration registers are shadowed. The shadow registers values can be reloaded immediately to the active registers when writing to these registers or at the beginning of the vertical blanking period following the configuration in the LTDC_SRCR register. If the immediate reload configuration is selected, the reload should be only activated when all new registers have been written.

The shadow registers should not be modified again before the reload has been done. Reading from the shadow registers returns the actual active value. The new written value can only be read after the reload has taken place.

A register reload interrupt can be generated if enabled in the LTDC_IER register.

The shadowed registers are all the layer1 and layer2 registers except the LTDC_LxCLUTWR register.

Interrupt generation event

Refer to *Section 31.5: LTDC interrupts* for interrupt configuration.

31.4.2 Layer programmable parameters

Up to two layers can be enabled, disabled and configured separately. The layer display order is fixed and it is bottom up. If two layers are enabled, the layer2 is the top displayed window.

Windowing

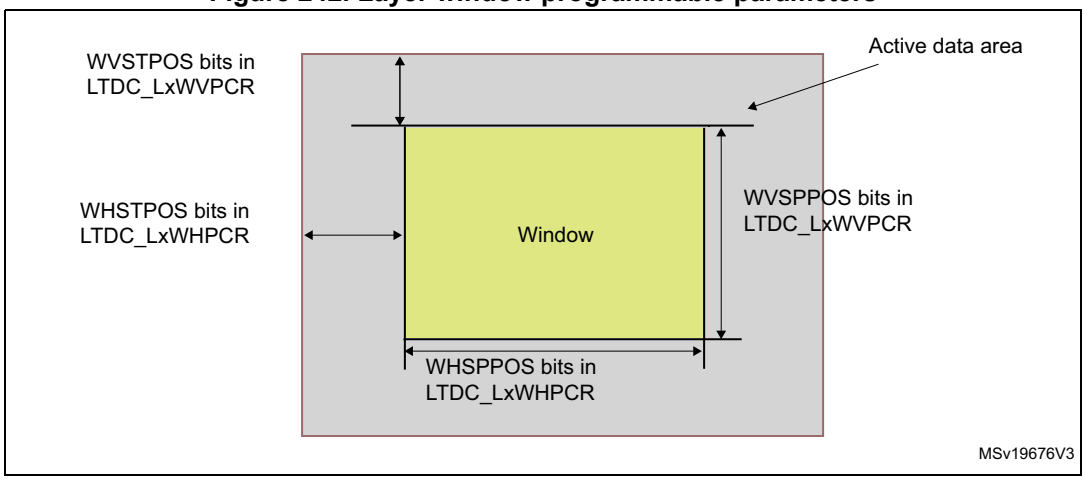
Every layer can be positioned and resized and it must be inside the active display area.

The window position and size are configured through the top-left and bottom-right X/Y positions and the internal timing generator that includes the synchronous, back porch size and the active data area. Refer to LTDC_LxWHPCR and LTDC_WVPCR registers.

The programmable layer position and size defines the first/last visible pixel of a line and the first/last visible line in the window. It allows to display either the full image frame or only a part of the image frame. Refer to *Figure 242*.

- The first and the last visible pixel in the layer are set by configuring the WHSTPOS[11:0] and WHSPPOS[11:0] in the LTDC_LxWHPCR register.
- The first and the last visible lines in the layer are set by configuring the WVSTPOS[10:0] and WVSPPOS[10:0] in the LTDC_LxWVPCR register.

Figure 242. Layer window programmable parameters



Pixel input format

The programmable pixel format is used for the data stored in the frame buffer of a layer.

Up to 8 input pixel formats can be configured for every layer through the LTDC_LxPFCR register

The pixel data is read from the frame buffer and then transformed to the internal 8888 (ARGB) format as follows: components having a width of less than 8 bits get expanded to 8 bits by bit replication. The selected bit range is concatenated multiple times until it is longer than 8 bits. Of the resulting vector, the 8 MSB bits are chosen. Example: 5 bits of an RGB565 red channel become (bit positions): 43210432 (the 3 LSBs are filled with the 3 MSBs of the 5 bits)

Table 263 describes the pixel data mapping depending on the selected format.

Table 263. Pixel data mapping versus color format

ARGB8888			
@+3 A_x[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 A_{x+1}[7:0]	@+6 R_{x+1}[7:0]	@+5 G_{x+1}[7:0]	@+4 B_{x+1}[7:0]
RGB888			
@+3 B_{x+1}[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]

Table 263. Pixel data mapping versus color format (continued)

@+7 $G_{x+2}[7:0]$	@+6 $B_{x+2}[7:0]$	@+5 $R_{x+1}[7:0]$	@+4 $G_{x+1}[7:0]$
RGB565			
@+3 $R_{x+1}[4:0]$ $G_{x+1}[5:3]$	@+2 $G_{x+1}[2:0]$ $B_{x+1}[4:0]$	@+1 $R_x[4:0]$ $G_x[5:3]$	@ $G_x[2:0]$ $B_x[4:0]$
@+7 $R_{x+3}[4:0]$ $G_{x+3}[5:3]$	@+6 $G_{x+3}[2:0]$ $B_{x+3}[4:0]$	@+5 $R_{x+2}[4:0]$ $G_{x+2}[5:3]$	@+4 $G_{x+2}[2:0]$ $B_{x+2}[4:0]$
ARGB1555			
@+3 $A_{x+1}[0]R_{x+1}[4:0]$ $G_{x+1}[4:3]$	@+2 $G_{x+1}[2:0]$ $B_{x+1}[4:0]$	@+1 $A_x[0]$ $R_x[4:0]$ $G_x[4:3]$	@ $G_x[2:0]$ $B_x[4:0]$
@+7 $A_{x+3}[0]R_{x+3}[4:0]$ $G_{x+3}[4:3]$	@+6 $G_{x+3}[2:0]$ $B_{x+3}[4:0]$	@+5 $A_{x+2}[0]R_{x+2}[4:0]G_{x+2}[4:3]$	@+4 $G_{x+2}[2:0]$ $B_{x+2}[4:0]$
ARGB4444			
@+3 $A_{x+1}[3:0]R_{x+1}[3:0]$	@+2 $G_{x+1}[3:0]$ $B_{x+1}[3:0]$	@+1 $A_x[3:0]$ $R_x[3:0]$	@ $G_x[3:0]$ $B_x[3:0]$
@+7 $A_{x+3}[3:0]R_{x+3}[3:0]$	@+6 $G_{x+3}[3:0]$ $B_{x+3}[3:0]$	@+5 $A_{x+2}[3:0]R_{x+2}[3:0]$	@+4 $G_{x+2}[3:0]$ $B_{x+2}[3:0]$
L8			
@+3 $L_{x+3}[7:0]$	@+2 $L_{x+2}[7:0]$	@+1 $L_{x+1}[7:0]$	@ $L_x[7:0]$
@+7 $L_{x+7}[7:0]$	@+6 $L_{x+6}[7:0]$	@+5 $L_{x+5}[7:0]$	@+4 $L_{x+4}[7:0]$
AL44			
@+3 $A_{x+3}[3:0]$ $L_{x+3}[3:0]$	@+2 $A_{x+2}[3:0]$ $L_{x+2}[3:0]$	@+1 $A_{x+1}[3:0]$ $L_{x+1}[3:0]$	@ $A_x[3:0]$ $L_x[3:0]$
@+7 $A_{x+7}[3:0]$ $L_{x+7}[3:0]$	@+6 $A_{x+6}[3:0]$ $L_{x+6}[3:0]$	@+5 $A_{x+5}[3:0]$ $L_{x+5}[3:0]$	@+4 $A_{x+4}[3:0]$ $L_{x+4}[3:0]$
AL88			
@+3 $A_{x+1}[7:0]$	@+2 $L_{x+1}[7:0]$	@+1 $A_x[7:0]$	@ $L_x[7:0]$
@+7 $A_{x+3}[7:0]$	@+6 $L_{x+3}[7:0]$	@+5 $A_{x+2}[7:0]$	@+4 $L_{x+2}[7:0]$

Color look-up table (CLUT)

The CLUT can be enabled at run-time for every layer through the LTDC_LxCR register and it is only useful in case of indexed color when using the L8, AL44 and AL88 input pixel format.

First, the CLUT has to be loaded with the R, G and B values that replace the original R, G, B values of that pixel (indexed color). Each color (RGB value) has its own address which is the position within the CLUT.

The R, G and B values and their own respective address are programmed through the LTDC_LxCLUTWR register.

- In case of L8 and AL88 input pixel format, the CLUT has to be loaded by 256 colors. The address of each color is configured in the CLUTADD bits in the LTDC_LxCLUTWR register.
- In case of AL44 input pixel format, the CLUT has to be only loaded by 16 colors. The address of each color must be filled by replicating the 4-bit L channel to 8-bit as follows:
  - L0 (indexed color 0), at address 0x00
  - L1, at address 0x11
  - L2, at address 0x22
  - .....
  - L15, at address 0xFF

Color frame buffer address

Every layer has a start address for the color frame buffer configured through the LTDC_LxCFBAR register.

When a layer is enabled, the data is fetched from the color frame buffer.

Color frame buffer length

Every layer has a total line length setting for the color frame buffer in bytes and a number of lines in the frame buffer configurable in the LTDC_LxCFBLR and LTDC_LxCFBLNR register respectively.

The line length and the number of lines settings are used to stop the prefetching of data to the layer FIFO at the end of the frame buffer.

- If it is set to less bytes than required, a FIFO underrun interrupt is generated if it has been previously enabled.
- If it is set to more bytes than actually required, the useless data read from the FIFO is discarded. The useless data is not displayed.

Color frame buffer pitch

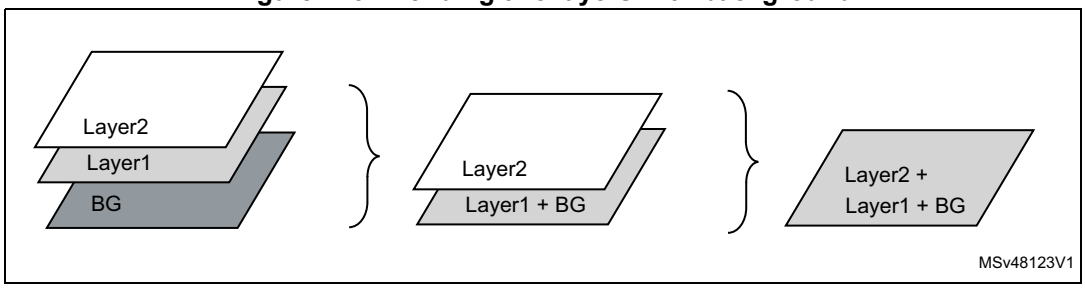
Every layer has a configurable pitch for the color frame buffer, which is the distance between the start of one line and the beginning of the next line in bytes. It is configured through the LTDC_LxCFBLR register.

Layer blending

The blending is always active and the two layers can be blended following the blending factors configured through the LTDC_LxBFCR register.

The blending order is fixed and it is bottom up. If two layers are enabled, first the Layer1 is blended with the Background color, then the layer2 is blended with the result of blended color of layer1 and the background. Refer to *Figure 243*.

Figure 243. Blending two layers with background



Default color

Every layer can have a default color in the format ARGB which is used outside the defined layer window or when a layer is disabled.

The default color is configured through the LTDC_LxDCCR register.

The blending is always performed between the two layers even when a layer is disabled. To avoid displaying the default color when a layer is disabled, keep the blending factors of this layer in the LTDC_LxBFCR register to their reset value.

Color keying

A color key (RGB) can be configured to be representative for a transparent pixel.

If the color keying is enabled, the current pixels (after format conversion and before CLUT respectively blending) are compared to the color key. If they match for the programmed RGB value, all channels (ARGB) of that pixel are set to 0.

The color key value can be configured and used at run-time to replace the pixel RGB value.

The color keying is enabled through the LTDC_LxCKCR register.

The color keying is configured through the LTDC_LxCKCR register. The programmed value depends on the pixel format as it is compared to current pixel after pixel format conversion to ARGB888.

Example: if the a mid-yellow color (50% red + 50% green) is used as the transparent color key:

- In RGB565, the mid-yellow color is 0x8400. Set the LTDC_LxCKCR to 0x848200.
- In ARGB8888, the mid-yellow color is 0x808000, set LTDC_LxCKCR to 0x808000.
- In all CLUT-based color modes (L8, AL88, AL44), set one of the palette entry to the mid-yellow color 0x808000 and set the LTDC_LxCKCR to 0x808000.

31.5 LTDC interrupts

The LTDC provides four maskable interrupts logically ORed to two interrupt vectors.

The interrupt sources can be enabled or disabled separately through the LTDC_IER register. Setting the appropriate mask bit to 1 enables the corresponding interrupt.

The two interrupts are generated on the following events:

- Line interrupt: generated when a programmed line is reached. The line interrupt position is programmed in the LTDC_LIPCR register
- Register reload interrupt: generated when the shadow registers reload was performed during the vertical blanking period
- FIFO underrun interrupt: generated when a pixel is requested from an empty layer FIFO
- Transfer error interrupt: generated when an AXI bus error occurs during data transfer

Those interrupts events are connected to the NVIC controller as described in *Figure 244*.

Figure 244. Interrupt events

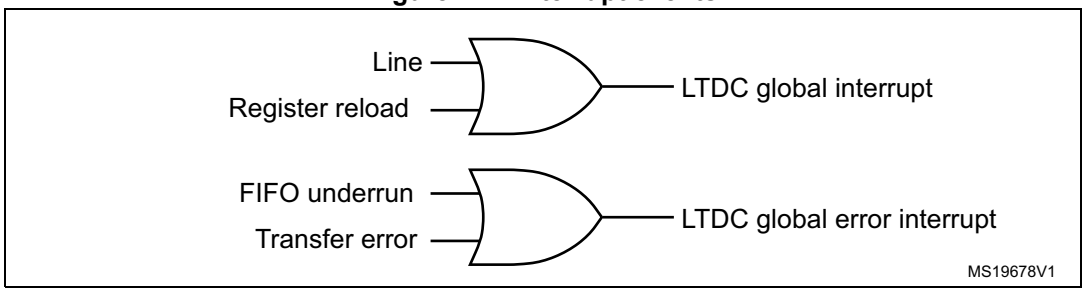


Table 264. LTDC interrupt requests

Interrupt event	Event flag	Enable control bit
Line	LIF	LIE
Register reload	RRIF	RRIEN
FIFO underrun	FUDERRIF	FUDERRIE
Transfer error	TERRIF	TERRIE

31.6 LTDC programming procedure

- Enable the LTDC clock in the RCC register.
- Configure the required pixel clock following the panel datasheet.
- Configure the synchronous timings: VSYNC, HSYNC, vertical and horizontal back porch, active data area and the front porch timings following the panel datasheet as described in the *Section 31.4.1: LTDC global configuration parameters*.
- Configure the synchronous signals and clock polarity in the LTDC_GCR register.
- If needed, configure the background color in the LTDC_BCCR register.
- Configure the needed interrupts in the LTDC_IER and LTDC_LIPCR register.
- Configure the layer1/2 parameters by:
  - programming the layer window horizontal and vertical position in the LTDC_LxWHPCR and LTDC_WVPCR registers. The layer window must be in the active data area.
  - programming the pixel input format in the LTDC_LxPFCR register
  - programming the color frame buffer start address in the LTDC_LxCFBAR register
  - programming the line length and pitch of the color frame buffer in the LTDC_LxCFBLR register
  - programming the number of lines of the color frame buffer in the LTDC_LxCFBLNR register
  - if needed, loading the CLUT with the RGB values and its address in the LTDC_LxCLUTWR register
  - If needed, configuring the default color and the blending factors respectively in the LTDC_LxDCCR and LTDC_LxBFCR registers
- Enable layer1/2 and if needed the CLUT in the LTDC_LxCR register.
- If needed, enable dithering and color keying respectively in the LTDC_GCR and LTDC_LxCKCR registers. They can be also enabled on the fly.
- Reload the shadow registers to active register through the LTDC_SRCR register.
- Enable the LCD-TFT controller in the LTDC_GCR register.
- All layer parameters can be modified on the fly except the CLUT. The new configuration has to be either reloaded immediately or during vertical blanking period by configuring the LTDC_SRCR register.

Note: All layer's registers are shadowed. Once a register is written, it must not be modified again before the reload has been done. Thus, a new write to the same register overrides the previous configuration if not yet reloaded.

31.7 LTDC registers

31.7.1 LTDC synchronization size configuration register (LTDC_SSCR)

This register defines the number of horizontal synchronization pixels minus 1 and the number of vertical synchronization lines minus 1. Refer to *Figure 241* and *Section 31.4: LTDC programmable parameters* for an example of configuration.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HSW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	VSH[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HSW[11:0]**: horizontal synchronization width (in units of pixel clock period)
These bits define the number of Horizontal Synchronization pixel minus 1.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **VSH[10:0]**: vertical synchronization height (in units of horizontal scan line)
These bits define the vertical Synchronization height minus 1. It represents the number of horizontal synchronization lines.

31.7.2 LTDC back porch configuration register (LTDC_BPCR)

This register defines the accumulated number of horizontal synchronization and back porch pixels minus 1 (**HSYNC width + HBP - 1)** and the accumulated number of vertical synchronization and back porch lines minus 1 (**VSYNC height + VBP - 1)**. Refer to *Figure 241* and *Section 31.4: LTDC programmable parameters* for an example of configuration.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AHBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	AVBP[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AHBP[11:0]**: accumulated horizontal back porch (in units of pixel clock period)

These bits define the accumulated horizontal back porch width that includes the horizontal synchronization and horizontal back porch pixels minus 1.

The horizontal back porch is the period between horizontal synchronization going inactive and the start of the active display part of the next scan line.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **AVBP[10:0]:** accumulated Vertical back porch (in units of horizontal scan line)

These bits define the accumulated vertical back porch width that includes the vertical synchronization and vertical back porch lines minus 1.

The vertical back porch is the number of horizontal scan lines at a start of frame to the start of the first active scan line of the next frame.

31.7.3 LTDC active width configuration register (LTDC_AWCR)

This register defines the accumulated number of horizontal synchronization, back porch and active pixels minus 1 (**HSYNC width + HBP + active width - 1)** and the accumulated number of vertical synchronization, back porch lines and active lines minus 1 (**VSYNC height + BVBP + active height - 1)**. Refer to *Figure 241* and *Section 31.4: LTDC programmable parameters* for an example of configuration.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AAW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	AAH[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AAW[11:0]**: accumulated active width (in units of pixel clock period)

These bits define the accumulated active width which includes the horizontal synchronization, horizontal back porch and active pixels minus 1.

The active width is the number of pixels in active display area of the panel scan line.

Refer to device datasheet for maximum active width supported following maximum pixel clock.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **AAH[10:0]**: accumulated active height (in units of horizontal scan line)

These bits define the accumulated height which includes the vertical synchronization, vertical back porch and the active height lines minus 1. The active height is the number of active lines in the panel.

Refer to device datasheet for maximum active height supported following maximum pixel clock.

31.7.4 LTDC total width configuration register (LTDC_TWCR)

This register defines the accumulated number of horizontal synchronization, back porch, active and front porch pixels minus 1 (**HSYNC width + HBP + active width + HFP - 1)** and the accumulated number of vertical synchronization, back porch lines, active and front lines minus 1 (**VSYNC height + BVBP + active height + VFP - 1)**. Refer to *Figure 241* and *Section 31.4: LTDC programmable parameters* for an example of configuration.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TOTALW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TOTALH[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **TOTALW[11:0]**: total width (in units of pixel clock period)

These bits defines the accumulated total width which includes the horizontal synchronization, horizontal back porch, active width and horizontal front porch pixels minus 1.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **TOTALH[10:0]**: total height (in units of horizontal scan line)

These bits defines the accumulated height which includes the vertical synchronization, vertical back porch, the active height and vertical front porch height lines minus 1.

31.7.5 LTDC global control register (LTDC_GCR)

This register defines the global configuration of the LCD-TFT controller.

Address offset: 0x18

Reset value: 0x0000 2220

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN
rw	rw	rw	rw												rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	Res.	LTDCEN
	r	r	r		r	r	r		r	r	r				rw

Bit 31 **HSPOL**: horizontal synchronization polarity
This bit is set and cleared by software.
0: horizontal synchronization polarity is active low.
1: horizontal synchronization polarity is active high.

Bit 30 **VSPOL**: vertical synchronization polarity
This bit is set and cleared by software.
0: vertical synchronization is active low.
1: vertical synchronization is active high.

Bit 29 **DEPOL**: not data enable polarity
This bit is set and cleared by software.
0: not data enable polarity is active low.
1: not data enable polarity is active high.

Bit 28 **PCPOL**: pixel clock polarity
This bit is set and cleared by software.
0: pixel clock polarity is active low.
1: pixel clock is active high.

Bits 27:17 Reserved, must be kept at reset value.

Bit 16 **DEN**: dither enable
This bit is set and cleared by software.
0: dither disable
1: dither enable

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DRW[2:0]**: dither red width
These bits return the Dither Red Bits.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DGW[2:0]**: dither green width
These bits return the dither green bits.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DBW[2:0]**: dither blue width
These bits return the dither blue bits.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **LTDCEN**: LCD-TFT controller enable
This bit is set and cleared by software.
0: LTDC disable
1: LTDC enable

31.7.6 LTDC shadow reload configuration register (LTDC_SRCR)

This register allows to reload either immediately or during the vertical blanking period, the shadow registers values to the active registers. The shadow registers are all Layer1 and Layer2 registers except the LTDC_L1CLUTWR and the LTDC_L2CLUTWR.

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VBR**: vertical blanking reload

This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).

0: no effect

1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: immediate reload

This bit is set by software and cleared only by hardware after reload.

0: no effect

1: The shadow registers are reloaded immediately.

Note: The shadow registers read back the active values. Until the reload has been done, the 'old' value is read.

31.7.7 LTDC background color configuration register (LTDC_BCCR)

This register defines the background color (RGB888).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCGREEN[7:0]								BCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **BCRED[7:0]**: background color red value
These bits configure the background red value.

Bits 15:8 **BCGREEN[7:0]**: background color green value
These bits configure the background green value.

Bits 7:0 **BCBLUE[7:0]**: background color blue value
These bits configure the background blue value.

31.7.8 LTDC interrupt enable register (LTDC_IER)

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIE	TERRIE	FUIE	LIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RRIE**: register reload interrupt enable
This bit is set and cleared by software.
0: register reload interrupt disable
1: register reload interrupt enable

Bit 2 **TERRIE**: transfer error interrupt enable
This bit is set and cleared by software.
0: transfer error interrupt disable
1: transfer error interrupt enable

Bit 1 **FUIE**: FIFO underrun interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disable
1: FIFO underrun Interrupt enable

Bit 0 **LIE**: line interrupt enable
This bit is set and cleared by software.
0: line interrupt disable
1: line interrupt enable

31.7.9 LTDC interrupt status register (LTDC_ISR)

This register returns the interrupt status flag.

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIF	TERRIF	FUIF	LIF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RRIF**: register reload interrupt flag

0: no register reload interrupt generated

1: register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

Bit 2 **TERRIF**: transfer error interrupt flag

0: no transfer error interrupt generated

1: transfer error interrupt generated when a bus error occurs

Bit 1 **FUIF**: FIFO underrun interrupt flag

0: no FIFO underrun interrupt generated.

1: FIFO underrun interrupt generated, if one of the layer FIFOs is empty and pixel data is read from the FIFO

Bit 0 **LIF**: line interrupt flag

0: no line interrupt generated

1: line interrupt generated when a programmed line is reached

31.7.10 LTDC Interrupt Clear Register (LTDC_ICR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRRIF	CTERRIF	CFUIF	CLIF
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CRRIF**: clears register reload interrupt flag
0: no effect
1: clears the RRIF flag in the LTDC_ISR register

Bit 2 **CTERRIF**: clears the transfer error interrupt flag
0: no effect
1: clears the TERRIF flag in the LTDC_ISR register.

Bit 1 **CFUIF**: clears the FIFO underrun interrupt flag
0: no effect
1: clears the FUDERRIF flag in the LTDC_ISR register.

Bit 0 **CLIF**: clears the line interrupt flag
0: no effect
1: clears the LIF flag in the LTDC_ISR register.

31.7.11 LTDC line interrupt position configuration register (LTDC_LIPCR)

This register defines the position of the line interrupt. The line value to be programmed depends on the timings parameters. Refer to *Figure 241*.

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LIPOS[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **LIPOS[10:0]**: line interrupt position
These bits configure the line interrupt position.

31.7.12 LTDC current position status register (LTDC_CPSR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CXPOS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYPOS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CXPOS[15:0]**: current X position
These bits return the current X position.

Bits 15:0 **CYPOS[15:0]**: current Y position
These bits return the current Y position.

31.7.13 LTDC current display status register (LTDC_CDSR)

This register returns the status of the current display phase which is controlled by the HSYNC, VSYNC, and horizontal/vertical DE signals.

Example: if the current display phase is the vertical synchronization, the VSYNCS bit is set (active high). If the current display phase is the horizontal synchronization, the HSYNCS bit is active high.

Address offset: 0x48

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSYNCS	VSYNCS	HDES	VDES
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **HSYNCS**: horizontal synchronization display status
0: active low
1: active high

Bit 2 **VSYNCS**: vertical synchronization display status
0: active low
1: active high

Bit 1 **HDES**: horizontal data enable display status
0: active low
1: active high

Bit 0 **VDES**: vertical data enable display status
0: active low
1: active high

*Note: The returned status does not depend on the configured polarity in the **LTDC_GCR** register, instead it returns the current active display phase.*

31.7.14 LTDC layer x control register (LTDC_LxCR)

Address offset: 0x84 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLUTEN	Res.	Res.	COLKEN	LEN
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CLUTEN**: color look-up table enable
This bit is set and cleared by software.
0: color look-up table disable
1: color look-up table enable
The CLUT is only meaningful for L8, AL44 and AL88 pixel format. Refer to *Color look-up table (CLUT)*

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **COLKEN**: color keying enable
This bit is set and cleared by software.
0: color keying disable
1: color keying enable

Bit 0 **LEN**: layer enable
This bit is set and cleared by software.
0: layer disable
1: layer enable

31.7.15 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR)

This register defines the horizontal position (first and last pixel) of the layer 1 or 2 window.

The first visible pixel of a line is the programmed value of AHBP[11:0] bits + 1 in the LTDC_BPCR register.

The last visible pixel of a line is the programmed value of AAW[10:0] bits in the LTDC_AWCR register.

Address offset: 0x88 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WHSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WHSPPOS[11:0]**: window horizontal stop position
These bits configure the last visible pixel of a line of the layer window.
WHSPPOS[11:0] must be ≥ ***AHBP[11:0] bits + 1*** (programmed in LTDC_BPCR register).

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WHSTPOS[11:0]**: window horizontal start position
These bits configure the first visible pixel of a line of the layer window.
WHSTPOS[11:0] must be ≤ ***AAW[11:0] bits*** (programmed in LTDC_AWCR register).

Example:

The LTDC_BPCR register is configured to 0x000E0005 (AHBP[11:0] is 0xE) and the LTDC_AWCR register is configured to 0x028E01E5 (AAW[11:0] is 0x28E). To configure the horizontal position of a window size of 630x460, with horizontal start offset of 5 pixels in the active data area:

1. layer window first pixel, WHSTPOS[11:0], must be programmed to 0x14 (0xE+1+0x5)
2. layer window last pixel, WHSPPOS[11:0], must be programmed to 0x28A.

31.7.16 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR)

This register defines the vertical position (first and last line) of the layer1 or 2 window.

The first visible line of a frame is the programmed value of AVBP[10:0] bits + 1 in the register LTDC_BPCR register.

The last visible line of a frame is the programmed value of AAH[10:0] bits in the LTDC_AWCR register.

Address offset: 0x8C + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	WVSPPOS[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	WVSTPOS[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **WVSPPOS[10:0]**: window vertical stop position
These bits configure the last visible line of the layer window.
WVSPPOS[10:0] must be ≥ ***AVBP[10:0] bits + 1*** (programmed in LTDC_BPCR register).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **WVSTPOS[10:0]**: window vertical start position
These bits configure the first visible line of the layer window.
WVSTPOS[10:0] must be ≤ ***AAH[10:0] bits*** (programmed in LTDC_AWCR register).

Example:

The LTDC_BPCR register is configured to 0x000E0005 (AVBP[10:0] is 0x5) and the LTDC_AWCR register is configured to 0x028E01E5 (AAH[10:0] is 0x1E5).

To configure the vertical position of a window size of 630x460, with vertical start offset of 8 lines in the active data area:

1. layer window first line: WVSTPOS[10:0] must be programmed to 0xE (0x5 + 1 + 0x8).
2. layer window last line: WVSTPOS[10:0] must be programmed to 0x1DA.

31.7.17 LTDC layer x color keying configuration register (LTDC_LxCKCR)

This register defines the color key value (RGB), that is used by the color keying.

Address offset: 0x90 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CKGREEN[7:0]								CKBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CKRED[7:0]**: color key red value

Bits 15:8 **CKGREEN[7:0]**: color key green value

Bits 7:0 **CKBLUE[7:0]**: color key blue value

31.7.18 LTDC layer x pixel format configuration register (LTDC_LxPFCR)

This register defines the pixel format that is used for the stored data in the frame buffer of a layer. The pixel data is read from the frame buffer and then transformed to the internal format 8888 (ARGB).

Address offset: 0x94 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PF[2:0]**: pixel format

These bits configure the pixel format
000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
101: L8 (8-bit luminance)
110: AL44 (4-bit alpha, 4-bit luminance)
111: AL88 (8-bit alpha, 8-bit luminance)

31.7.19 LTDC layer x constant alpha configuration register (LTDC_LxCACR)

This register defines the constant alpha value (divided by 255 by hardware), that is used in the alpha blending. Refer to LTDC_LxBFCR register.

Address offset: 0x98 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 00FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CONSTA[7:0]**: constant alpha

These bits configure the constant alpha used for blending. The constant alpha is divided by 255 by hardware.
Example: if the programmed constant alpha is 0xFF, the constant alpha value is 255 / 255 = 1.

31.7.20 LTDC layer x default color configuration register (LTDC_LxDCCR)

This register defines the default color of a layer in the format ARGB. The default color is used outside the defined layer window or when a layer is disabled. The reset value of 0x00000000 defines a transparent black color.

Address offset: 0x9C + 0x80 * (x - 1), (*x* = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCALPHA[7:0]								DCRED[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCGREEN[7:0]								DCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DCALPHA[7:0]**: default color alpha
These bits configure the default alpha value.

Bits 23:16 **DCRED[7:0]**: default color red
These bits configure the default red value.

Bits 15:8 **DCGREEN[7:0]**: default color green
These bits configure the default green value.

Bits 7:0 **DCBLUE[7:0]**: default color blue
These bits configure the default blue value.

31.7.21 LTDC layer x blending factors configuration register (LTDC_LxBFCR)

This register defines the blending factors F1 and F2.

The general blending formula is: BC = BF1 x C + BF2 x Cs

- BC = blended color
- BF1 = blend factor 1
- C = current layer color
- BF2 = blend factor 2
- Cs = subjacent layers blended color

Address offset: 0xA0 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0607

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
					rw	rw	rw						rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **BF1[2:0]**: blending factor 1

These bits select the blending factor F1.

000: reserved
001: reserved
010: reserved
011: reserved
100: constant alpha
101: reserved
110: pixel alpha x constant alpha
111: reserved

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **BF2[2:0]**: blending factor 2

These bits select the blending factor F2

000: reserved
001: reserved
010: reserved
011: reserved
100: reserved
101: 1 - constant alpha
110: reserved
111: 1 - (pixel alpha x constant alpha)

Note: The constant alpha value, is the programmed value in the LxCACR register divided by 255 by hardware.

Example: Only layer1 is enabled, BF1 configured to constant alpha. BF2 configured to 1 - constant alpha. The constant alpha programmed in the LxCACR register is 240 (0xF0). Thus, the constant alpha value is 240/255 = 0.94. C: current layer color is 128. Cs: background color is 48. Layer1 is blended with the background color. BC = constant alpha x C + (1 - Constant Alpha) x Cs = 0.94 x 128 + (1- 0.94) x 48 = 123.

31.7.22 LTDC layer x color frame buffer address register (LTDC_LxCFBAR)

This register defines the color frame buffer start address which has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer.

Address offset: 0xAC + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFBADD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFBADD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFBADD[31:0]**: color frame buffer start address
These bits define the color frame buffer start address.

31.7.23 LTDC layer x color frame buffer length register (LTDC_LxCFBLR)

This register defines the color frame buffer line length and pitch.

Address offset: 0xB0 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CFBP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CFBLL[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **CFBP[12:0]**: color frame buffer pitch in bytes
These bits define the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **CFBLL[12:0]**: color frame buffer line length

These bits define the length of one line of pixels in bytes + 7.
The line length is computed as follows:
active high width * number of bytes per pixel + 7.

Example:

- A frame buffer having the format RGB565 (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format RGB888 (3 bytes per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 3 = 960), where pitch = line length requires a value of 0x03C003C7 to be written into this register.

31.7.24 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR)

This register defines the number of lines in the color frame buffer.

Address offset: 0xB4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CFBLNBR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **CFBLNBR[10:0]**: frame buffer line number

These bits define the number of lines in the frame buffer that corresponds to the active high width.

Note: The number of lines and line length settings define how much data is fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt will be generated if enabled.

The start address and pitch settings on the other hand define the correct start of every line in memory.

31.7.25 LTDC layer x CLUT write register (LTDC_LxCLUTWR)

This register defines the CLUT address and the RGB value.

Address offset: 0xC4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLUTADD[7:0]								RED[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **CLUTADD[7:0]**: CLUT address
These bits configure the CLUT address (color position within the CLUT) of each RGB value.

Bits 23:16 **RED[7:0]**: red value
These bits configure the red value.

Bits 15:8 **GREEN[7:0]**: green value
These bits configure the green value.

Bits 7:0 **BLUE[7:0]**: blue value
These bits configure the blue value.

Note: The CLUT write register should only be configured during blanking period or if the layer is disabled. The CLUT can be enabled or disabled in the LTDC_LxCR register.

The CLUT is only meaningful for L8, AL44 and AL88 pixel format.

31.7.26 LTDC register map

The following table summarizes the LTDC registers. Refer to the register boundary addresses table for the LTDC register base address.

Table 265. LTDC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0008	LTDC_SSCR	Res.	Res.	Res.	Res.	HSW[11:0]												Res.	Res.	Res.	Res.	Res.	VSH[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x000C	LTDC_BPCR	Res.	Res.	Res.	Res.	AHBP[11:0]												Res.	Res.	Res.	Res.	Res.	AVBP[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0010	LTDC_AWCR	Res.	Res.	Res.	Res.	AAW[11:0]												Res.	Res.	Res.	Res.	Res.	AAH[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0014	LTDC_TWCR	Res.	Res.	Res.	Res	TOTALW[11:0]												Res	Res	Res	Res	Res	TOTALH[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0018	LTDC_GCR	HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN	Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	Res.	LTDCEN
	Reset value	0	0	0	0												0		0	1	0		0	1	0		0	1	0				0
0x0024	LTDC_SRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
	Reset value																															0	0
0x002C	LTDC_BCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]								BCGREEN[7:0]								BCBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	LTDC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIE	TERRIE	FUIE	LIE
	Reset value																													0	0	0	0
0x0038	LTDC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIF	TERRIF	FUIF	LIF
	Reset value																													0	0	0	0
0x003C	LTDC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRRIF	CTERRIF	CFUIF	CLIF
	Reset value																													0	0	0	0
0x0040	LTDC_LIPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIPOS10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x0044	LTDC_CPSR	CXPOS[15:0]																CYPOS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 265. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0048	LTDC_CDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		HSYNCS	VSYNCS	HDES	VDES
	Reset value																													1	1	1	1
0x0084	LTDC_L1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLUTEN	Res.	Res.	COLKEN	LEN
	Reset value																												0			0	0
0x0088	LTDC_L1WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x008C	LTDC_L1WVPCR	Res.	Res.	Res.	Res.	Res.	WVSPPOS[10:0]											Res.	Res.	Res.	Res.	Res.	WVSTPOS[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0090	LTDC_L1CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094	LTDC_L1PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x0098	LTDC_L1CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x009C	LTDC_L1DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	LTDC_L1BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value																						1	1	0						1	1	1
0x00AC	LTDC_L1CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B0	LTDC_L1CFBLR	Res.	Res.	Res.	CFBP[12:0]													Res.	Res.	Res.	CFBLL[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B4	LTDC_L1CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x00C4	LTDC_L1CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0104	LTDC_L2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLUTEN	Res.	Res.	COLKEN	LEN
	Reset value																												0			0	0

Table 265. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0108	LTDC_L2WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x010C	LTDC_L2WVPCR	Res.	Res.	Res.	Res.	Res.	WVSPPOS[10:0]											Res.	Res.	Res.	Res.	Res.	WVSTPOS[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0110	LTDC_L2CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114	LTDC_L2PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x0118	LTDC_L2CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x011C	LTDC_L2DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0120	LTDC_L2BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value																						1	1	0						1	1	1
0x012C	LTDC_L2CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0130	LTDC_L2CFBLR	Res.	Res.	Res.	CFBP[12:0]													Res.	Res.	Res.	CFBLL[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134	LTDC_L2CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x0144	LTDC_L2CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2* for the register boundary addresses.

32 JPEG codec (JPEG)

32.1 Introduction

The hardware 8-bit JPEG codec encodes uncompressed image data stream or decodes JPEG-compressed image data stream. It also fully manages JPEG headers.

32.2 JPEG codec main features

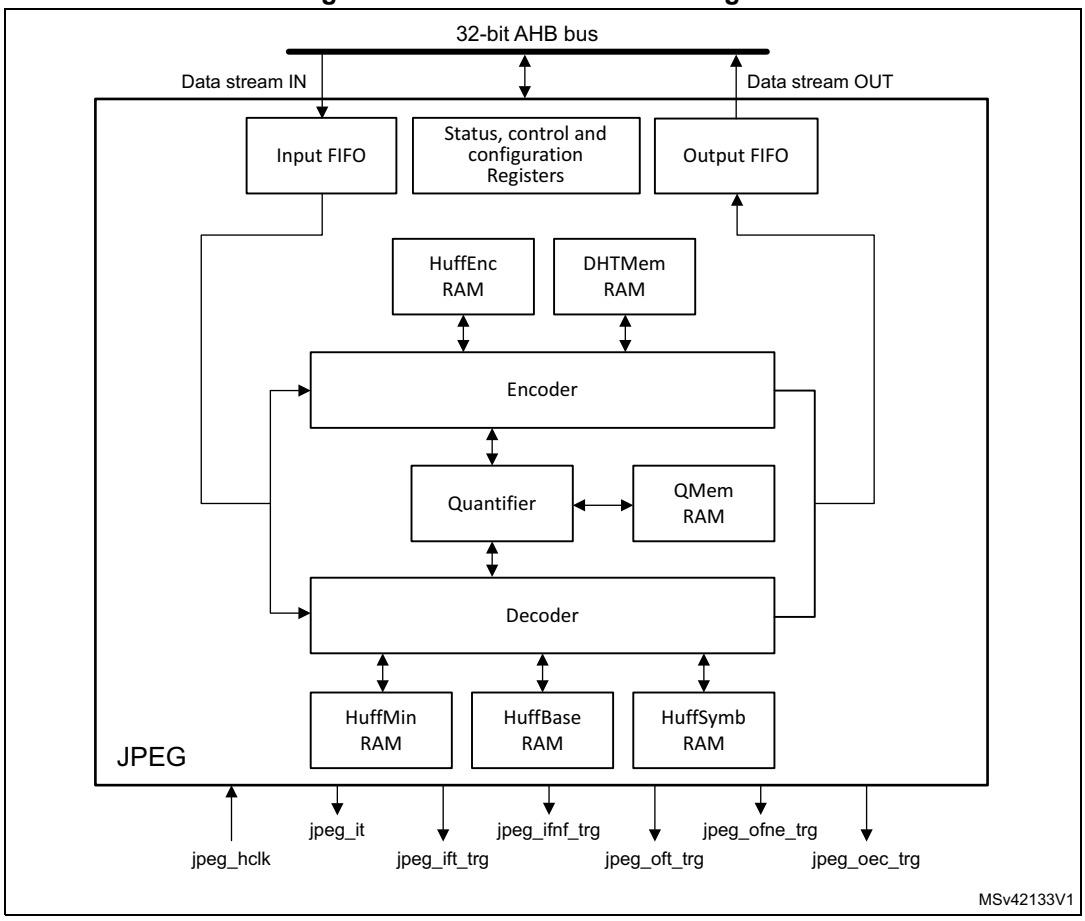
- High-speed fully-synchronous operation
- Configurable as encoder or decoder
- Single-clock-per-pixel encode/decode
- RGB, YCbCr, YCMK and BW (grayscale) image color space support
- 8-bit depth per image component at encode/decode
- JPEG header generator/parser with enable/disable
- Four programmable quantization tables
- Single-clock Huffman coding and decoding
- Fully-programmable Huffman tables (two AC and two DC)
- Fully-programmable minimum coded unit (MCU)
- Concurrent input and output data stream interfaces

32.3 JPEG codec block functional description

32.3.1 General description

The block diagram of the JPEG codec is shown in *Figure 245: JPEG codec block diagram*.

Figure 245. JPEG codec block diagram



32.3.2 JPEG internal signals

Table 266 lists the JPEG internal signals.

Table 266. JPEG internal signals

Signal name	Signal type	Description
jpeg_hclk	Digital input	JPEG kernel and register interface clock
jpeg_it	Digital output	JPEG global interrupt
jpeg_ift_trg	Digital output	JPEG input FIFO threshold for MDMA
jpeg_ifnf_trg	Digital output	JPEG input FIFO not full for MDMA
jpeg_oft_trg	Digital output	JPEG output FIFO threshold for MDMA

Table 266. JPEG internal signals (continued)

Signal name	Signal type	Description
jpeg_ofne_trg	Digital output	JPEG output FIFO not empty for MDMA
jpeg_oec_trg	Digital output	JPEG end of conversion for MDMA

32.3.3 JPEG decoding procedure

The JPEG codec can decode a JPEG stream as defined in the *ISO/IEC 10918-1* specification.

It can optionally parse the JPEG header and update accordingly the JPEG codec registers, the quantization tables and the Huffman tables.

The JPEG codec is configured in decode mode setting the DE bit (decode enable) of the JPEG_CONFR1 register.

The JPEG decode starts by setting the START bit of the JPEG_CONFR0 register.

The JPEG codec requests data for its input FIFO through generating one of:

- MDMA trigger
- interrupts

Interrupt or MDMA trigger generation for input FIFO

Input FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:

- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or MDMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the decoding process, extra bytes may remain in the input FIFO and/or an interrupt request / MDMA trigger may be pending. The FIFO can be flushed by setting the IFF bit (Input FIFO Flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the input FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The consequence of not flushing the FIFO at the end of the decoding process is that any remaining data is taken into the next JPEG decoding.

Header parsing

The header parsing can be activated setting the HDR bit of the JPEG_CONFR1 register.

The JPEG header parser supports all markers relevant to the JPEG baseline algorithm indicated in *Annex B* of the *ISO/IEC 10918-1*.

When parsing a supported marker, the JPEG header parser extracts the required parameters and stores them in shadow registers. At the end of the parsing the JPEG codec registers are updated.

If a DQT marker segment is located, quantization data associated with it is written into the quantization table memory.

If a DHT marker segment is located, the Huffman table data associated with it is converted into three different table formats (HuffMin, HuffBase and HuffSymb) and stored in their respective memories.

Once the parsing operation is completed, the HPDF (header parsing done flag) bit of the JPEG_SR register is set. An interrupt is generated if the EHPIE (end of header parsing interrupt enable) bit of the JPEG_CR register is set.

JPEG decoding

Once the JPEG header is parsed or JPEG codec registers and memories are properly programmed, the incoming data stream is decoded and the resulting MCUs are sent to the output FIFO.

When decoding two images successively, the START bit of the JPEG_CONFR0 register must be set again (even if already 1) after the header processing of the second image is completed.

Interrupt or MDMA trigger generation for output FIFO

The output FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:

- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO Threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by reseting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. If the FIFO needs to be flushed, it shall be done by software setting the FF bit (FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the output FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

32.3.4 JPEG encoding procedure

The JPEG codec can encode a JPEG stream as defined in the *ISO/IEC 10918-1* specification.

It can optionally generate the JPEG Header.

The JPEG codec is configured in encode mode resetting the DE bit (decode enable) of the JPEG_CONFR1 register.

The configuration used for encoding the JPEG must be loaded in the JPEG codec:

- JPEG codec configuration registers
- quantization tables
- Huffman tables

The JPEG codec is started setting the START bit of the JPEG_CONFR0 register.

Once the JPEG codec has been started, it request data for its input FIFO generating one of:

- MDMA trigger
- interrupts

Interrupt or MDMA trigger generation for input FIFO

Input FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:

- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or MDMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the encoding process, extra bytes may remain in the input FIFO and/or an interrupt request / MDMA trigger may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the input FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The consequence of not flushing the FIFO at the end of the encoding process is that any remaining data is taken into the next JPEG encoding.

JPEG encoding

Once the JPEG header generated, the incoming MCUs are encoded and the resulting data stream sent to the output FIFO.

Interrupt or MDMA trigger generation for output FIFO

Output FIFO can be managed using interrupts or MDMA triggers through two flags according to the FIFO state:

- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by reseting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. The FIFO can be flushed by setting the FF bit (FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the output FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The MDMA channel must be stopped to prevent unwanted MDMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

The EOCF bit (end of conversion flag) of the JPEG_SR register can only be cleared when the output FIFO is empty.

Clearing either of the HDR bit (header processing) of the JPEG_CONFR1 register and the JCEN bit (JPEG codec enable) of the JPEG_CR register is allowed only when the EOCF bit of the JPEG_SR register is cleared.

32.4 JPEG codec interrupts

An interrupt can be produced on the following events:

- input FIFO threshold reached
- input FIFO not full
- output FIFO threshold reached
- output FIFO not empty
- end of conversion
- header parsing done

Separate interrupt enable bits are available for flexibility.

Table 267. JPEG codec interrupt requests

Interrupt event	Event flag	Enable Control bit
Input FIFO threshold reached	IFTF	IFTIE
Input FIFO not full	IFNFF	IFNFIE
Output FIFO threshold reached	OFTF	OFTIE
Output FIFO not empty	OFNEF	OFNEIE
End of conversion	EOCF	EOCIE
Header parsing done	HPDF	HPDIE

32.5 JPEG codec registers

32.5.1 JPEG codec control register (JPEG_CONFR0)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START
															w

Bits 31: 1 Reserved

Bit 0 **START**: Start

This bit start or stop the encoding or decoding process.

0: Stop/abort

1: Start

Reads always return 0.

32.5.2 JPEG codec configuration register 1 (JPEG_CONFR1)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
YSIZE[15:0]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDR	NS[1:0]		COLSPACE[1:0]		DE	Res.	NF[1:0]	

Bits 31: 16 **YSIZE[15:0]**: Y Size

This field defines the number of lines in source image.

Bits 15: 9 Reserved

Bit 8 **HDR**: Header processing

This bit enables the header processing (generation/parsing).

0: Disable

1: Enable

Bits 7: 6 **NS[1:0]**: Number of components for scan

This field defines the number of components minus 1 for scan header marker segment.

Bits 5: 4 **COLORSPACE[1:0]**: Color space
This filed defines the number of quantization tables minus 1 to insert in the output stream.
00: Grayscale (1 quantization table)
01: YUV (2 quantization tables)
10: RGB (3 quantization tables)
11: CMYK (4 quantization tables)

Bit 3 **DE**: Codec operation as coder or decoder
This bit selects the code or decode process
0: Code
1: Decode

Bit 2 Reserved

Bits 1: 0 **NF[1:0]**: Number of color components
This field defines the number of color components minus 1.
00: Grayscale (1 color component)
01: - (2 color components)
10: YUV or RGB (3 color components)
11: CMYK (4 color components)

32.5.3 JPEG codec configuration register 2 (JPEG_CONFR2)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	NMCU[25:16]									
						rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMCU[15:0]															
rw															

Bits 31: 26 Reserved

Bits 25: 0 **NMCU[25:0]**: Number of MCUs
For encoding: this field defines the number of MCU units minus 1 to encode.
For decoding: this field indicates the number of complete MCU units minus 1 to be decoded (this field is updated after the JPEG header parsing). If the decoded image size has not a X or Y size multiple of 8 or 16 (depending on the sub-sampling process), the resulting incomplete or empty MCU must be added to this value to get the total number of MCUs generated.

32.5.4 JPEG codec configuration register 3 (JPEG_CONFR3)

Address offset: 0x000C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XSIZE[15:0]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31: 16 **XSIZE[15:0]**: X size
This field defines the number of pixels per line.

Bits 15: 0 Reserved

32.5.5 JPEG codec configuration register 4-7 (JPEG_CONFR4-7)

Address offset: 0x0010 + 0x4 * i, where “i” is image component from 0 to 3

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
rw				rw				rw				rw		rw	rw

Bits 31: 16 Reserved

Bits 15: 12 **HSF[3:0]**: Horizontal sampling factor
Horizontal sampling factor for component i.

Bits 11: 8 **VSF[3:0]**: Vertical sampling factor
Vertical sampling factor for component i.

Bits 7: 4 **NB[3:0]**: Number of blocks
Number of data units minus 1 that belong to a particular color in the MCU.

Bits 3: 2 **QT[1:0]**: Quantization table
Selects quantization table used for component i.
00: Quantization table 0
01: Quantization table 1
10: Quantization table 2
11: Quantization table 3

Bit 1 **HA**: Huffman AC
Selects the Huffman table for encoding AC coefficients.
0: Not selected
1: Selected

Bit 0 **HD**: Huffman DC
Selects the Huffman table for encoding DC coefficients.
0: Not selected
1: Selected

32.5.6 JPEG control register (JPEG_CR)

Address offset: 0x0030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OFF	IFF	Res.	Res.	Res.	Res.	Res.	Res.	HPDIE	EOCIE	OFNEIE	OFTIE	IFNFIE	IFTIE	JCEN
	r0	r0							rw	rw	rw	rw	rw	rw	rw

Bits 31: 15 Reserved

Bit 14 **OFF**: Output FIFO flush
This bit flushes the output FIFO.
0: No effect
1: Output FIFO is flushed
Reads always return 0.

Bit 13 **IFF**: Input FIFO flush
This bit flushes the input FIFO.
0: No effect
1: Input FIFO is flushed
Reads always return 0.

Bits 12: 7 Reserved

Bit 6 **HPDIE**: Header parsing done interrupt enable
This bit enables interrupt generation upon the completion of the header parsing operation.
0: Disabled
1: Enabled

Bit 5 **EOCIE**: End of conversion interrupt enable
This bit enables interrupt generation at the end of conversion.
0: Disabled
1: Enabled

Bit 4 **OFNEIE**: Output FIFO not empty interrupt enable
This bit enables interrupt generation when the output FIFO is not empty.
0: Disabled
1: Enabled

Bit 3 **OFTIE**: Output FIFO threshold interrupt enable
This bit enables interrupt generation when the output FIFO reaches a threshold.
0: Disabled
1: Enabled

Bit 2 **IFNFIE**: Input FIFO not full interrupt enable
This bit enables interrupt generation when the input FIFO is not empty.
0: Disabled
1: Enabled

Bit 1 **IFTIE**: Input FIFO threshold interrupt enable
This bit enables interrupt generation when the input FIFO reaches a threshold.
0: Disabled
1: Enabled

Bit 0 **JCEN**: JPEG core enable
This bit enables the JPEG codec core.
0: Disabled (internal registers are reset).
1: Enabled (internal registers are accessible).

32.5.7 JPEG status register (JPEG_SR)

Address offset: 0x0034

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF	HPDF	EOCF	OFNEF	OFTF	IFNFF	IFTF	Res.
								ro	ro	ro	ro	ro	ro	ro	

Bits 31: 8 Reserved

Bit 7 **COF**: Codec operation flag
This bit flags code/decode operation in progress.
0: Not in progress
1: In progress

Bit 6 **HPDF**: Header parsing done flag
In decode mode, this bit flags the completion of header parsing and updating internal registers.
0: Not completed
1: Completed

Bit 5 **EOCF**: End of conversion flag
This bit flags the completion of encode/decode process and data transfer to the output FIFO.
0: Not completed
1: Completed

Bit 4 **OFNEF**: Output FIFO not empty flag
This bit flags that data is available in the output FIFO.
0: Empty (data not available)
1: Not empty (data available)

Bit 4 **OFTF**: Output FIFO threshold flag
This bit flags that the amount of data in the output FIFO reaches or exceeds a threshold.
0: Below threshold
1: At or above threshold

Bit 2 **IFNFF**: Input FIFO not full flag
This bit flags that the input FIFO is not full (data can be written).
0: Full
1: Not full

Bit 1 **IFTF**: Input FIFO threshold flag
This bit flags that the amount of data in the input FIFO is below a threshold.
0: At or above threshold
1: Below threshold.

Bit 0 Reserved

32.5.8 JPEG clear flag register (JPEG_CFR)

Address offset: 0x0038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHPDF	CEOCF	Res.	Res.	Res.	Res.	Res.
									w1c	w1c	Res.	Res.	Res.	Res.	Res.

Bits 31: 7 Reserved

Bit 6 **CHPDF**: Clear header parsing done flag
Writing 1 clears the HPDF bit of the JPEG_SR register.
0: No effect
1: Clear

Bit 5 **CEOCF**: Clear end of conversion flag
Writing 1 clears the ECF bit of the JPEG_SR register.
0: No effect
1: Clear

Bits 4: 0 Reserved

32.5.9 JPEG data input register (JPEG_DIR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
wo															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
wo															

Bits 31: 0 **DATAIN[31:0]**: Data input FIFO
Input FIFO data register.

32.5.10 JPEG data output register (JPEG_DOR)

Address offset: 0x0044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAOUT[31:16]															
ro															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUT[15:0]															
ro															

Bits 31: 0 **DATAOUT[31:0]**: Data output FIFO
Output FIFO data register.

32.5.11 JPEG codec register map

The following table summarizes the JPEG codec registers. Refer to the register boundary addresses table for the JPEG codec register base address.

Table 268. JPEG codec register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**JPEG_CONFR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START
	Reset value																																0
0x0004	**JPEG_CONFR1**	YSIZE[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDR	NS[1:0]		COLSPACE[1:0]		DE	Res.	NF[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0		0	0
0x0008	**JPEG_CONFR2**	Res.	Res.	Res.	Res.	Res.	Res.	NMCU[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	**JPEG_CONFR3**	XSIZE[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x0010	**JPEG_CONFR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	**JPEG_CONFR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	**JPEG_CONFR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	**JPEG_CONFR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020-0x002C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0030	**JPEG_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFF	IFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOCIE	OFNEIE	OFTIE	IFNFIE	IFTIE	JCEN
	Reset value																		0	0								0	0	0	0	0	0
0x0034	**JPEG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF	HPDF	EOCF	OFNEF	OFTF	IFNFF	IFTF	Res.
	Reset value																									0	0	0	0	0	1	1	
0x0038	**JPEG_CFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHPDF	CEOCF	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0					
0x0040	**JPEG_DIR**	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 268. JPEG codec register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0044	**JPEG_DOR**	DATAOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050-0x014C	**QMEM**	QMem RAM																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0150-0x018C	**HUFFMIN**	HuffMin RAM																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0190-0x020C	**HUFFBASE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HuffBase RAM									Res.	Res.	Res.	Res.	Res.	Res.	Res.	HuffBase RAM								
	Reset value								X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X
0x0210-0x035C	**HUFFSYMB**	HuffSymb RAM																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0360-0x04FC	**DHTMEM**	DHTMem RAM																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0500-0x07FC	**HUFFENC**	Res.	Res.	Res.	Res.	HuffEnc RAM												Res.	Res.	Res.	Res.	HuffEnc RAM											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

33 True random number generator (RNG)

33.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG can be used to construct a NIST compliant Deterministic Random Bit Generator (DRBG), acting as a live entropy source.

The RNG true random number generator has been validated according to the German AIS-31 standard.

33.2 RNG main features

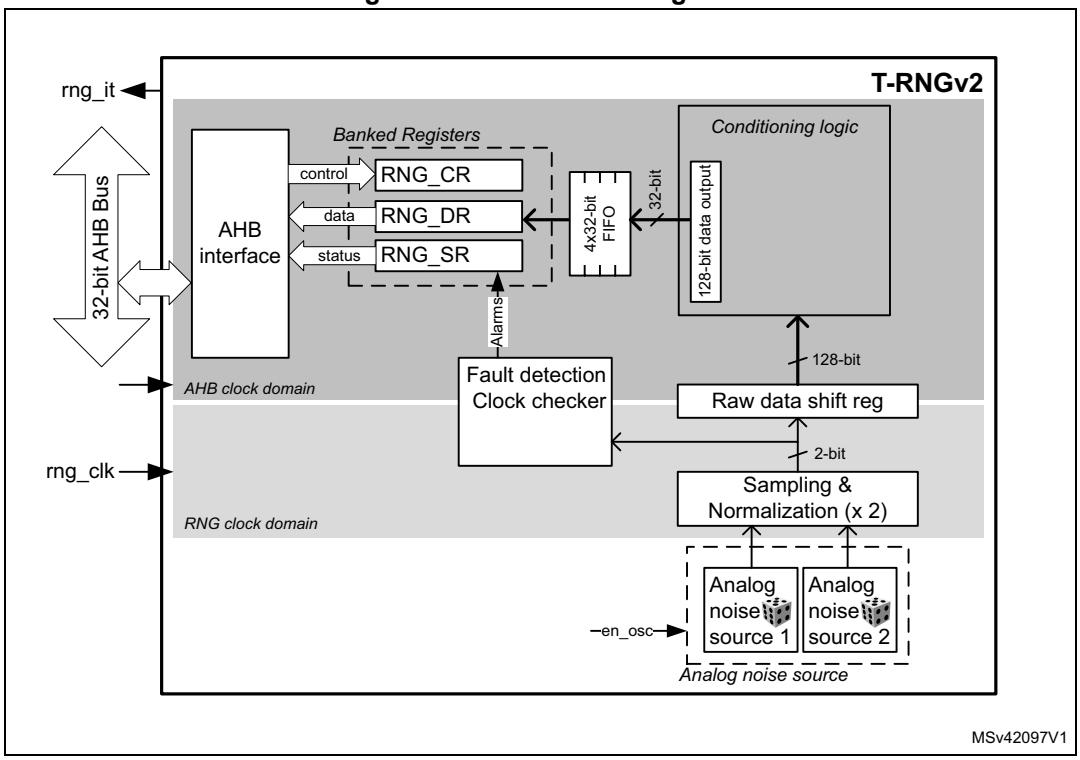
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- It is validated according to the AIS-31 pre-defined class PTG.2 evaluation methodology, which is part of the German Common Criteria (CC) scheme.
- It produces four 32-bit random samples every 16x $\frac{f_{AHB}}{f_{RNG}}$ AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management
  - Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- It has an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (else for write accesses an AHB bus error is generated), Warning! any write not equal to 32 bits might corrupt the register content.

33.3 RNG functional description

33.3.1 RNG block diagram

Figure 246 shows the RNG block diagram.

Figure 246. RNG block diagram



33.3.2 RNG internal signals

Table 269 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 269. RNG internal input/output signals

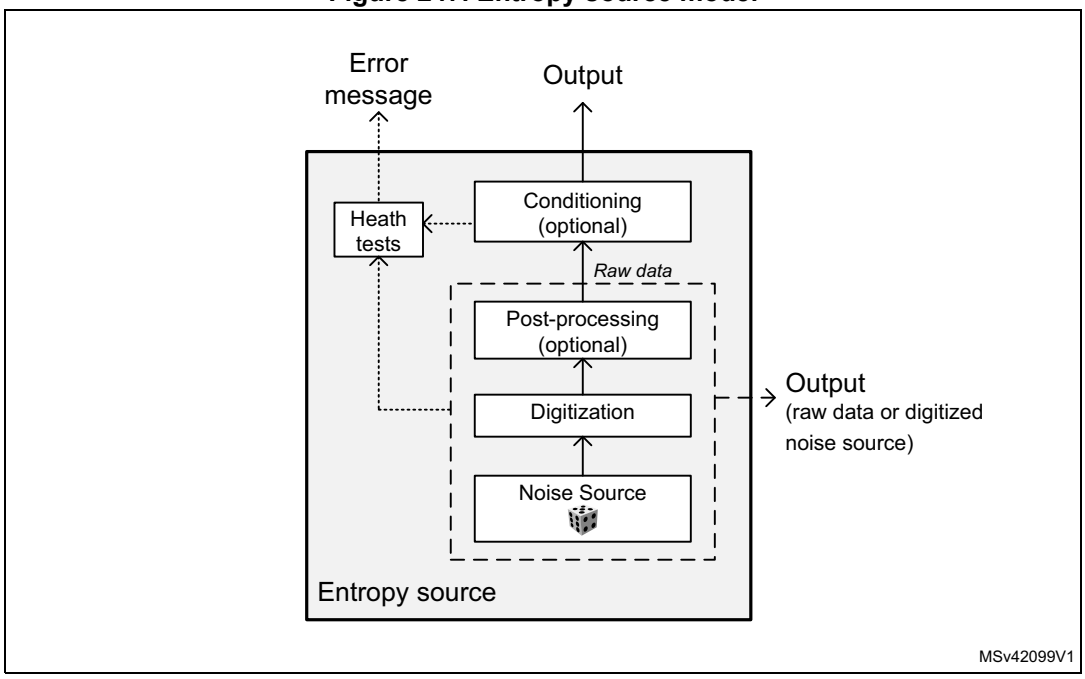
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk2	Digital input	AHB2 clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk2

33.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. The RNG implements the entropy source model pictured on *Figure 247*, and provides three main functions to the application:

- Collects the bitstring output of the entropy source box
- Obtains samples of the noise source for validation purpose
- Collects error messages from continuous health tests

Figure 247. Entropy source model



The main components of the RNG are:

- A source of physical randomness (analog noise sources)
- A digitization stage for those analog noise sources
- A stage delivering a cryptographically conditioned noise source
- Output buffers, for both entropy source output (buffered) and noise source samples (also buffered)
- A health monitoring block performing tests on the whole entropy source

All those components are detailed below.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. It is composed of:

- Two analog noise sources, each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 33.4: RNG low-power usage*.
- A sampling stage of these outputs clocked by a dedicated clock input (**rng_clk**), delivering a 2-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (**rng_hclk**).

Note: In Section 33.7: Entropy source validation recommended RNG clock frequencies are given.

Post processing

The sample values obtained from a true random noise source consist of 2-bit bitstrings. Because this noise source output is biased, the RNG implements a post-processing component that reduces that bias to a tolerable level.

More specifically, for each of the two noise source bits the RNG takes half of the bits from the sampled noise source, and half of the bits from inverted sampled noise source. Thus, if the source generates more ‘1’ than ‘0’ (or the opposite), it is filtered

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit).

Note: The latency during the RNG initialization is described in Section 33.6: RNG processing time.

Also note that post-processing computations are triggered when at least 32 bits of raw data has been received and when output FIFO needs a refill. Thus the RNG output entropy is maximum when the RNG 128-bit FIFO is emptied by application after 64 RNG clock cycles.

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 33.6: RNG processing time*.

The conditioning component is clocked by the faster AHB clock.

Output buffer

A data output buffer can store up to four 32-bit words which have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register 213 AHB clock cycles later.

Whenever a random number is available through the RNG_DR register the DRDY flag transitions from “0” to “1”. This flag remains high until output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from “0” to “1”. Interrupt is then cleared automatically by the RNG as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features:

1. Behavior tests, applied to the entropy source *at run-time*
   – Repetition count test, flagging an error when:
   a) One of the noise source has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bits patterns (“01” or “10”)
   b) Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bits patterns (“01” or “10”)
2. Vendor specific continuous test
   – Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle is smaller than AHB clock cycle divided by 32.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 33.3.7: Error management*.

Note: An interrupt can be generated when an error is detected.

33.3.4 RNG initialization

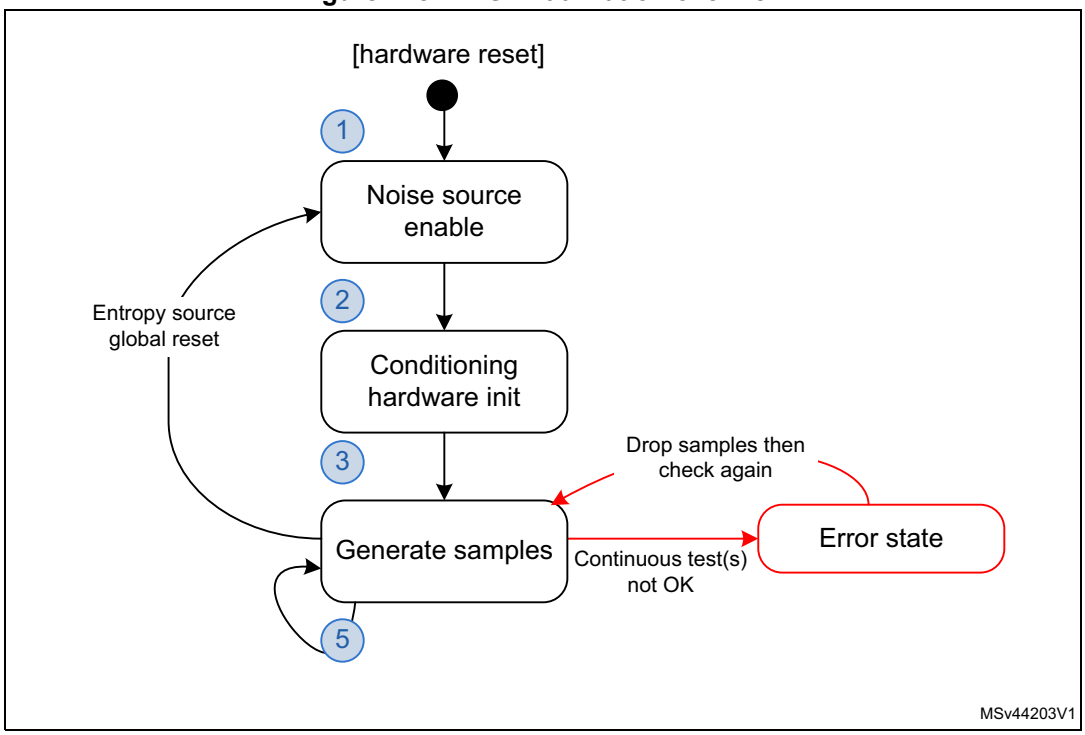
The RNG simplified state machine is pictured on *Figure 248*

When a hardware reset occurs the following chain of events occurs:

1. The analog noise source is enabled, and logic immediately starts sampling the analog output, filling 128-bit conditioning shift register
2. The conditioning logic is enabled and post-processing context is initialized using two 128 noise source bits.
3. The conditioning stage internal input data buffer is filled again with 128-bit and one conditioning round is performed. The output buffer is then filled with post processing result.
4. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 33.6: RNG processing time*.

Figure 248. RNG initialization overview



33.3.5 RNG operation

Normal operations

To run the RNG using interrupts the following steps are recommended:

1. Enable the interrupts by setting the IE bit in the RNG_CR register. At the same time enable the RNG by setting the bit RNGEN=1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits should be set to ‘0’ in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to ‘1’ in the RNG_SR register.
   - If above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit will still be high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in *Section 33.3.7* shall be used.

To run the RNG in polling mode following steps are recommended:

1. Enable the random number generation by setting the RNGEN bit to “1” in the RNG_CR register.
2. Read the RNG_SR register and check that:
   – No error occurred (the SEIS and CEIS bits should be set to ‘0’)
   – A random number is ready (the DRDY bit should be set to ‘1’)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit will still be high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in *Section 33.3.7* shall be used.

Note: When data is not ready (DRDY=”0”) RNG_DR returns zero.

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 33.4: RNG low-power usage on page 1237*.

Software post-processing

No specific software post-processing/conditioning is required to meet AIS-31 approvals. If a NIST approved DRBG with 128 bits of security strength is required an approved random generator software must be built around the RNG true random number generator.

Built-in health check functions are described in *Section 33.3.3: Random number generation*.

33.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock is used for noise source sampling. Recommended clock configurations are detailed in *Section 33.7: Entropy source validation*.

Caution: When the CED bit in the RNG_CR register is set to “0”, the RNG clock frequency **must be higher** than AHB clock frequency divided by 32, otherwise the clock checker will flag a clock error (CECS or CEIS in the RNG_SR register) and the RNG will stop producing random numbers.

See *Section 33.3.1: RNG block diagram* for details (AHB and RNG clock domains).

33.3.7 Error management

In parallel to random number generation an health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG stops generating random numbers and sets to “1” both the **CEIS** and **CECS** bits to indicate that a clock error occurred. In this case, the application should check that the RNG clock is configured correctly (see *Section 33.3.6: RNG clocking*) and then it must clear

the CEIS bit interrupt flag. As soon as the RNG clock operates correctly, the CECS bit will be automatically cleared.

The RNG operates only when the CECS flag is set to “0”. However note that the clock error has no impact on the previously generated random numbers, and the RNG_DR register contents can still be used.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to “1” both **SEIS** and **SECS** bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy. If the error was detected during the initialization phase the whole initialization sequence will be automatically restarted by the RNG.

The following sequence shall be used to fully recover from a seed error after the RNG initialization:

1. Clear the SEIS bit by writing it to “0”.
2. Read out 12 words from the RNG_DR register, and discard each of them in order to clean the pipeline.
3. Confirm that SEIS is still cleared. Random number generation is back to normal.

33.4 RNG low-power usage

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to “1” by setting the RNGEN bit to “0” in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN=’0’ following features are available to software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case the RNG must be re-enabled by the application until at least 32 new bits have been collected from the noise source and a complete conditioning round has been done. It corresponds to 16 RNG clock cycles to sample new bits, and 216 AHB clock cycles to run a conditioning round.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (i.e. well before the DRDY bit rises for the first time), the initialization sequence will resume from where it was stopped when RNGEN bit is set to “1”.

33.5 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 33.3.7: Error management*
- Clock error, see *Section 33.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 270*

Table 270. RNG interrupt requests

Interrupt event	Event flag	Enable control bit
Data ready flag	DRDY	IE
Seed error flag	SEIS	IE
Clock error flag	CEIS	IE

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

33.6 RNG processing time

The conditioning stage can produce four 32-bit random numbers every 16x $\frac{f_{AHB}}{f_{RNG}}$ clock cycles, if the value is higher than 213 cycles (213 cycles otherwise).
More time is needed for the first set of random numbers after the device exits reset (see *Section 33.3.4: RNG initialization*). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if $f_{AHB} < f_{threshold}$
- 192 RNG clock cycles + 213 AHB cycles, if $f_{AHB} \geq f_{threshold}$

With $f_{threshold} = (213 \times f_{RNG})/ 64$

33.7 Entropy source validation

33.7.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral against AIS-31 PTG.2 set of tests. The results can be provided on demand or the customer can reproduce the measurements using the AIS reference software. The customer could also test the RNG against an older NIST SP800-22 set of tests.

33.7.2 Validation conditions

STMicroelectronics has validated the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz (CED bit = ’0’ in RNG_CR register) and rng_clk = 400 kHz (CED bit = ‘1’ in RNG_CR register).
- AHB clock rng_hclk= 216 MHz

33.7.3 Data collection

In order to run statistical tests it is required to collect samples from the entropy source at raw data level as well as at the output of the entropy source.

Please contact STMicroelectronics if above samples need to be retrieved for your product.

33.8 RNG registers

The RNG is associated with a control register, a data register and a status register.

33.8.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
										rw		rw	rw		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enable
1: Clock error detection is disable

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, i.e. to enable or disable CED the RNG must be disabled.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt Enable

0: RNG Interrupt is disabled
1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY=’1’, SEIS=’1’ or CEIS=’1’ in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.
1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

33.8.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing it to ‘0’.

0: No faulty sequence detected

1: At least one faulty sequence has been detected. See **SECS** bit description for details.

An interrupt is pending if IE = ‘1’ in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing it to ‘0’.

0: The RNG clock is correct (fRNGCLK> fHCLK/32)

1: The RNG has been detected too slow (fRNGCLK< fHCLK/32)

An interrupt is pending if IE = ‘1’ in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

– One of the noise source has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bit patterns (“01” or “10”)

– Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bit patterns (“01” or “10”)

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct (fRNGCLK> fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow (fRNGCLK< fHCLK/32).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to “0”.

Bit 0 **DRDY:** Data Ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to ‘0’ until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN=’0’ in the RNG_CR register).

If IE=’1’ in the RNG_CR register, an interrupt is generated when DRDY=’1’.

33.8.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. After being read this register delivers a new random value after 216 periods of AHB clock if the output FIFO is empty.

The content of this register is valid when DRDY='1', even if RNGEN='0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data which are valid when DRDY='1'. When DRDY='0' RNDATA value is zero.

33.8.4 RNG register map

Table 271 gives the RNG register map and reset values.

Table 271. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value																											0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
	Reset value																										0	0			0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

34 Cryptographic processor (CRYP)

34.1 Introduction

The cryptographic processor (CRYP) can be used both to encrypt and decrypt data using the DES, Triple-DES or AES algorithms. It is a fully compliant implementation of the following standards:

- The data encryption standard (DES) and Triple-DES (TDES) as defined by Federal Information Processing Standards Publication (FIPS PUB 46-3, Oct 1999), and the American National Standards Institute (ANSI X9.52)
- The advanced encryption standard (AES) as defined by Federal Information Processing Standards Publication (FIPS PUB 197, Nov 2001)

Multiple key sizes and chaining modes are supported:

- DES/TDES chaining modes ECB and CBC, supporting standard 56-bit keys with 8-bit parity per key
- AES chaining modes ECB, CBC, CTR, GCM, GMAC, CCM for key sizes of 128, 192 or 256 bits

The CRYP is a 32-bit AHB peripheral. It supports DMA transfers for incoming and outgoing data (two DMA channels are required). The peripheral also includes input and output FIFOs (each 8 words deep) for better performance.

The CRYP peripheral provides hardware acceleration to AES and DES cryptographic algorithms packaged in STM32 cryptographic library.

34.2 CRYP main features

- Compliant implementation of the following standards:
  - NIST *FIPS publication 46-3, Data Encryption Standard (DES)*
  - ANSI *X9.52, Triple Data Encryption Algorithm Modes of Operation*
  - NIST *FIPS publication 197, Advanced Encryption Standard (AES)*
- AES symmetric block cipher implementation
  - 128-bit data block processing
  - Support for 128-, 192- and 256-bit cipher key lengths
  - Encryption and decryption with multiple chaining modes: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Counter mode (CTR), Galois Counter Mode (GCM), Galois Message Authentication Code mode (GMAC) and Counter with CBC-MAC (CCM).
  - 14 (respectively 18) clock cycles for processing one 128-bit block of data with a 128-bit (respectively 256-bit) key in AES-ECB mode
  - Integrated key scheduler with its key derivation stage (ECB or CBC decryption only)
- DES/TDES encryption/decryption implementation
  - 64-bit data block processing
  - Support for 64-, 128- and 192-bit cipher key lengths (including parity)
  - Encryption and decryption with support of ECB and CBC chaining modes

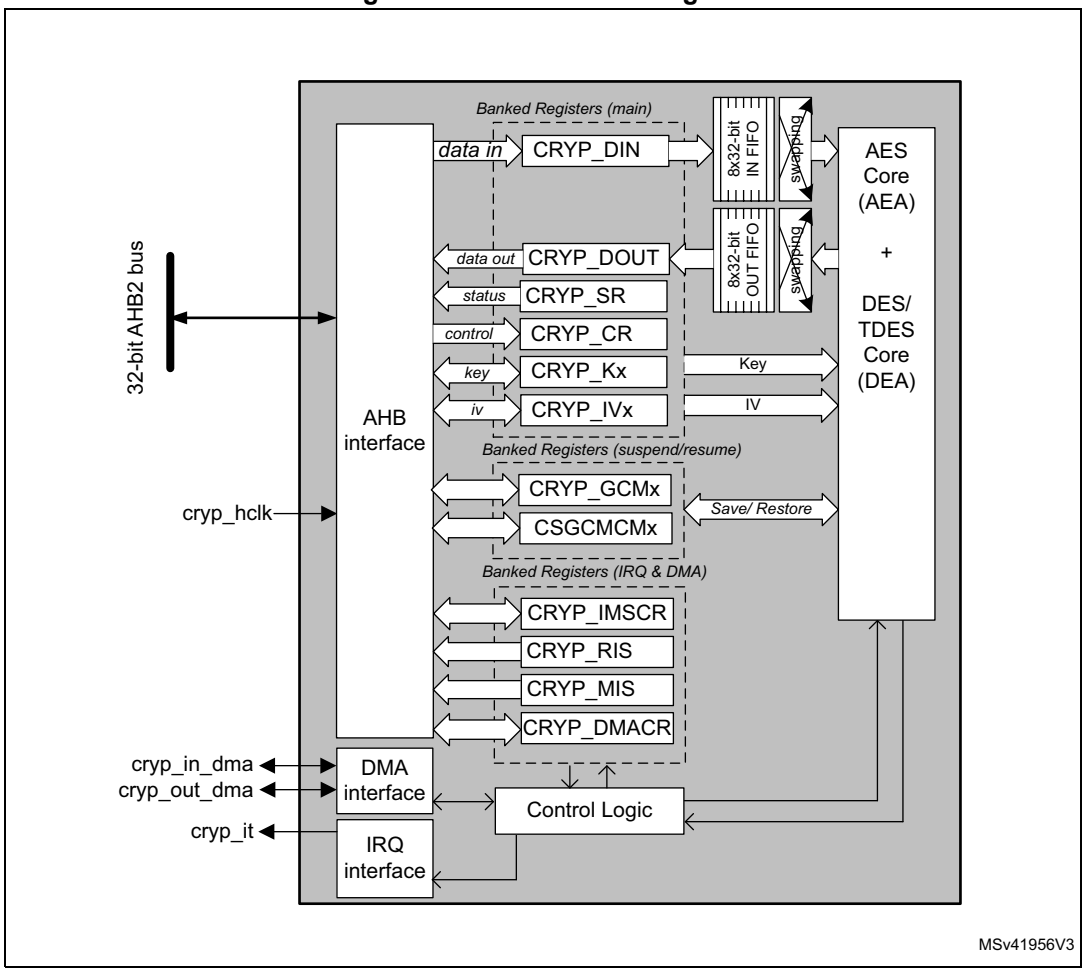
  - Direct implementation of simple DES algorithms (a single key K1 is used)
  - 16 (respectively 64) clock cycles for processing one 64-bit block of data in DES (respectively TDES) ECB mode
  - Software implementation of ciphertext stealing
- Features common to DES/TDES and AES
  - AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
  - 256-bit register for storing the cryptographic key (8x 32-bit registers)
  - 128-bit registers for storing initialization vectors (4× 32-bit)
  - 1x32-bit INPUT buffer associated with an internal IN FIFO of eight 32- bit words, corresponding to four incoming DES blocks or two AES blocks
  - 1x32-bit OUTPUT buffer associated with an internal OUT FIFO of eight 32-bit words, corresponding to four processed DES blocks or two AES blocks
  - Automatic data flow control supporting direct memory access (DMA) using two channels (one for incoming data, one for processed data). Single and burst transfers are supported.
  - Data swapping logic to support 1-, 8-, 16- or 32-bit data
  - Possibility for software to suspend a message if the cryptographic processor needs to process another message with higher priority (suspend/resume operation)

34.3 CRYP functional description

34.3.1 CRYP block diagram

Figure 249 shows the block diagram of the cryptographic processor.

Figure 249. CRYP block diagram



34.3.2 CRYP internal signals

Table 272 provides a list of useful-to-know internal signals available at cryptographic processor level and not at STM32 product level (on pads).

Table 272. CRYP internal input/output signals

Signal name	Signal type	Description
cryp_hclk	digital input	AHB bus clock
cryp_it	digital output	Cryptographic processor global interrupt request
cryp_in_dma	digital input/output	IN FIFO DMA burst request/ acknowledge
cryp_out_dma	digital input/output	OUT FIFO DMA burst request/ acknowledge (with single request for DES)

34.3.3 CRYP DES/TDES cryptographic core

Overview

The DES/Triple-DES cryptographic core consists of three components:

- The DES Algorithm (DEA core)
- Multiple keys (one for the DES algorithm, one to three for the TDES algorithm)
- The initialization vector, which is used only in CBC mode

The DES/Triple-DES cryptographic core provides two operating modes:

- **ALGODIR=0**: Plaintext encryption using the key stored in the CRYP_Kx registers.
- **ALGODIR=1**: Ciphertext decryption using the key stored in the CRYP_Kx registers.

The operating mode is selected by programming the ALGODIR bit in the CRYP_CR register.

Typical data processing

Typical usage of the cryptographic processor in DES modes can be found in*Section 34.3.10: CRYP DES/TDES basic chaining modes (ECB, CBC)*.

Note: The outputs of the intermediate DEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IV registers in CBC mode.

DES keying and chaining modes

The TDES allows three different keying options:

- *Three independent keys*

  The first option specifies that all the keys are independent, that is, K1, K2 and K3 are independent. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this option as the Keying Option 1 and, to the TDES as 3-key TDES.

- *Two independent keys*

  The second option specifies that K1 and K2 are independent and K3 is equal to K1, that is, K1 and K2 are independent, K3 = K1. FIPS PUB 46-3 – 1999 (and ANSI X9.52

– 1998) refers to this second option as the Keying Option 2 and, to the TDES as 2-key TDES.

- *Three equal keys*

  The third option specifies that K1, K2 and K3 are equal, that is:

  K1 = K2 = K3

  FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to the third option as the Keying Option 3. This “1-key” TDES is equivalent to single DES.

The following chaining algorithms are supported by the DES hardware and can be selected through the ALGOMODE bits in the CRYP_CR register:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)

These modes are described in details in *Section 34.3.10: CRYP DES/TDES basic chaining modes (ECB, CBC)*.

34.3.4 CRYP AES cryptographic core

Overview

The AES cryptographic core consists of the following components:

- The AES Algorithm (AEA core)
- The Multiplier over a binary Galois field (GF2mul)
- The key information
- The initialization vector (IV) or Nonce information
- Chaining algorithms logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks of (four words) with 128-, 192- or 256-bit key lengths. Depending on the chaining mode, the peripheral requires zero or one 128-bit initialization vector (IV).

The cryptographic peripheral features two operating modes:

- **ALGODIR=0**: Plaintext encryption using the key stored in the CRYP_Kx registers.
- **ALGODIR=1**: Ciphertext decryption using the key stored in the CRYP_Kx registers. When ECB and CBC chaining modes are selected, an initial key derivation process is automatically performed by the cryptographic peripheral.

The operating mode is selected by programming the ALGODIR bit in the CRYP_CR register.

Typical data processing

A description of cryptographic processor typical usage in AES mode can be found in *Section 34.3.11: CRYP AES basic chaining modes (ECB, CBC)*.

Note: The outputs of the intermediate AEA stages is never revealed outside the cryptographic boundary, with the exclusion of the IV registers.

AES chaining modes

The following chaining algorithms are supported by the cryptographic processor and can be selected through the ALGOMODE bits in the CRYP_CR register:

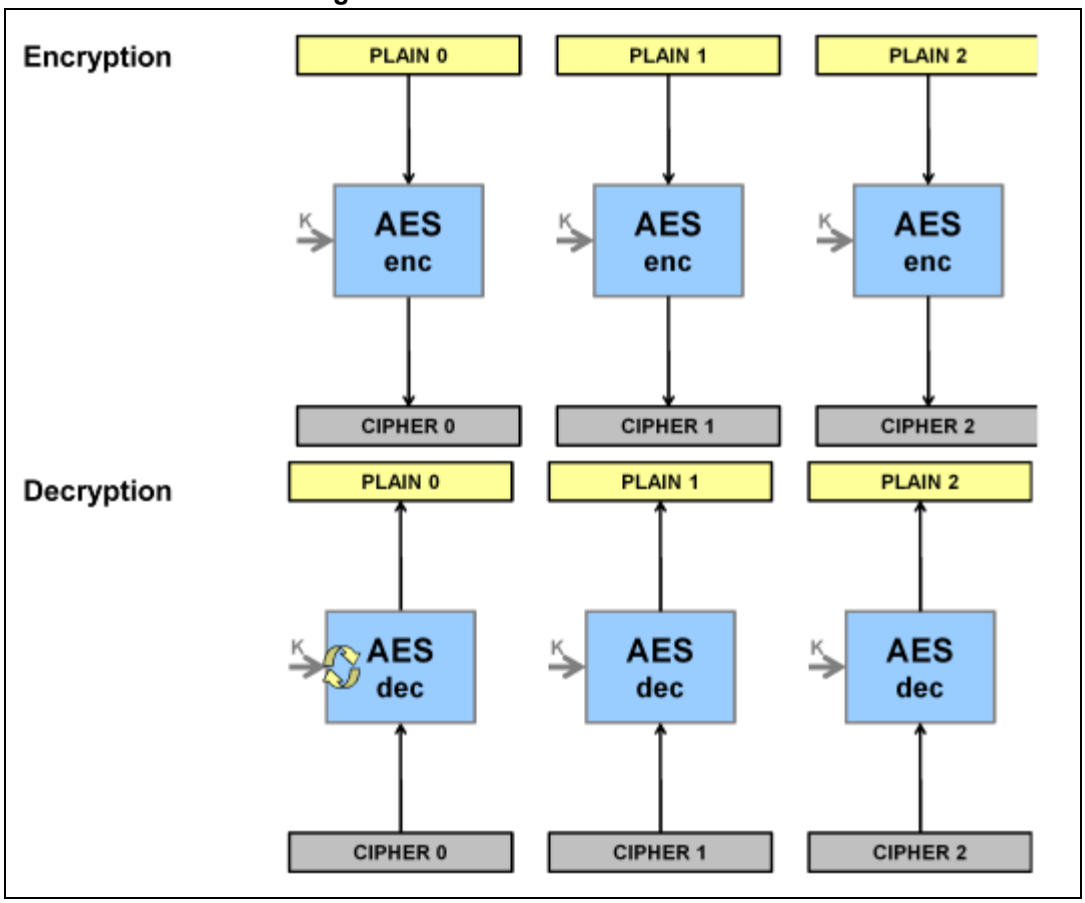
- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Counter Mode (CTR)
- Galois/Counter Mode (GCM)
- Galois Message Authentication Code mode (GMAC)
- Counter with CBC-MAC (CCM)

A quick introduction on these chaining modes can be found in the following subsections.

For detailed instructions, refer to *Section 34.3.11: CRYP AES basic chaining modes (ECB, CBC)* and onward.

AES Electronic CodeBook (ECB)

Figure 250. AES-ECB mode overview

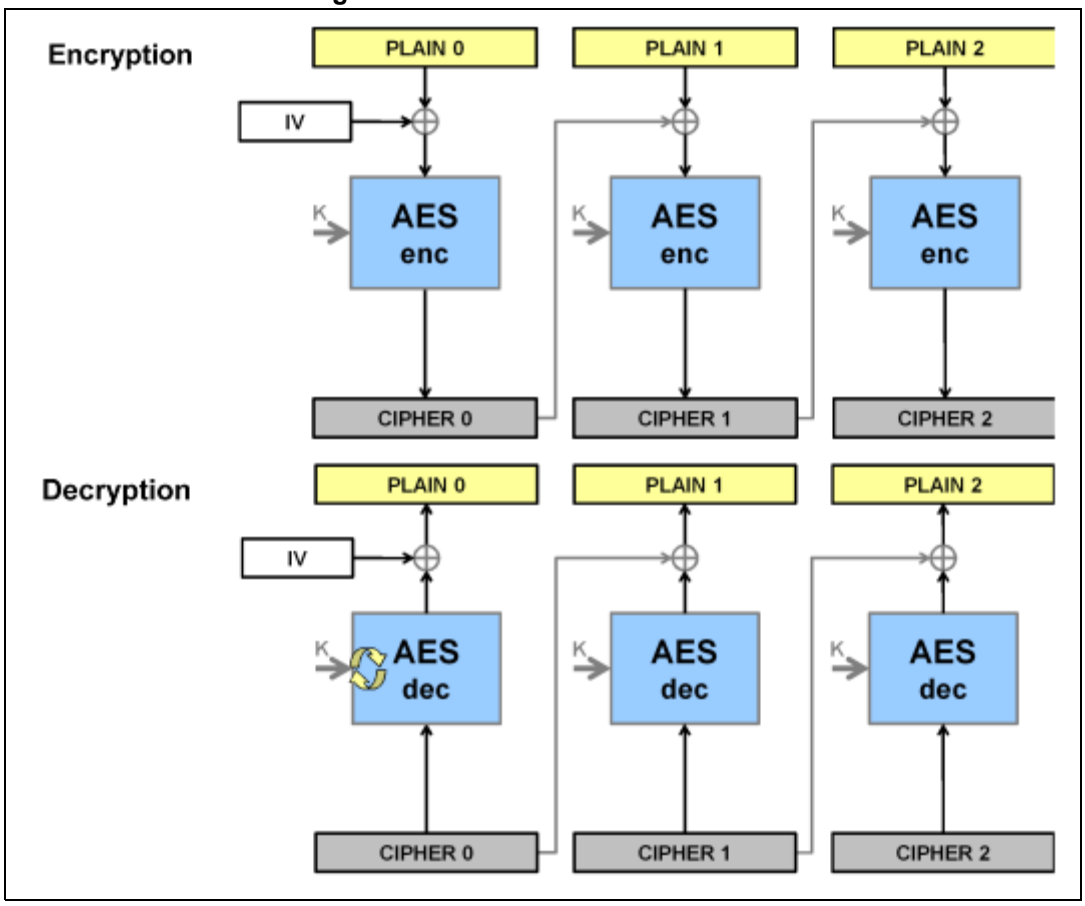


ECB is the simplest operating mode. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

AES Cipher block chaining (CBC)

Figure 251. AES-CBC mode overview

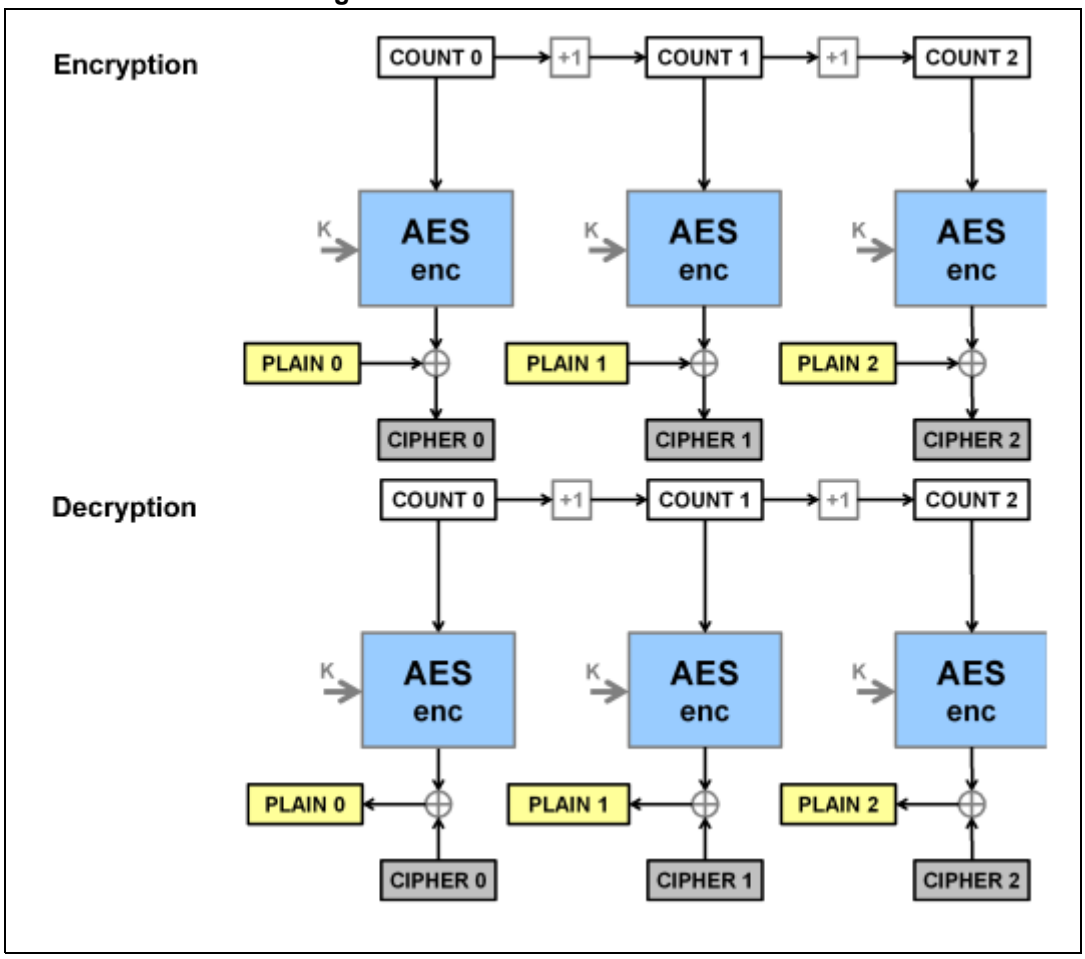


CBC operating mode chains the output of each block with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption,a special key scheduling is required before processing the first block.

AES Counter mode (CTR)

Figure 252. AES-CTR mode overview

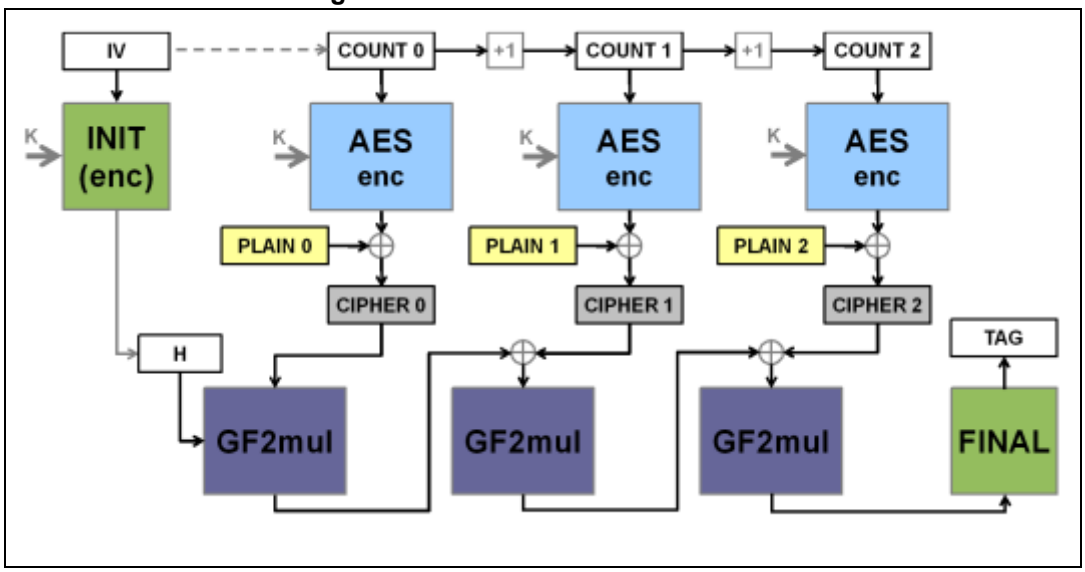


The CTR mode uses the AES core to generate a key stream; these keys are then XORed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Note: Unlike ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the counter blocks.

AES Galois/Counter mode (GCM)

Figure 253. AES-GCM mode overview

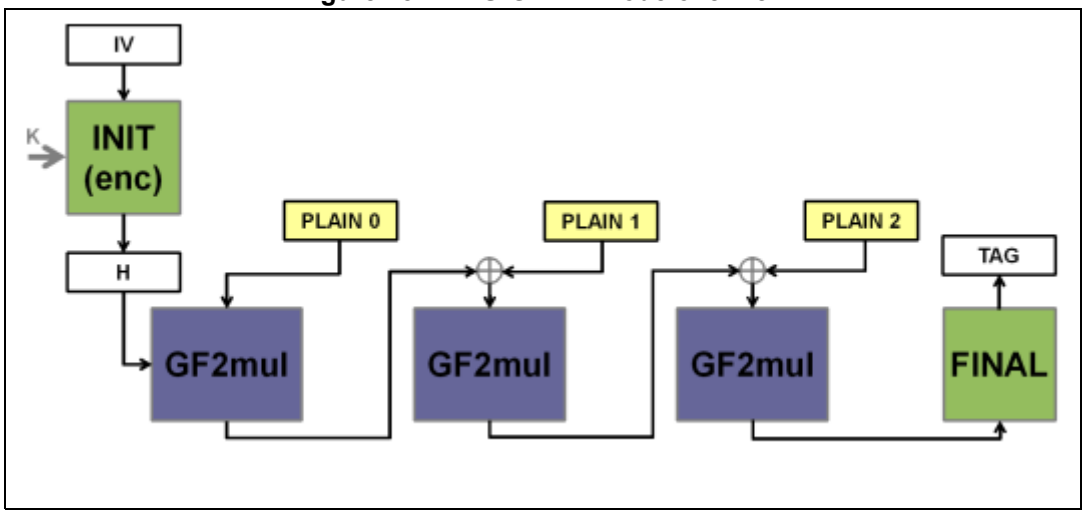


In Galois/Counter mode (GCM), the plaintext message is encrypted, while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

AES Galois Message Authentication Code (GMAC)

Figure 254. AES-GMAC mode overview

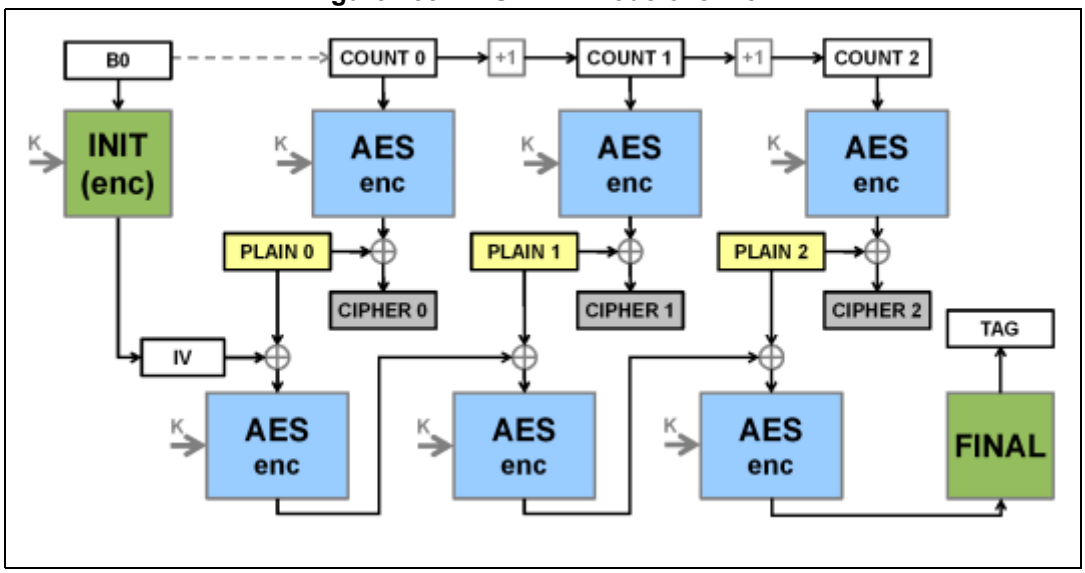


Galois Message Authentication Code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

GMAC is similar to Galois/Counter mode (GCM), except that it is applied on a message composed only by clear-text authenticated data (i.e. only header, no payload).

AES Counter with CBC-MAC (CCM)

Figure 255. AES-CCM mode overview



In Counter with Cipher Block Chaining-Message Authentication Code (CCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM CCM chaining mode, AES-CCM mode can be applied on a message composed only by cleartext authenticated data (i.e. only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its usage is not recommended by NIST.

34.3.5 CRYP procedure to perform a cipher operation

Introduction

To understand how the cryptographic peripheral operates, a typical cipher operation is described below. For the detailed peripheral usage according to the cipher mode, refer to the specific section, e.g. *Section 34.3.11: CRYP AES basic chaining modes (ECB, CBC)*.

The flowcharts shown in *Figure 256* and *Figure 257* describe the way STM32 cryptographic library implements DES (respectively AES) algorithm. The cryptographic processor accelerates the execution of the following cryptographic algorithms:

- AES-128, AES-192, AES-256 bit in the following modes: ECB, CBC, CTR, CCM, GCM
- DES, TripleDES in the following modes: ECB, CBC

Note: For more details on the cryptographic library, refer to use manual UM1924 "STM32 crypto library" available from www.st.com.

Figure 256. STM32 cryptolib DES/TDES flowcharts

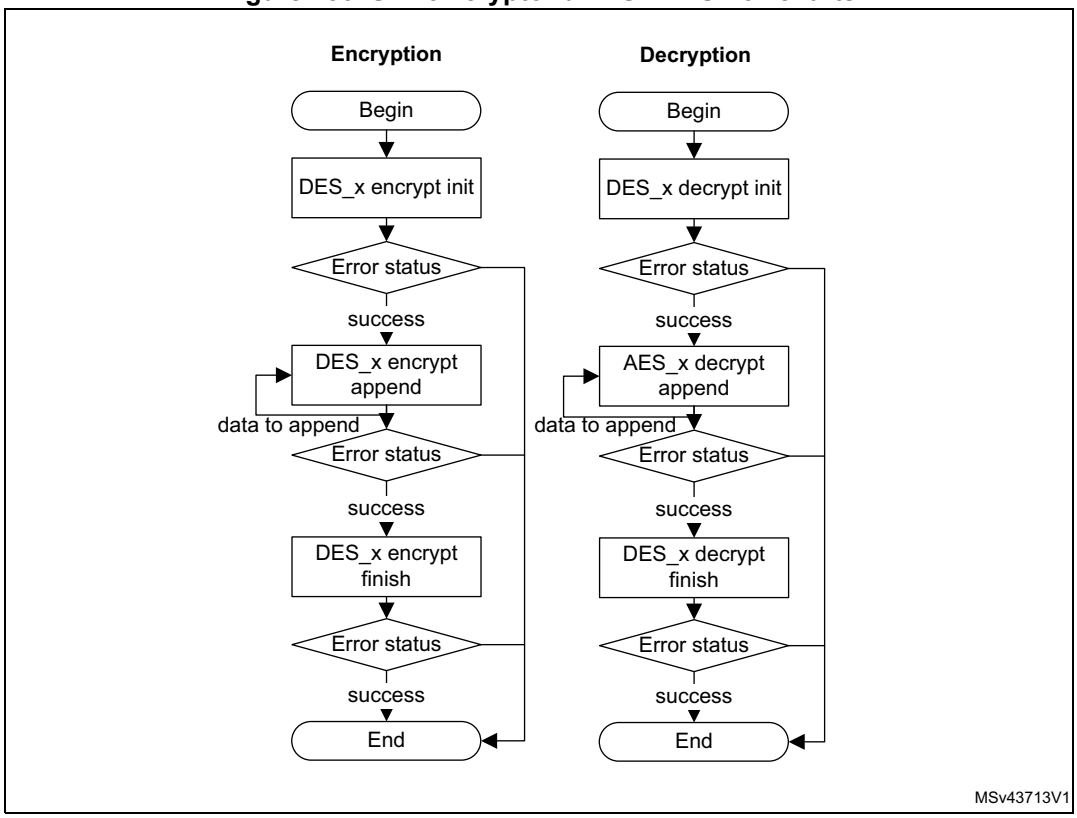
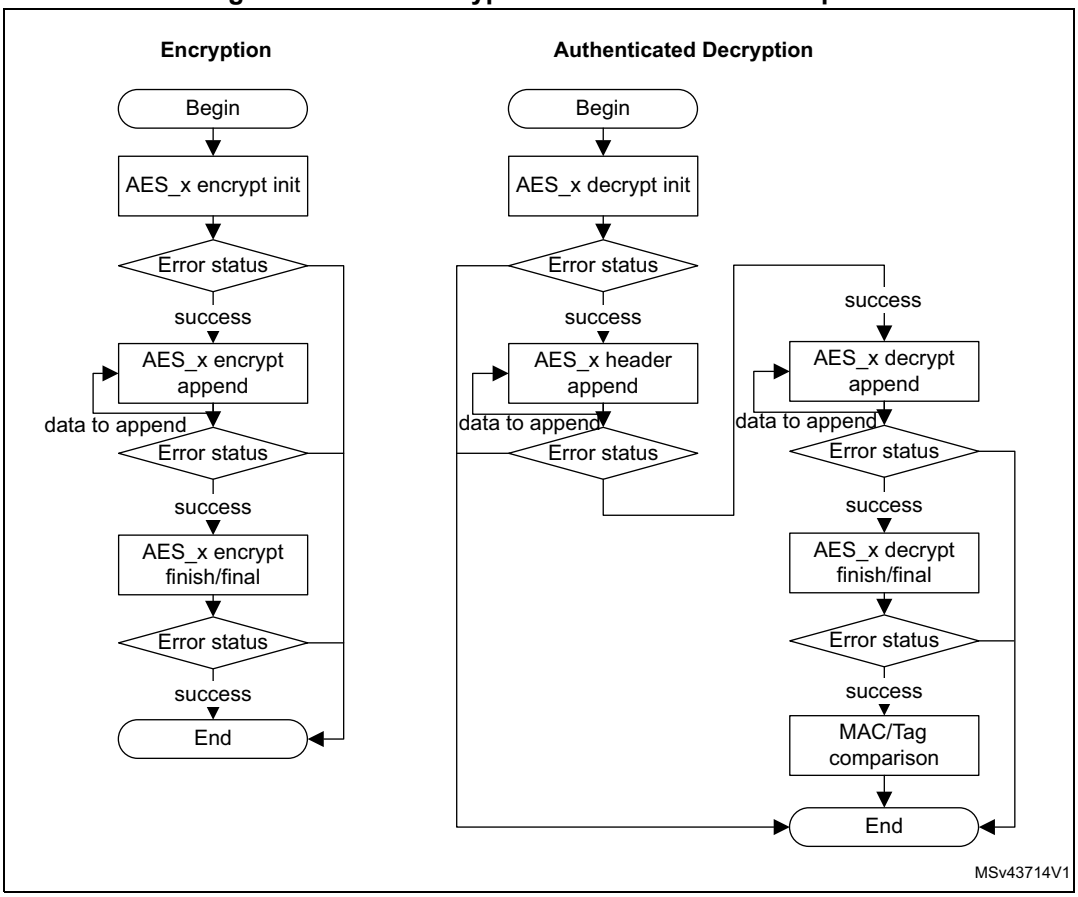


Figure 257. STM32 cryptolib AES flowchart examples



CRYP initialization

1. Initialize the cryptographic processor. The order of operations is not important except for AES-ECB and AES-CBC decryption, where the key preparation requires a specific sequence.
   a) Disable the cryptographic processor by setting to 0 the CRYPEN bit in the CRYP_CR register.
   b) Configure the key size (128-, 192- or 256-bit, in the AES only) with the KEYSIZE bits in the CRYP_CR register.
   c) Write the symmetric key into the CRYP_KxL/R registers (2 to 8 registers to be written depending on the algorithm).
   d) Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE bits in the CRYP_CR register.
   e) In case of decryption in AES-ECB or AES-CBC mode, prepare the key that has been written. First configure the key preparation mode by setting the ALGOMODE bits to 0b111 in the CRYP_CR register. Then write the CRYPEN bit to 1: the BUSY

bit is set automatically. Wait until BUSY returns to 0 (CRYPEN is automatically cleared as well): the key is prepared for decryption.

f) Configure the algorithm and chaining with the ALGOMODE bits in the CRYP_CR register.

g) Configure the direction (encryption/decryption) through the ALGODIR bit in the CRYP_CR register.

h) When it is required (e.g. CBC or CTR chaining modes), write the initialization vectors into the CRYP_IVxL/R register.

2. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_CR register.

Preliminary warning for all cases

If the ECB or CBC mode is selected and data are not a multiple of 64 bits (for DES) or 128 bits (for AES), the second and the last block management is more complex than the sequences below. Refer to *Section 34.3.8: CRYP stealing and data padding* for more details.

Appending data using the CPU in polling mode

1. Enable the cryptographic processor by setting to 1 the CRYPEN bit in the CRYP_CR register.
2. Write data in the IN FIFO (one block or until the FIFO is full).
3. Repeat the following sequence until the second last block of data has been processed:
   a) Wait until the not-empty-flag OFNE is set to 1, then read the OUT FIFO (one block or until the FIFO is empty).
   b) Wait until the not-full-flag IFNF is set to 1, then write the IN FIFO (one block or until the FIFO is full) except if it is the last block.
4. The BUSY bit is set automatically by the cryptographic processor. At the end of the processing, the BUSY bit returns to 0 and both FIFOs are empty (IN FIFO empty flag IFEM=1 and OUT FIFO not empty flag OFNE=0).
5. If the next processing block is the last block, the CPU must pad (when applicable) the data with zeroes to obtain a complete block
6. When the operation is complete, the cryptographic processor can be disabled by clearing the CRYPEN bit in CRYP_CR register.

Appending data using the CPU in interrupt mode

1. Enable the interrupts by setting the INIM and OUTIM bits in the CRYP_IMSCR register.
2. Enable the cryptographic processor by setting to 1 the CRYPEN bit in the CRYP_CR register.
3. In the interrupt service routine that manages the input data:
   a) If the last block is being loaded, the CPU must pad (when applicable) the data with zeroes to have a complete block. Then load the block into the IN FIFO.
   b) If it is not the last block, load the data into the IN FIFO. You can load only one block (2 words for DES, 4 words for AES), or load data until the FIFO is full.
   c) In all cases, after the last word of data has been written, disable the interrupt by clearing the INIM interrupt mask.
4. In the interrupt service routine that manages the input data:

a) Read the output data from the OUT FIFO. You can read only one block (2 words for DES, 4 words for AES), or read data until the FIFO is empty.

b) When the last word has been read, INIM and BUSY bits are set to 0 and both FIFOs are empty (IFEM=1 and OFNE=0). You can disable the interrupt by clearing the OUTIM bit, and disable the peripheral by clearing the CRYPEN bit.

c) If you read the last block of cleartext data (i.e. decryption), optionally discard the data that is not part of message/payload.

Appending data using the DMA

1. Prepare the last block of data by optionally padding it with zeroes to have a complete block.
2. Configure the DMA controller to transfer the input data from the memory and transfer the output data from the peripheral to the memory, as described in *Section 34.3.19: CRYP DMA interface*. The DMA should be configured to set an interrupt on transfer completion to indicate that the processing is complete.
3. Enable the cryptographic processor by setting to 1 the CRYPEN bit in CRYP_CR register, then enable the DMA IN and OUT requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.
4. All the transfers and processing are managed by the DMA and the cryptographic processor. The DMA interrupt indicates that the processing is complete. Both FIFOs are normally empty and BUSY flag is set 0.

Caution: It is important that DMA controller empties the cryptographic processor output FIFO before filling up the cryptographic processor input FIFO. To achieve this, the DMA controller should be configured so that the transfer from the cryptographic peripheral to the memory has a higher priority than the transfer from the memory to the cryptographic peripheral.

34.3.6 CRYP busy state

The cryptographic processor is busy and processing data (BUSY set to 1 in CRYP_SR register) when all the conditions below are met:

- CRYPEN = 1 in CRYP_CR register.
- There are enough data in the input FIFO (at least two words for the DES or TDES algorithm mode, four words for the AES algorithm mode).
- There is enough free-space in the output FIFO (at least two word locations for DES, four for AES).

Write operations to the CRYP_Kx(L/R)R key registers, to the CRYP_IVx(L/R)R initialization registers, or to bits [9:2] of the CRYP_CR register, are ignored when cryptographic processor is busy (i.e. the registers are not modified). It is thus not possible to modify the configuration of the cryptographic processor while it is processing a data block.

It is possible to clear the CRYPEN bit while BUSY bit is set to 1. In this case the ongoing DES/TDES or AES processing first completes (i.e. the word results are written to the output FIFO) before the BUSY bit is cleared by hardware.

Note: If the application needs to suspend a message to process another one with a higher priority, refer to Section 34.3.9: CRYP suspend/resume operations

When a block is being processed in DES or TDES mode, if the output FIFO becomes full and the input FIFO contains at least one new block, then the new block is popped off the

input FIFO and the BUSY bit remains high until there is enough space to store this new block into the output FIFO.

34.3.7 Preparing the CRYP AES key for decryption

When performing an AES **ECB or CBC decryption**, the AES key has to be prepared, i.e. a complete key schedule of encryption is required before performing the decryption. In other words, the key in the last round of encryption must be used as the first round key for decryption.

This preparation is not required in any other AES modes than AES ECB or CBC decryption.

If the application software stores somehow the initial key prepared for decryption, the key scheduling operation can be performed only once for all the data to be decrypted with a given cipher key.

Note: The latency of the key preparation operation is 14, 16 or 18 clock cycles depending on the key size (128-, 192- or 256-bit).

The CRYP key preparation process is performed as follow:

1. Write the encryption key to K0...K3 key registers.
2. Program ALGOMODE bits to 0b111 in CRYP_CR. Writing this value when CRYPEN is et to 1 immediately starts an AES round for key preparation. The BUSY bit in the CRYP_SR register is set to 1.
3. When the key processing is complete, the resulting key is copied back into the K0...K3 key registers, and the BUSY bit is cleared.

Note: As the CRYPEN bitfield is reset by hardware at the end of the key preparation, the application software must set it again for the next operation.

34.3.8 CRYP stealing and data padding

When using DES or AES algorithm in **ECB** or **CBC** modes to manage messages that are not multiple of the block size (64 bits for DES, 128 bits for AES), use ciphertext stealing techniques such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since the cryptographic processor does not implement such techniques, **the last two blocks** must be handled in a special way by the application.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when the AES algorithm is used in other modes than ECB or CBC, incomplete input data blocks (i.e. block shorter than 128 bits) have to be padded with zeroes by the application prior to encryption (i.e. extra bits should be appended to the trailing end of the data string). After decryption, the extra bits have to be discarded. The cryptographic processor does not implement automatic data padding operation to **the last block**, so the application should follow the recommendation given in *Section 34.3.5: CRYP procedure to perform a cipher operation* to manage messages that are not multiple of 128 bits.

Note: Padding data are swapped in a similar way as normal data, according to the DATATYPE field in CRYP_CR register (see Section 34.3.16: CRYP data registers and data swapping for details).

With this version of cryptographic processor, a special workaround is required in order to properly compute authentication tags while doing a **GCM encryption** or a **CCM decryption** with the last block of payload size **inferior to 128 bits**. This workaround is described below:

- During GCM encryption payload phase and before inserting a last plaintext block smaller than 128 bits, the application has to follow the below sequence:
  a) Disable the peripheral by setting the CRYPEN bit to 0 in CRYP_CR.
  b) Load CRYP_IV1R register content in a temporary variable. Decrement the value by 1 and reinsert the result in CRYP_IV1R register.
  c) Change the AES mode to CTR mode by writing the ALGOMODE bitfield to 0b0110 in the CRYP_CR register.
  a) Set the CRYPEN bit to 1 in CRYP_CR to enable again the peripheral.
  b) Pad the last block (smaller than 128 bits) with zeros to have a complete block of 128 bits, then write it into CRYP_DIN register.
  c) Upon encryption completion, read the 128-bit generated ciphertext from the CRYP_DOUT register and store it as intermediate data.
  d) Change again the AES mode to GCM mode by writing the ALGOMODE bitfield to 0b1000 in the CRYP_CR register.
  e) Select Final phase by writing the GCM_CCMPH bitfield to 0b11 in the CRYP_CR register.
  f) In the intermediate data, set to 0 the bits corresponding to the padded bits of the last payload block then insert the resulting data to CRYP_DIN register.
  g) When the operation is complete, read data from CRYP_DOUT. These data have to be discarded.
  h) Apply the normal Final phase as described in *Section 34.3.13: CRYP AES Galois/counter mode (GCM)*.

- During CCM decryption payload phase and before inserting a last ciphertext block smaller than 128 bits, the application has to follow the below sequence:
  a) To disable the peripheral, set the CRYPEN bit to 0 in CRYP_CR.
  b) Load CRYP_IV1R in a temporary variable (named here `IV1temp`).
  c) Load CRYP_CSGCMCCM0R, CRYP_CSGCMCCM1R, CRYP_CSGCMCCM2R, and CRYP_CSGCMCCM3R registers content from LSB to MSB in 128-bit temporary variable (named here `temp1`).
  d) Load in CRYP_IV1R the content previously stored in `IV1temp`.
  e) Change the AES mode to CTR mode by writing the ALGOMODE bitfield to 0b0110 in the CRYP_CR register.
  a) Set the CRYPEN bit to 1 in CRYP_CR to enable again the peripheral.
  b) Pad the last block (smaller than 128 bits) with zeros to have a complete block of 128 bits, then write it to CRYP_DIN register.
  c) Upon decryption completion, read the 128-bit generated data from DOUT register, and store them as intermediate data (here named `intdata_o`).
  d) Save again CRYP_CSGCMCCM0R, CRYP_CSGCMCCM1R, CRYP_CSGCMCCM2R, and CRYP_CSGCMCCM3R registers content, from LSB to MSB, in a new 128-bit temporary variable (named here `temp2`).
  e) Change again the AES mode to CCM mode by writing the ALGOMODE bitfield to 0b1001 in the CRYP_CR register.

f) Select the header phase by writing the GCM_CCMPH bitfield to 0b01 in the CRYP_CR register.

g) In the intermediate data (`intdata_o` which was generated with CTR), set to 0 the bits corresponding to the padded bits of the last payload block, XOR with `temp1`, XOR with `temp2`, and insert the resulting data into CRYP_DIN register. In other words:

```
CRYP_DIN=(intdata_o AND mask) XOR temp1 XOR temp2.
```

h) Wait for operation completion.

i) Apply the normal Final phase as described in *Section 34.3.15: CRYP AES Counter with CBC-MAC (CCM)*.

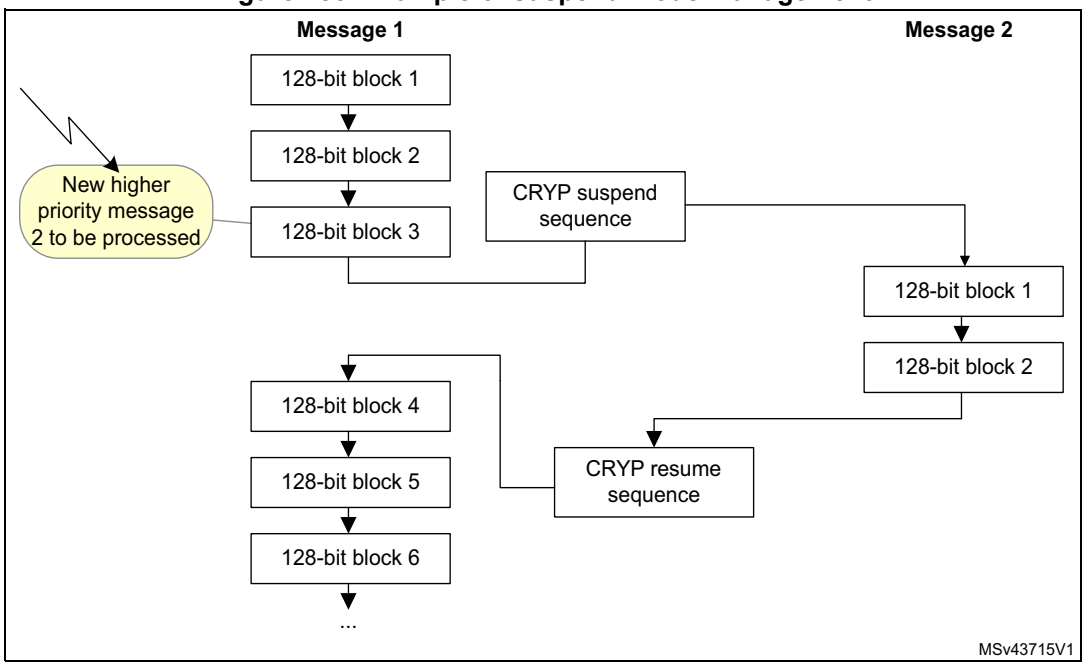
34.3.9 CRYP suspend/resume operations

A message can be suspended if another message with a higher priority has to be processed. When this highest priority message has been sent, the suspended message can be resumed in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can be resumed as soon as cryptographic processor is enabled again to receive the next data block.

Figure 258 gives an example of suspend.resume operation: message 1 is suspended in order to send a higher priority message (message 2), which is shorter than message 1 (AES algorithm).

Figure 258. Example of suspend mode management



A detailed description of suspend/resume operations can be found in each AES mode section.

34.3.10 CRYP DES/TDES basic chaining modes (ECB, CBC)

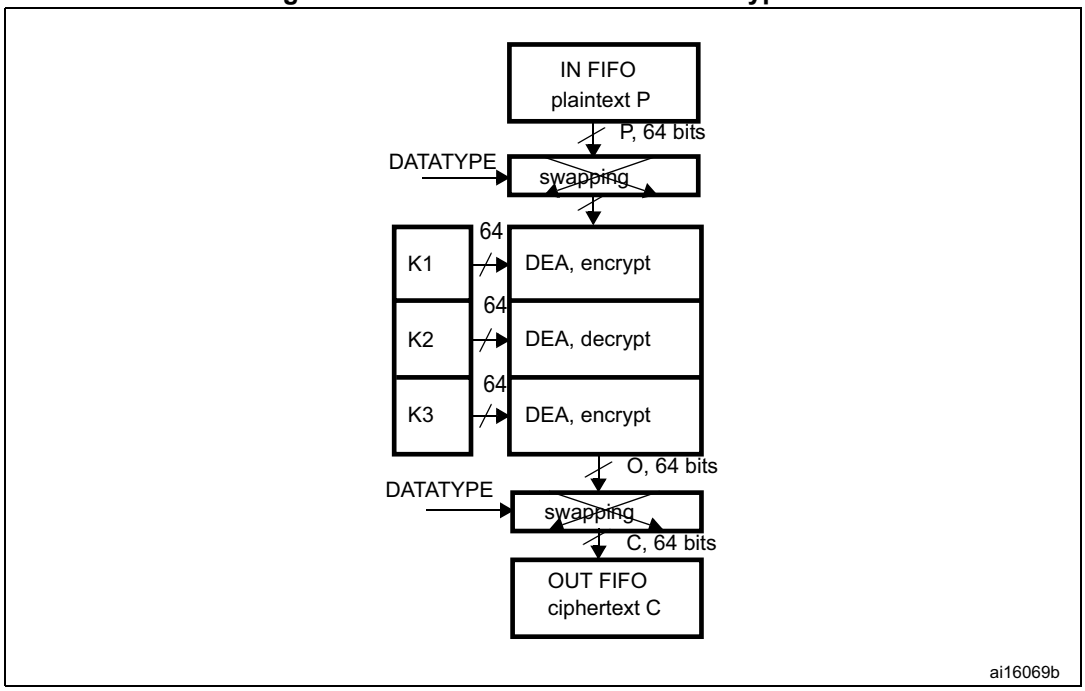
Overview

FIPS PUB 46-3 – 1999 (and ANSI X9.52-1998) provides a thorough explanation of the processing involved in the four operation modes supplied by the DES computing core: TDES-ECB encryption, TDES-ECB decryption, TDES-CBC encryption and TDES-CBC decryption. This section only gives a brief explanation of each mode.

DES/TDES-ECB encryption

Figure 259 illustrates the encryption in DES and TDES Electronic CodeBook (DES/TDES-ECB) mode. This mode is selected by writing in ALGOMODE to 0b000 and ALGODIR to 0 in CRYP_CR.

Figure 259. DES/TDES-ECB mode encryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

A 64-bit plaintext data block (P) is used after bit/byte/half-word as the input block (I). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA where the DES is performed in the decrypt state using K2. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K3. The resultant 64-bit output block (O) is used, after bit/byte/half-word swapping, as ciphertext (C) and it is pushed into the OUT FIFO.

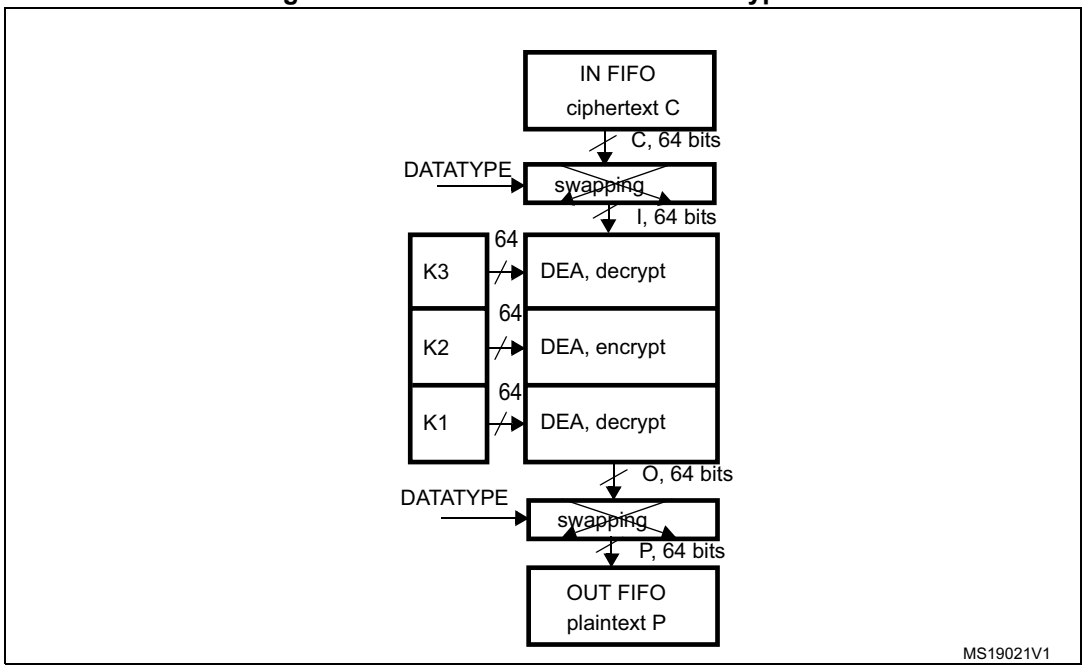
Note: For more information on data swapping, refer to Section 34.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 34.3.5: CRYP procedure to perform a cipher operation.

DES/TDES-ECB mode decryption

Figure 260 illustrates the decryption in DES and TDES Electronic CodeBook (DES/TDES-ECB) mode. This mode is selected by writing ALGOMODE to 0b000 and ALGODIR to 1 in CRYP_CR.

Figure 260. DES/TDES-ECB mode decryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

A 64-bit ciphertext block (C) is used, after bit/byte/half-word swapping, as the input block (I). The keying sequence is reversed compared to that used in the encryption process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K2. The new result is directly fed to the input of the DEA where the DES is performed in the decrypt state using K1. The resultant 64-bit output block (O), after bit/byte/half-word swapping, produces the plaintext (P).

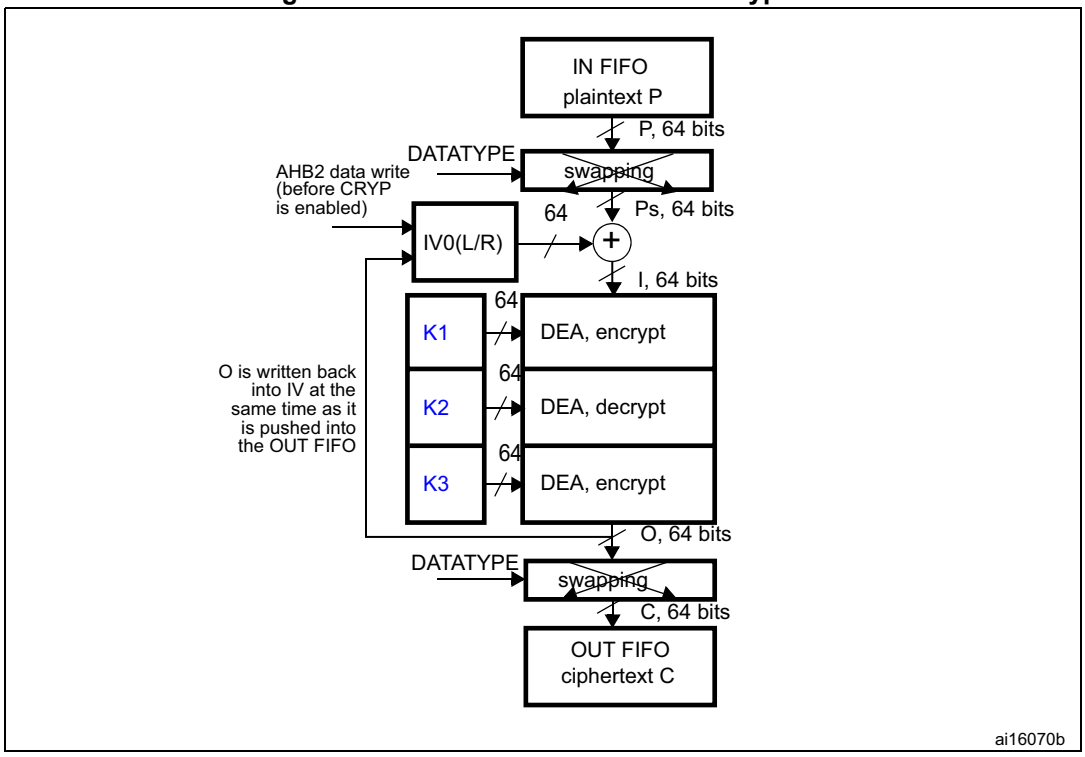
Note: For more information on data swapping refer to Section 34.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 34.3.5: CRYP procedure to perform a cipher operation.

DES/TDES-CBC encryption

Figure 261 illustrates the encryption in DES and TDES Cipher Block Chaining (DES/TDES-ECB) mode. This mode is selected by writing in ALGOMODE to 0b001 and ALGODIR to 0 in CRYP_CR.

Figure 261. DES/TDES-CBC mode encryption



K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

This mode begins by dividing a plaintext message into 64-bit data blocks. In TCBC encryption, the first input block (I_1), obtained after bit/byte/half-word swapping, is formed by exclusive-ORing the first plaintext data block (P_1) with a 64-bit initialization vector IV ($I_1 = IV \oplus P_1$). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA, which performs the DES in the decrypt state using K2. The output of this process is fed directly to the input of the DEA, which performs the DES in the encrypt state using K3. The resultant 64-bit output block (O_1) is used directly as the ciphertext (C_1), that is, $C_1 = O_1$.

This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I_2) = ($C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the TDEA to produce the second ciphertext block.

This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted.

If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.

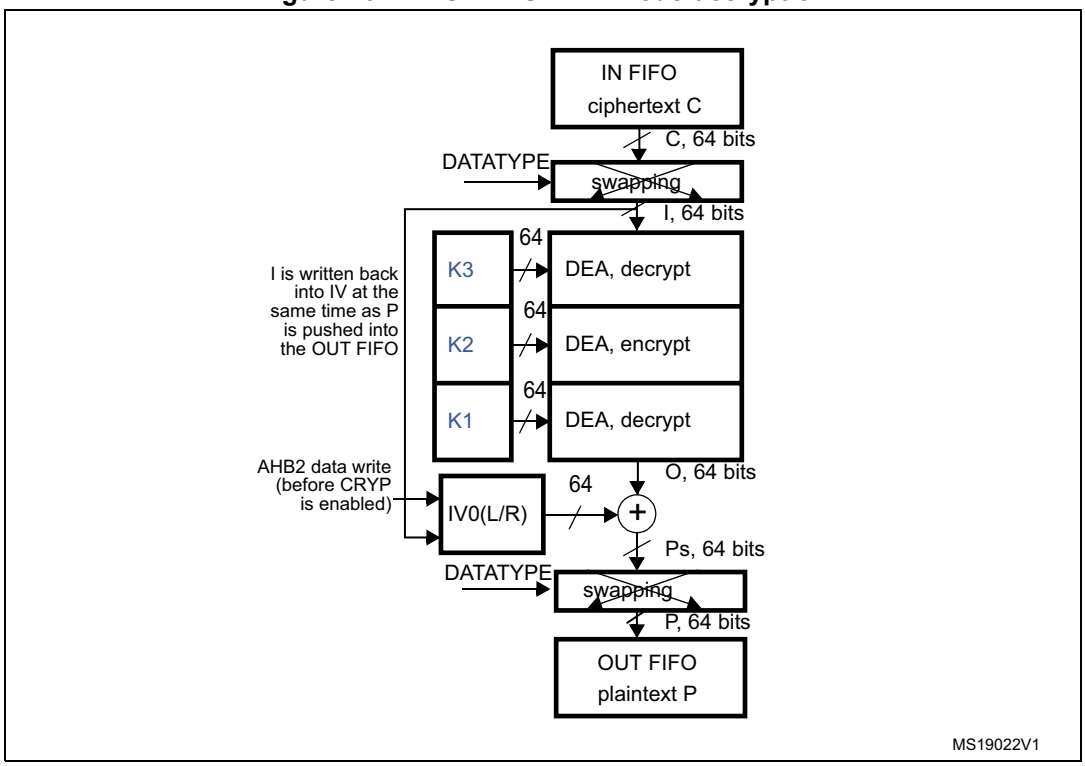
Note: For more information on data swapping refer to Section 34.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 34.3.5: CRYP procedure to perform a cipher operation.

DES/TDES-CBC decryption

Figure 261 illustrates the decryption in DES and TDES Cipher Block Chaining (DES/TDES-ECB) mode. This mode is selected by writing ALGOMODE to 0b001 and ALGODIR to 1 in CRYP_CR.

Figure 262. DES/TDES-CBC mode decryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

In this mode the first ciphertext block (C_1) is used directly as the input block (I_1). The keying sequence is reversed compared to that used for the encrypt process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed directly to the input of the DEA where the DES is processed in the encrypt state using K2. This resulting value is directly fed to the input of the DEA where the DES is processed in the decrypt state using K1. The resulting output block is exclusive-ORed with the IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus IV$).

The second ciphertext block is then used as the next input block and is processed through the TDEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). Note that P_2 and O_2 refer to the second block of data.

The DES/TDES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

Ciphertext representing a partial data block must be decrypted in a manner specified for the application.

Note: For more information on data swapping refer to Section 34.3.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 34.3.5: CRYP procedure to perform a cipher operation.

DES/TDES suspend/resume operations in ECB/CBC modes

Before interrupting the current message, the user application must respect the following steps:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR) and the BUSY bit is cleared. Alternatively, as the input FIFO can contain up to four unprocessed DES blocks, the application could decide for real-time reason to interrupt the cryptographic processing without waiting for the IN FIFO to be empty. In this case, the alternative is:
   a) Wait until OUT FIFO is empty (OFNE=0).
   b) Read back the data loaded in the IN FIFO that have not been processed and save them in the memory until the IN FIFO is empty.
3. If DMA is used stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2] in the CRYP_CR register). If CBC mode is selected, save the initialization vector registers, since CRYP_IVx registers have changed from initial values during the data processing.

Note: Key registers do not need to be saved as the original key value is known by the application.

5. If DMA is used, save the DMA controller status (such as the pointers to IN and OUT data transfers, number of remaining bytes).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be 0).
3. Configure again the cryptographic processor with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. If the CBC mode is selected, restore CRYP_IVx registers using the saved configuration.
5. Optionally, write the data that were saved during context saving into the IN FIFO.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again DMA requests for the cryptographic processor, by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

34.3.11 CRYP AES basic chaining modes (ECB, CBC)

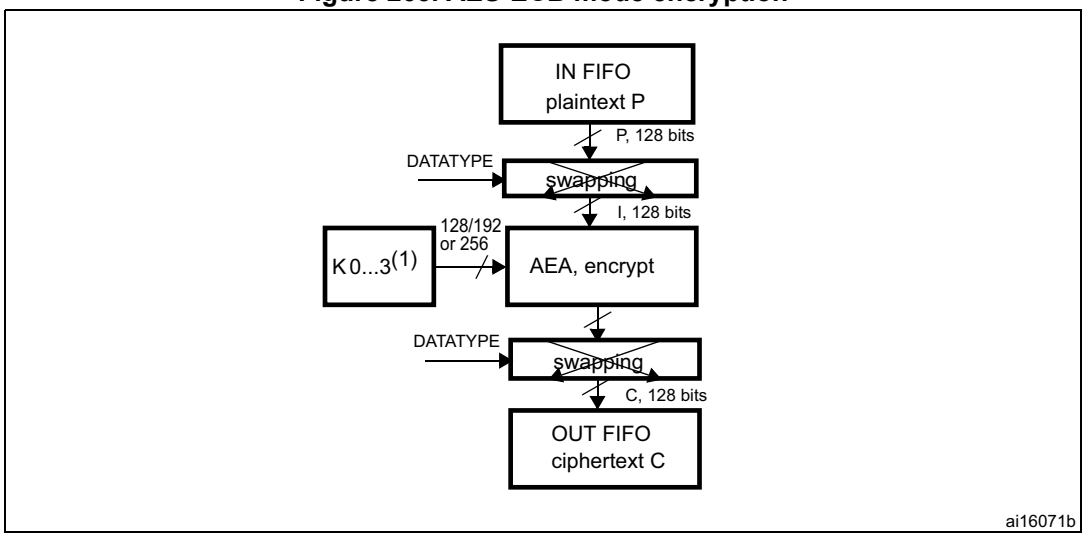
Overview

FIPS PUB 197 (November 26, 2001) provides a thorough explanation of the processing involved in the four basic operation modes supplied by the AES computing core: AES-ECB encryption, AES-ECB decryption, AES-CBC encryption and AES-CBC decryption. This section only gives a brief explanation of each mode.

AES ECB encryption

Figure 263 illustrates the AES Electronic codebook (AES-ECB) mode encryption. This mode is selected by writing ALGOMODE to 0b100 and ALGODIR to 0 in CRYP_CR.

Figure 263. AES-ECB mode encryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128: Key = [K3 K2].
   If Key size = 192: Key = [K3 K2 K1]
   If Key size = 256: Key = [K3 K2 K1 K0].

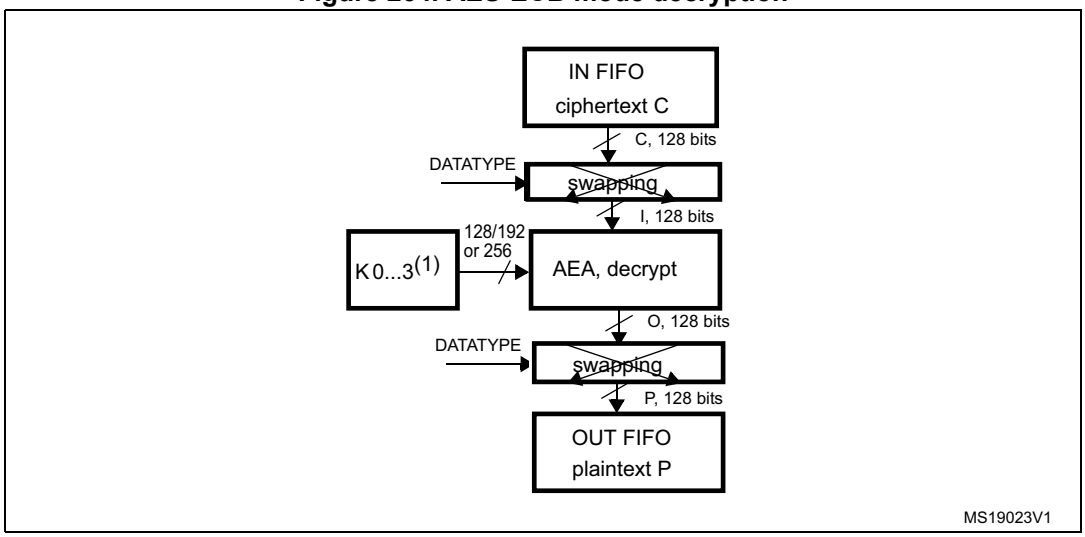
In this mode a 128- bit plaintext data block (P) is used after bit/byte/half-word swapping as the input block (I). The input block is processed through the AEA in the encrypt state using the 128, 192 or 256-bit key. The resultant 128-bit output block (O) is used after bit/byte/half-word swapping as ciphertext (C). It is then pushed into the OUT FIFO.

For more information on data swapping refer to *Section 34.3.16: CRYP data registers and data swapping*.

AES ECB decryption

Figure 264 illustrates the AES Electronic codebook (AES-ECB) mode decryption. This mode is selected by writing in ALGOMODE to 0b100 and ALGODIR to 1 in CRYP_CR.

Figure 264. AES-ECB mode decryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

To perform an AES decryption in ECB mode, the secret key has to be prepared (it is necessary to execute the complete key schedule for encryption) by collecting the last round key, and using it as the first round key for the decryption of the ciphertext. This preparation phase is computed by the AES core. Refer to *Section 34.3.7: Preparing the CRYP AES key for decryption* for more details on how to prepare the key.

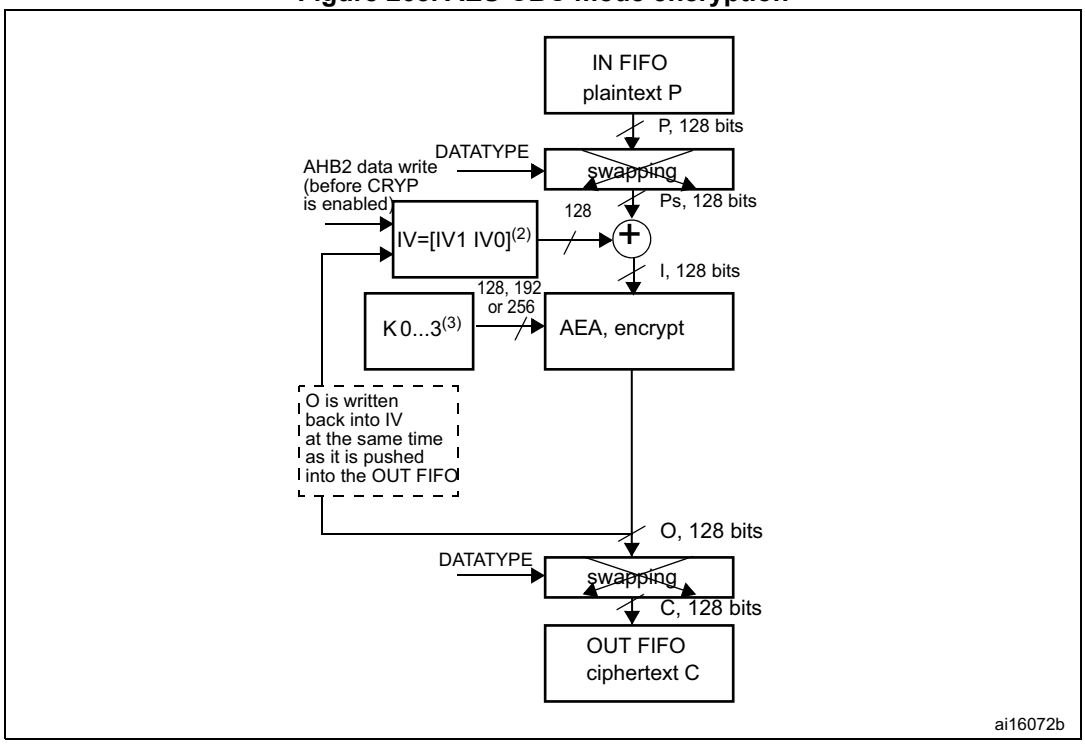
When the key preparation is complete, the decryption proceed as follow: a 128-bit ciphertext block (C) is used after bit/byte/half-word swapping as the input block (I). The keying sequence is reversed compared to that of the encryption process. The resultant 128-bit output block (O), after bit/byte or half-word swapping, produces the plaintext (P). The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

For more information on data swapping refer to *Section 34.3.16: CRYP data registers and data swapping*.

AES CBC encryption

Figure 265 illustrates the AES Cipher block chaining (AES-CBC) mode encryption. This mode is selected by writing ALGOMODE to 0b101 and ALGODIR to 0 in CRYP_CR.

Figure 265. AES-CBC mode encryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. IVx=[IVxR IVxL], R=right, L=left.
3. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

In this mode the first input block (I_1) obtained after bit/byte/half-word swapping is formed by exclusive-ORing the first plaintext data block (P_1) with a 128-bit initialization vector IV (I_1 = IV $\oplus$ P_1). The input block is processed through the AEA in the encrypt state using the 128-, 192- or 256-bit key (K0...K3). The resultant 128-bit output block (O_1) is used directly as ciphertext (C_1), that is, $C_1 = O_1$. This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I_2) = ($C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the AEA to produce the second ciphertext block. This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted.

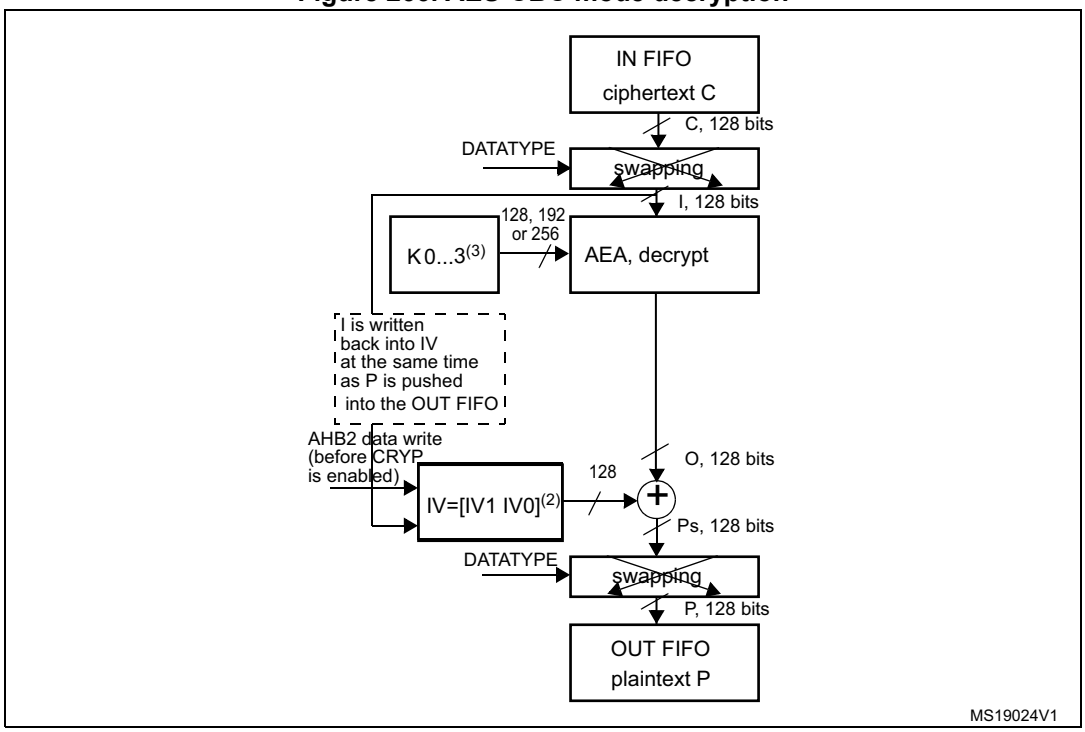
If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application, as explained in *Section 34.3.8: CRYP stealing and data padding*.

For more information on data swapping, refer to *Section 34.3.16: CRYP data registers and data swapping*.

AES CBC decryption

Figure 266 illustrates the AES Cipher block chaining (AES-CBC) mode decryption. This mode is selected by writing ALGOMODE to 0b101 and ALGODIR to 1 in CRYP_CR.

Figure 266. AES-CBC mode decryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. IVx=[IVxR IVxL], R=right, L=left.
3. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

In CBC mode, like in ECB mode, the secret key must be prepared to perform an AES decryption. Refer to *Section 34.3.7: Preparing the CRYP AES key for decryption* for more details on how to prepare the key.

When the key preparation process is complete, the decryption proceeds as follow: the first 128-bit ciphertext block (C_1) is used directly as the input block (I_1). The input block is processed through the AEA in the decrypt state using the 128-, 192- or 256-bit key. The resulting output block is exclusive-ORed with the 128-bit initialization vector IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus$ IV).

The second ciphertext block is then used as the next input block and is processed through the AEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). Note that P_2 and O_2 refer to the second block of data. The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

Ciphertext representing a partial data block must be decrypted in a manner specified for the application, as explained in *Section 34.3.8: CRYP stealing and data padding*.

For more information on data swapping, refer to *Section 34.3.16: CRYP data registers and data swapping*.

AES suspend/resume operations in ECB/CBC modes

Before interrupting the current message, the user application must respect the following sequence:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR) and the BUSY bit is cleared.
3. If DMA is used, stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the CRYP by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2] in the CRYP_CR register). If ECB mode is not selected, save the initialization vector registers, because CRYP_IVx registers have changed from initial values during the data processing.

Note: Key registers do not need to be saved as the original key value is known by the application.

5. If DMA is used, save the DMA controller status (such as pointers to IN and OUT data transfers, number of remaining bytes).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be set to 0).
3. Configure the cryptographic processor again with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. For AES-ECB or AES-CBC decryption, the key must be prepared again, as described in *Section 34.3.7: Preparing the CRYP AES key for decryption*.
5. If ECB mode is not selected, restore CRYP_IVx registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again the DMA requests from the cryptographic processor, by setting DIEN and DOEN bits to 1 in the CRYP_DMACR register.

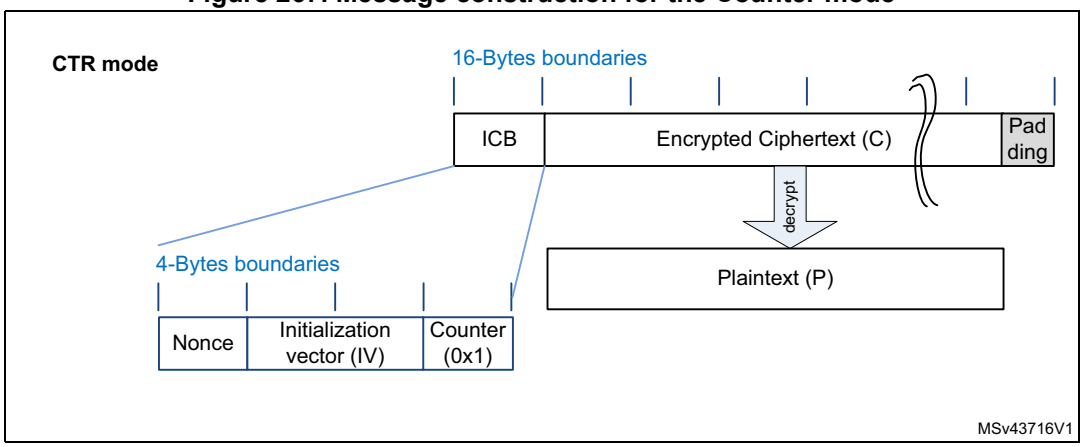
34.3.12 CRYP AES counter mode (AES-CTR)

Overview

The AES counter mode (CTR) uses the AES block as a key stream generator. The generated keys are then XORed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.* A typical message construction in CTR mode is given in *Figure 267*.

Figure 267. Message construction for the Counter mode



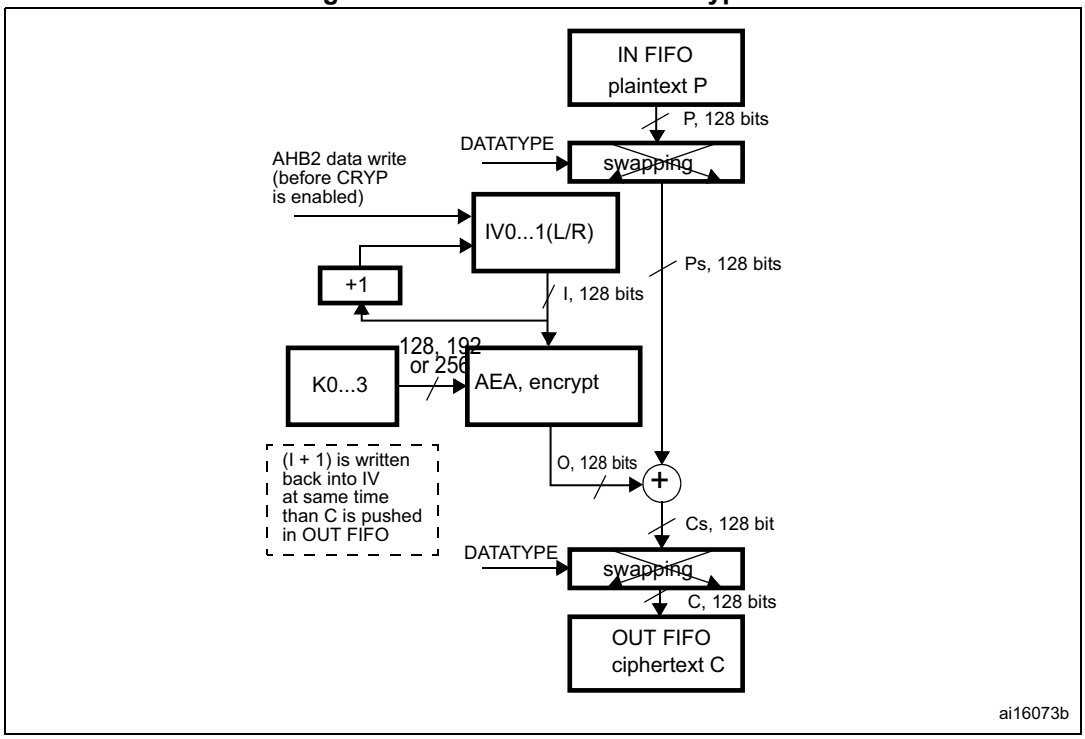
The structure of this message is as below:

- A 16-byte Initial Counter Block (ICB), composed of three distinct fields:
  - A *nonce:* a 32-bit, single-use value (i.e. a new nonce should be assigned to each new communication).
  - The *initialization vector* (IV): a 64-bit value that must be unique for each execution of the mode under a given key.
  - The *counter:* a 32-bit big-endian integer that is incremented each time a block has been processed. The initial value of the counter should be set to 1.
- The plaintext (P) is both authenticated and encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

AES CTR processing

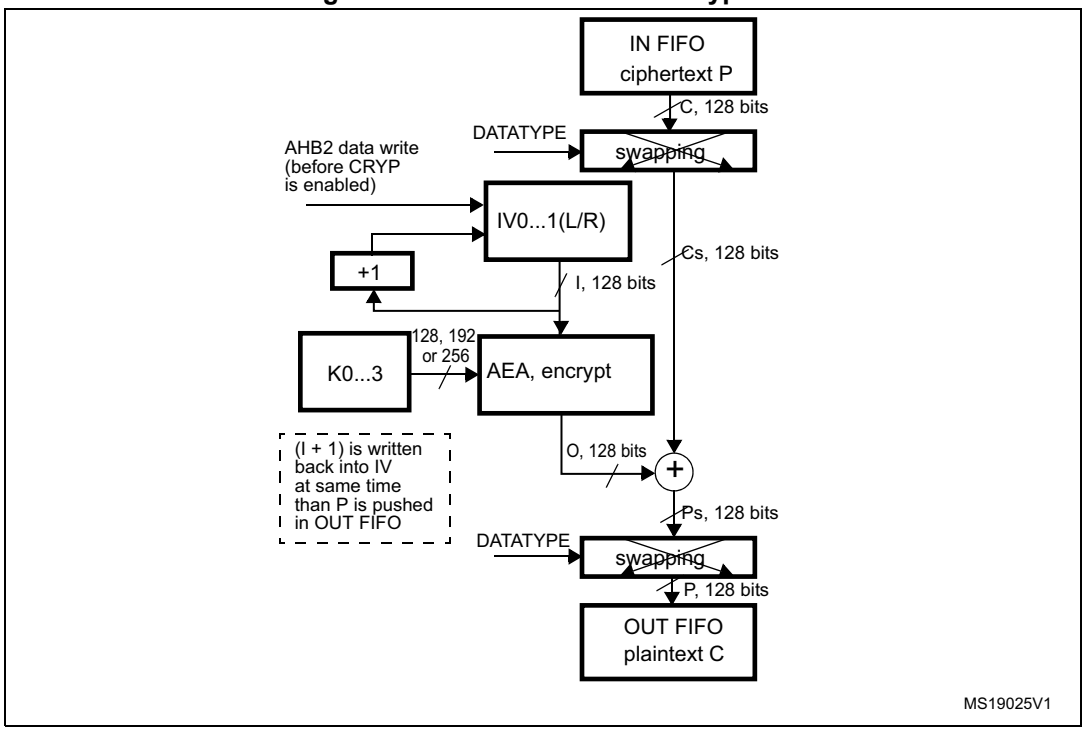
Figure 268 (respectively *Figure 269*) describes the AES-CTR encryption (respectively decryption) process implemented within this peripheral. This mode is selected by writing in ALGOMODE bitfield to 0b110 in CRYP_CR.

Figure 268. AES-CTR mode encryption



1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

Figure 269. AES-CTR mode decryption



1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

In CTR mode, the output block is XORed with the subsequent input block before it is input to the algorithm. Initialization vectors in the peripheral must be initialized as shown on *Table 273*.

Table 273. Counter mode initialization vector

CRYP_IV1R[31:0]	CRYP_IV1L[31:0]	CRYP_IV0R[31:0]	CRYP_IV0L[31:0]
nonce	IV[63:32]	IV[31:0]	32-bit counter= 0x1

Unlike in CBC mode, which uses the CRYP_IVx registers only once when processing the first data block, in CTR mode IV registers are used for processing each data block, and the peripheral increments the least significant 32 bits (leaving the other most significant 96 bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that will be XORed with the plaintext or cipher as input. Thus when ALGOMODE is set to 0b110, ALGODIR is don't care.

Note: In this mode the key must NOT be prepared for decryption.

The following sequence must be used to perform an encryption or a decryption in CTR chaining mode:

1. Make sure the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.
2. Configure CRYP_CR as follows:
   a) Program ALGOMODE bits to 0b110 to select CTR mode.
   b) Configure the data type (1, 8, 16 or 32 bits) through the DATATYPE bits.
3. Initialize the key registers (128,192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) as described in *Table 273*.
4. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_CR register.
5. If it is the last block, optionally pad the data with zeros to have a complete block.
6. Append data in the cryptographic processor and read the result. The three possible scenarios are described in *Section 34.3.5: CRYP procedure to perform a cipher operation*.
7. Repeat the previous step until the second last block is processed. For the last block, execute the two previous steps. For this last block, the driver must discard the data that is not part of the data when the last block size is less than 16 bytes.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message which was interrupted. Detailed CBC sequence can be found in *Section 34.3.11: CRYP AES basic chaining modes (ECB, CBC)*.

Note: Like for CBC mode, IV registers must be reloaded during the resume operation.

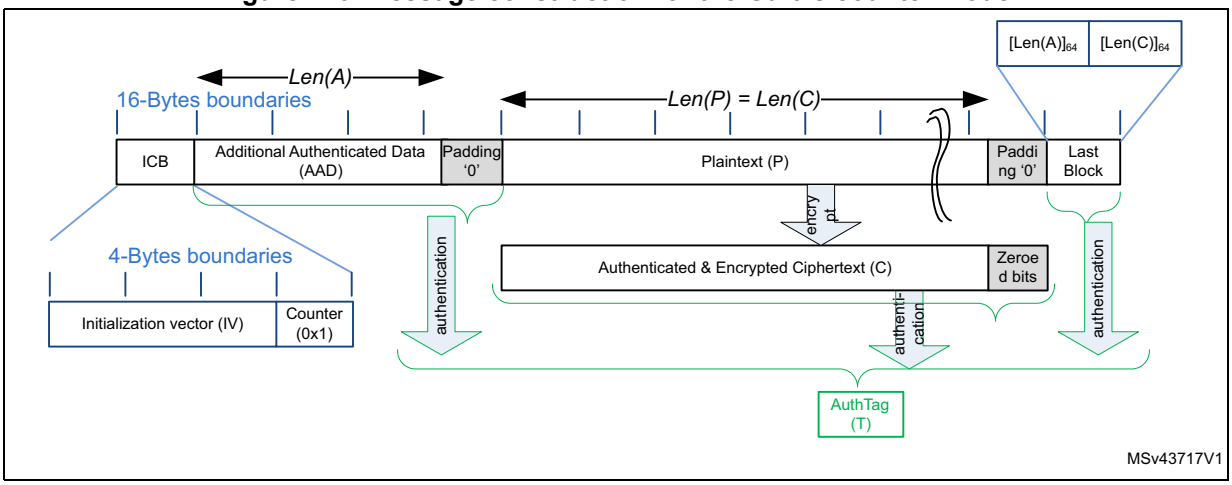
34.3.13 CRYP AES Galois/counter mode (GCM)

Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating the plaintext, and generating the correspondent ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.* A typical message construction in GCM mode is given in *Figure 270*.

Figure 270. Message construction for the Galois/counter mode



The structure of this message is defined as below:

- A 16-byte Initial Counter Block (ICB), composed of two distinct fields:
  - The *initialization vector* (IV): a 96-bit value that must be unique for each execution of the mode under a given key. Note that the GCM standard supports IV that are shorter than 96-bit, but in this case strict rules apply.
  - The *counter:* a 32-bit big-endian integer that is incremented each time a block has been processed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- The authenticated header A (also knows as Additional Authentication Data) has a known length *Len(A)* that can be non-multiple of 16 bytes and cannot exceed 2^{64}-1 bits. This part of the message is only authenticated, not encrypted.
- The plaintext message (P) is both authenticated and encrypted as ciphertext C, with a known length *Len(P)* that can be non-multiple of 16 bytes, and cannot exceed 2^{32} -2 blocks of 128-bits.

Note: GCM standard specifies that ciphertext C has same bit length as the plaintext P.

- When a part of the message (AAD or P) has a length which is non-multiple of 16 bytes, a special padding scheme is required.
- The last block is composed of the length of A (on 64 bits) and the length of ciphertext C (on 64 bits) as shown in *Table 274*.

Table 274. GCM last block definition

Endianness	Bit[0]	Bit[32]	Bit[64]	Bit[96]
Input data	0x0	Header length[31:0]	0x0	Payload length[31:0]

AES GCM processing

This mode is selected by writing ALGOMODE bitfield to 0b110 in CRYP_CR.

The mechanism for the confidentiality of the plaintext in GCM mode is a variation of the Counter mode, with a particular 32-bit incrementing function that generates the necessary sequence of counter blocks.

CRYP_IV registers are used for processing each data block. The cryptographic processor automatically increments the 32 least signification bits of the counter block. The first counter block (CB1) written by the application is equal to the Initial Counter Block incremented by one (see *Table 275*).

Table 275. GCM mode IV registers initialization

Register	CRYP_IV1R[31:0]	CRYP_IV1L[31:0]	CRYP_IV0R[31:0]	CRYP_IV0L[31:0]
Input data	ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter= 0x2

Note: In this mode the key must NOT be prepared for decryption.

The authentication mechanism in GCM mode is based on a hash function, called *GF2mul*, that performs multiplication by a fixed parameter, called the hash subkey (H), within a binary Galois field.

To process a GCM message, the driver must go through four phases, which are described in the following subsections.

- The Init phase: the peripheral prepares the GCM hash subkey (H) and performs the IV processing
- The Header phase: the peripheral processes the Additional Authenticated Data (AAD), with hash computation only.
- The Payload phase: the peripheral processes the plaintext (P) with hash computation, keystream encryption and data XORing. It operates in a similar way for ciphertext (C).
- The Final phase: the peripheral generates the authenticated tag (T) using the data last block.

1. **GCM init phase**

   During this first step, the GCM hash subkey (H) is calculated and saved internally to be used for processing all the blocks. It is recommended to follow the sequence below:

   a) Make sure the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.

   b) Select the GCM chaining mode by programming ALGOMODE bits to 0b01000 in CRYP_CR.

   c) Configure GCM_CCMPH bits to 0b00 in CRYP_CR to indicate that the init phase is ongoing.

   d) Initialize the key registers (128, 192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) as defined in *Table 275*.

   e) Set CRYPEN bit to 1 to start the calculation of the hash key.

   f) Wait for the CRYPEN bit to be cleared to 0 by the cryptographic processor, before moving on to the next phase.

2. **GCM header phase**

   The below sequence shall be performed after the GCM init phase. It must be complete before jumping to the payload phase. The sequence is identical for encryption and decryption.

   g) Set the GCM_CCMPH bits to 0b01 in CRYP_CR to indicate that the header phase is ongoing.

   h) Set the CRYPEN bit to 1 to start accepting data.

   i) If it is the last block of additional authenticated data, optionally pad the data with zeros to have a complete block.

   j) Append additional authenticated data in the cryptographic processor. The three possible scenarios are described in *Section 34.3.5: CRYP procedure to perform a cipher operation*.

   k) Repeat the previous step until the second last additional authenticated data block is processed. For the last block, execute the two previous steps. Once all the additional authenticated data have been supplied, wait until the BUSY flag is cleared before moving on to the next phase.

Note: This phase can be skipped if there is no additional authenticated data, i.e. Len(A)=0.

In header and payload phases, CRYPEN bit is not automatically cleared by the cryptographic processor.

3. **GCM payload phase (encryption or decryption)**

When the payload size is not null, this sequence must be executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the CRYP_DOUT register.

l) Set the CRYPEN bit to 0.

m) Configure GCM_CCMPH to 0b10 in the CRYP_CR register to indicate that the payload phase is ongoing.

n) Select the algorithm direction (0 for encryption, 1 for decryption) through the ALGODIR bit in CRYP_CR.

o) Set the CRYPEN bit to 1 to start accepting data.

p) If it is the last block of cleartext or plaintext, optionally pad the data with zeros to have a complete block. For encryption, refer to *Section 34.3.8: CRYP stealing and data padding* for more details.

q) Append payload data in the cryptographic processor, and read the result. The three possible scenarios are described in *Section 34.3.5: CRYP procedure to perform a cipher operation*.

r) Repeat the previous step until the second last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, the driver must discard the bits that are not part of the cleartext or the ciphertext when the last block size is less than 16 bytes. Once all payload data have been supplied, wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload data, i.e. Len(C)=0 (see GMAC mode).

4. **GCM final phase**

In this last step, the cryptographic processor generates the GCM authentication tag and stores it in CRYP_DOUT register.

s) Configure GCM_CCMPH[1:0] to 0b11 in CRYP_CR to indicate that the Final phase is ongoing. Set the ALGODIR bit to 0 in the same register.

t) Write the input to the CRYP_DIN register four times. The input must be composed of the length in bits of the additional authenticated data (coded on 64 bits) concatenated with the length in bits of the payload (coded of 64 bits), as show in *Table 274*.

Note: In this final phase data have to be swapped according to the DATATYPE programmed in CRYP_CR register.

u) Wait until the OFNE flag (FIFO output not empty) is set to 1 in the CRYP_SR register.

v) Read the CRYP_DOUT register 4four times: the output corresponds to the authentication tag.

w) Disable the cryptographic processor (CRYPEN bit = 0 in CRYP_CR)

x) If an authenticated decryption is being performed, compare the generated tag with the expected tag passed with the message.

Suspend/resume operations in GCM mode

Before interrupting the current message in header or payload phase, the user application must respect the following sequence:

1. If DMA is used, stop DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
3. If DMA is used, stop DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2], bits [17:16] and bits 19 of the CRYP_CR register). In addition, save the initialization vector registers, since CRYP_IVx registers have changed from their initial values during data processing.

Note: Key registers do not need to be saved as original their key value is known by the application.

5. Save context swap registers: CRYP_CSGCMCCM0..7 and CRYP_CSGCM0..7
6. If DMA is used, save the DMA controller status (pointers to IN and OUT data transfers, number of remaining bytes, etc.).

To resume message processing, the user must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be 0).
3. Configure again the cryptographic processor with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. Restore context swap registers: CRYP_CSGCMCCM0..7 and CRYP_CSGCM0..7
5. Restore CRYP_IVx registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again cryptographic processor DMA requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

Note: In Header phase, DMA OUT FIFO transfer is not used.

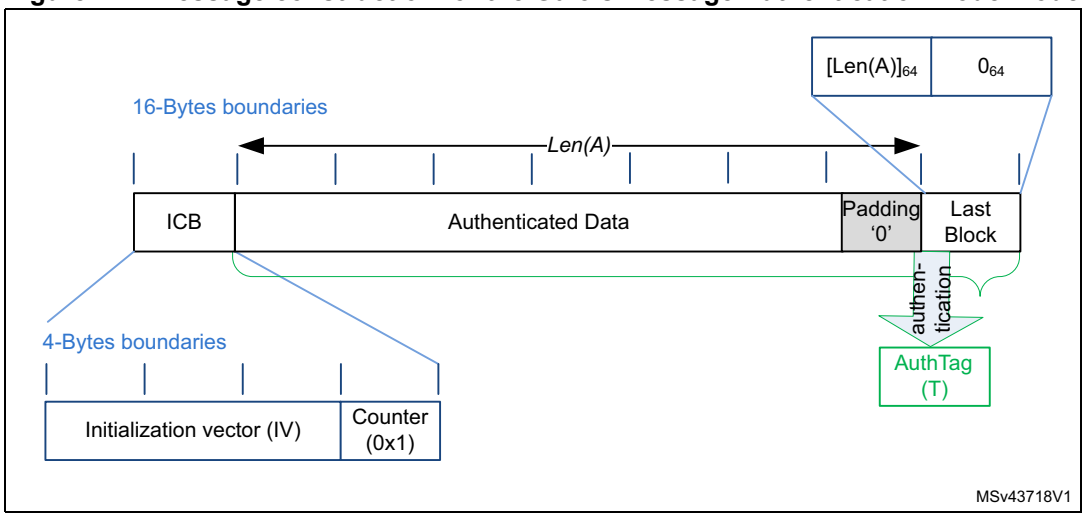
34.3.14 CRYP AES Galois message authentication code (GMAC)

Overview

The Galois message authentication code (GMAC) allows authenticating a plaintext and generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

A typical message construction in GMAC mode is given in *Figure 271*.

Figure 271. Message construction for the Galois Message Authentication Code mode



AES GMAC processing

This mode is selected by writing ALGOMODE bitfield to 0b110 in CRYP_CR.

GMAC algorithm corresponds to the GCM algorithm applied on a message composed only of an header. As a consequence, all steps and settings are the same as in GCM mode, except that the payload phase (3) is not used.

Suspend/resume operations in GMAC

GMAC is exactly the same as GCM algorithm except that only header phase (2) can be interrupted.

34.3.15 CRYP AES Counter with CBC-MAC (CCM)

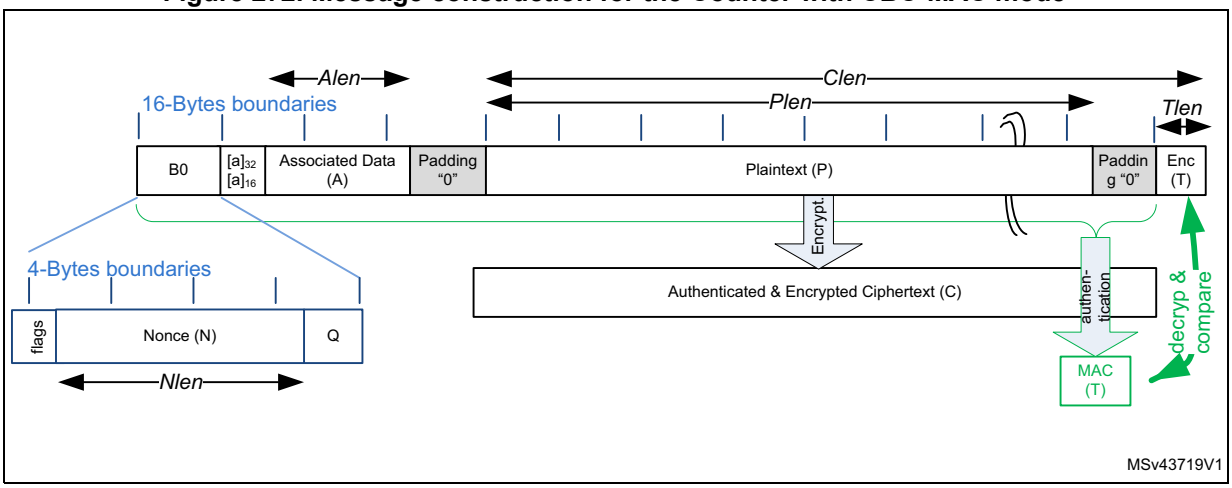
Overview

The AES Counter with Cipher Block Chaining-Message Authentication Code (CCM) algorithm allows encrypting and authenticating the plaintext, and generating the correspondent ciphertext and tag (also known as message authentication code). To ensure confidentiality, CCM algorithm is based on AES counter mode. It uses Cipher Block Chaining technique to generate the message authentication code. This is commonly called CBC-MAC

Note: NIST does not approve this CBC-MAC as an authentication mode outside of the context of the CCM specification.

CCM chaining is specified in NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*. A typical message construction in CCM mode is given in *Figure 272*

Figure 272. Message construction for the Counter with CBC-MAC mode



The structure of this message is as below:

- One 16-byte first authentication block (called B0 by the standard), composed of three distinct fields:
  - *Q*: a bit string representation of the byte length of P (Plen)
  - A *nonce* (N)*:* single-use value (i.e. a new nonce should be assigned to each new communication). Size of nonce *Nlen* + size of *Plen* shall be equal to 15 bytes.
  - *Flags*: most significant byte containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values t (MAC length expressed in bytes) and *q* (plaintext length such as Plen<2^{8q} bytes). Note that the counter blocks range associated to q is equal to 2^{8q-4}, i.e. if q maximum value is 8, the counter blocks used in cipher shall be on 60 bits.

Note: The cryptographic peripheral can only manage padded plaintext/ciphertext messages of length Plen < 2^{36} +1 bytes.

- 16-bytes blocks (B) associated to the Associated Data (A).

  This part of the message is only authenticated, not encrypted. This section has a known length, *ALen,* that can be a non-multiple of 16 bytes (see *Figure 272*). The standard also states that, on the MSB bits of the first message block (B1), the associated data length expressed in bytes (*a*) must be encoded as defined below:
  - If $0 < a < 2^{16}-2^8$, then it is encoded as $[a]_{16}$, i.e. two bytes.
  - If $2^{16}-2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, i.e. six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, i.e. ten bytes.
- 16-byte blocks (B) associated to the plaintext message (P), which is both authenticated and encrypted as ciphertext C, with a known length of *Plen*. This length can be a non-multiple of 16 bytes (see *Figure 272*) but cannot exceed 2^{32} blocks of 128-bit.
- The encrypted MAC (T) of length *Tlen* appended to the ciphertext C of overall length *Clen*.
- When a part of the message (A or P) has a length which is a non-multiple of 16 bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (i.e. no payload).

As an example, the C.1 section in *NIST Special Publication 800-38C* gives the following:

N: **10111213 141516** (Nlen= 56 bits or 0x7 bytes)

A: ***00010203 04050607*** (Alen= 64 bits or 0x8 bytes)

P: *20212223* (Plen= 32 bits i.e. Q= 0x*4* bytes)

T: 6084341b (Tlen= 32 bits or t= 4)

B0: 4f**101112 13141516** 00000000 00000004
B1: 0008***0001 02030405 0607***0000 00000000
B2: *20212223* 00000000 00000000 00000000

CTR0: 07**10111213 141516** 00000000 00000000

CTR1: 07**10111213 141516** 00000000 00000001

The usage of control blocks CTRx is explained in the following section. The generation of CTR0 from the first block (B0) must be managed by software.

AES CCM processing

This mode is selected by writing ALGOMODE bitfield to 0b1001 in CRYP_CR.

The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted data to generate a MAC, whose length is known. Counter mode encryption, which requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting data, called the ciphertext C, is the output of the generation-encryption process on plaintext P.

CRYP_IV registers are used for processing each data block. The cryptographic processor automatically increments the CTR counter with a bit length defined by the first block (B0). The first counter written by application, CTR1, is equal to B0 with the first 5 bits zeroed and the most significant bits containing P byte length also zeroed, then incremented by one (see *Table 276*).

Table 276. CCM mode IV registers initialization

Register	CRYP_IV0L[31:0]	CRYP_IV0R[31:0]	CRYP_IV1L[31:0]	CRYP_IV1R[31:0]
Endianness	IV[0:31]	IV[32:63]	IV[64:95]	IV[96:127]
Input data	B0[31:0], where the 5 most significant bits are set to 0 (flag bits)	B0[63:32]	B0[95:64]	B0[127:96], where Q length bits are set to 0, except for bit 0 that is set to 1

Note: In this mode, the key must NOT be prepared for decryption.

To process a CCM message, the driver must go through four phases, which are described below.

- The Init phase: the peripheral processes the first block and prepares the first counter block.
- The Header phase: the peripheral processes the Associated data (A), with hash computation only.
- The Payload phase: the peripheral processes the plaintext (P), with hash computation, counter block encryption and data XORing. It operates in a similar way for ciphertext (C).
- The Final phase: the peripheral generates the message authentication code (MAC).

1. **CCM init phase**

   In this first step, the first block (B0) of the CCM message is programmed into the CRYP_DIN register. During this phase, the CRYP_DOUT register does not contain any output data. It is recommended to follow the sequence below:

   a) Make sure that the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.

   b) Select the CCM chaining mode by programming the ALGOMODE bits to 0b01001 in the CRYP_CR register.

   c) Configure the GCM_CCMPH bits to 0b00 in CRYP_CR to indicate that we are in the init phase.

   d) Initialize the key registers (128, 192 and 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) with CTR1 information, as defined in *Table 276*.

   e) Set the CRYPEN bit to 1 in CRYP_CR to start accepting data.

   f) Write the B0 packet into CRYP_DIN register, then wait for the CRYPEN bit to be cleared to 0 by the cryptographic processor before moving on to the next phase.

Note: In this init phase data have to be swapped according to the DATATYPE programmed in CRYP_CR register.

2. **CCM header phase**

   The below sequence shall be performed after the CCM Init phase. It must be complete before jumping to the payload phase. The sequence is identical for encryption and decryption. During this phase, the CRYP_DOUT register does not contain any output data.

   g) Set the GCM_CCMPH bit to 0b01 in CRYP_CR to indicate that the header phase is ongoing.

   h) Set the CRYPEN bit to 1 to start accepting data.

   i) If it is the last block of associated data, optionally pad the data with zeros to have a complete block.

   j) Append the associated data in the cryptographic processor. The three possible scenarios are described in *Section 34.3.5: CRYP procedure to perform a cipher operation*.

   k) Repeat the previous step until the second last associated data block is processed. For the last block, execute the two previous steps. Once all the additional authenticated data have been supplied, wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no associated data (Alen=0).

Note: The first block of the associated data B1 must be formatted with the associated data length. This task must be managed by the driver.

3. CCM payload phase (encryption or decryption)

When the payload size is not null, this sequence must be performed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the CRYP_DOUT register.

l) Set the CRYPEN bit to 0.

m) Configure GCM_CCMPH bits to 0b10 in CRYP_CR to indicate that the payload phase is ongoing.

n) Select the algorithm direction (0 for encryption, 1 for decryption) through the ALGODIR bit in CRYP_CR.

o) Set the CRYPEN bit to 1 to start accepting data.

p) If it is the last block of cleartext, optionally pad the data with zeros to have a complete block (encryption only). For decryption, refer to *Section 34.3.8: CRYP stealing and data padding* for more details.

q) Append payload data in the cryptographic processor, and read the result. The three possible scenarios are described in *Section 34.3.5: CRYP procedure to perform a cipher operation*.

r) Repeat the previous step until the second last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block of ciphertext (decryption only), the driver must discard the data that is not part of the cleartext when the last block size is less than 16 bytes. Once all payload data have been supplied, wait until the BUSY flag is cleared

Note: This phase can be skipped if there is no payload data, i.e. Plen=0 or Clen=Tlen

Note: Do not forget to remove $LSB_{Tlen}(C)$ encrypted tag information when decrypting ciphertext C.

4. CCM final phase

In this last step, the cryptographic processor generates the CCM authentication tag and stores it in the CRYP_DOUT register.

s) Configure GCM_CCMPH[1:0] bits to 0b11 in CRYP_CR to indicate that the final phase is ongoing and set the ALGODIR bit to 0 in the same register.

t) Load in CRYP_DIN, the CTR0 information which is described in *Table 276* with bit[0] set to 0.

Note: In this final phase, data have to be swapped according to the DATATYPE programmed in CRYP_CR register.

u) Wait until the OFNE flag (FIFO output not empty) is set to 1 in the CRYP_SR register.

v) Read the CRYP_DOUT register four times: the output corresponds to the encrypted CCM tag.

w) Disable the cryptographic processor (CRYPEN bit set to 0 in CRYP_CR)

x) If an authenticated decryption is being performed, compare the generated encrypted tag with the encrypted tag padded in the ciphertext, i.e. $LSB_{Tlen}(C)$= MSB_{Tlen}(CRYP_DOUT data).

Suspend/resume operations in CCM mode

Before interrupting the current message in payload phase, the user application must respect the following sequence:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM=1 and OFNE=0 in the CRYP_SR register) and the BUSY bit is cleared.
3. If DMA is used, stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2], bits [17:16] and bits 19 in the CRYP_CR register). In addition, save the initialization vector registers, since CRYP_IVx registers have changed from their initial values during the data processing.

Note: Key registers do not need to be saved as their original key value is known by the application.

5. Save context swap registers: CRYP_CSGCMCCM0..7
6. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, etc.).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (must be 0).
3. Configure the cryptographic processor again with the initial setting in CRYP_CR and key registers using the saved configuration.
4. Restore context swap registers: CRYP_CSGCMCCM0..7
5. Restore CRYP_IVx registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again cryptographic processor DMA requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

Note: In Header phase DMA OUT FIFO transfer is not used.

34.3.16 CRYP data registers and data swapping

Introduction

The CRYP_DIN register is the 32-bit wide data input register of the peripheral. It is used to enter into the input FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

The first word written into the FIFO is the LSB of the input block. The MSB of the input block is written at the end. CRYP_DIN data endianness can be described as below when DATATYPE=”00” (no data swapping):

- In the DES/TDES modes

  Bit 1 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word entered into the FIFO, bit 64 (rightmost bit) corresponds to the LSB (bit 0) of the second word entered into the FIFO.
- In the AES mode

  Bit 0 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word written into the FIFO, bit 127 (rightmost bit) corresponds to the LSB (bit 0) of the 4th word written into the FIFO.

Similarly CRYP_DOUT register is the 32-bit wide data out register of the peripheral. It is a read-only register that is used to retrieve from the output FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

Like for the input data, the LSB of the output block is the first word read from the output FIFO. The MSB of the output block is read at the end. CRYP_DOUT data endianness can be described as below when DATATYPE=”00” (no data swapping):

- In the DES/TDES modes

  Bit 1 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 64 (rightmost bit) corresponds to the LSB (bit 0) of the second word read from the FIFO.
- In the AES mode

  Bit 0 (leftmost bit) of the data block corresponds to the MSB (bit 31) of the first word read from the FIFO, bit 127 (rightmost bit) corresponds to the LSB (bit 0) of the 4th word read from the FIFO.

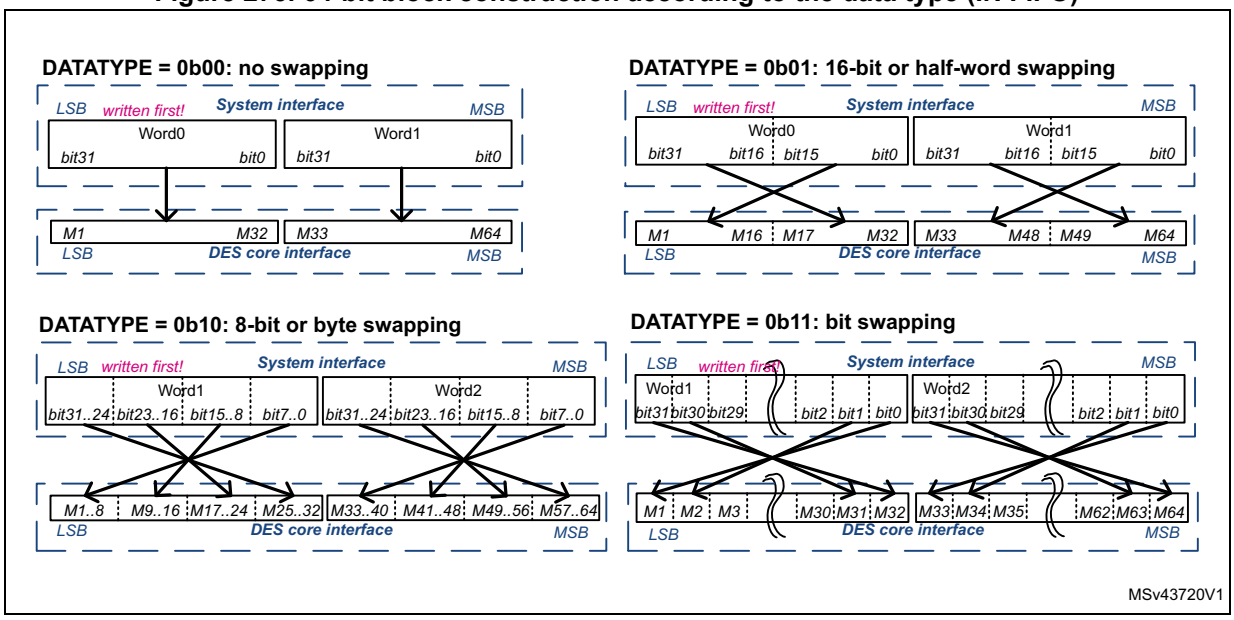
DES/TDES data swapping feature

Depending on the type of data to be processed (e.g. byte swapping when data are ASCII text stream), a bit, byte, half-word or no swapping operation must be done on the data read from the input FIFO before entering the little-endian DES processing core. The same swapping must be performed on the data produced by the little-endian DES processing core before they are written to the output FIFO.

Figure 273 shows how the DES processing core 64-bit data block M1...64 is constructed from two consecutive 32-bit words popped into IN FIFO by the driver. This is done according to the DATATYPE bitfield in the CRYP_CR register.

Note: The same swapping is performed between the IN FIFO and the CRYP data block, and between the CRYP data block and the OUT FIFO.

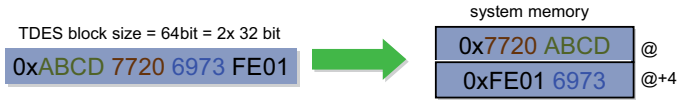
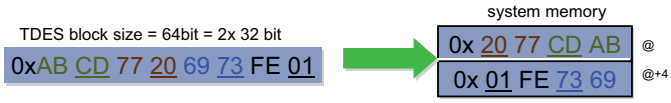
Figure 273. 64-bit block construction according to the data type (IN FIFO)



Note: The CRYP Key registers (CRYP_Kx(L/R)) and initialization registers (CRYP_IVx(L/R)) are not sensitive to the swap mode selected. They have a fixed little-endian configuration (refer to Section 34.3.17 and Section 34.3.18, respectively).

A typical example of data swapping is given in *Table 277*.

Table 277. DES/TDES data swapping feature

DATATYPE in CRYP_CR	Swapping performed	Data block representation (64-bit) 0xABCD7720 6973FE01 System memory data (plaintext or cypher)
0b00	No swapping	Address @: 0xABCD7720 (LSB, written first) Address @+4: 0x6973FE01
0b01	Half-word (16-bit) swapping	Address @: 0x7720ABCD (swapped LSB, written first) Address @+4: 0xFE016973 
0b10	Byte (8-bit) swapping	Address @: 0x2077CDAB (swapped LSB, written first) Address @+4: 0x01FE7369 
0b11	Bit swapping	LSB data word: 0xABCD7720 0b1010 1011 1100 1101 0111 0111 0010 0000 MSB data word: 0x6973FE01 0b0110 1001 0111 0011 1111 1110 0000 0001 Address @: 0x04EEB3D5 (swapped LSB, written first) Address @+4: 0x807FCE96

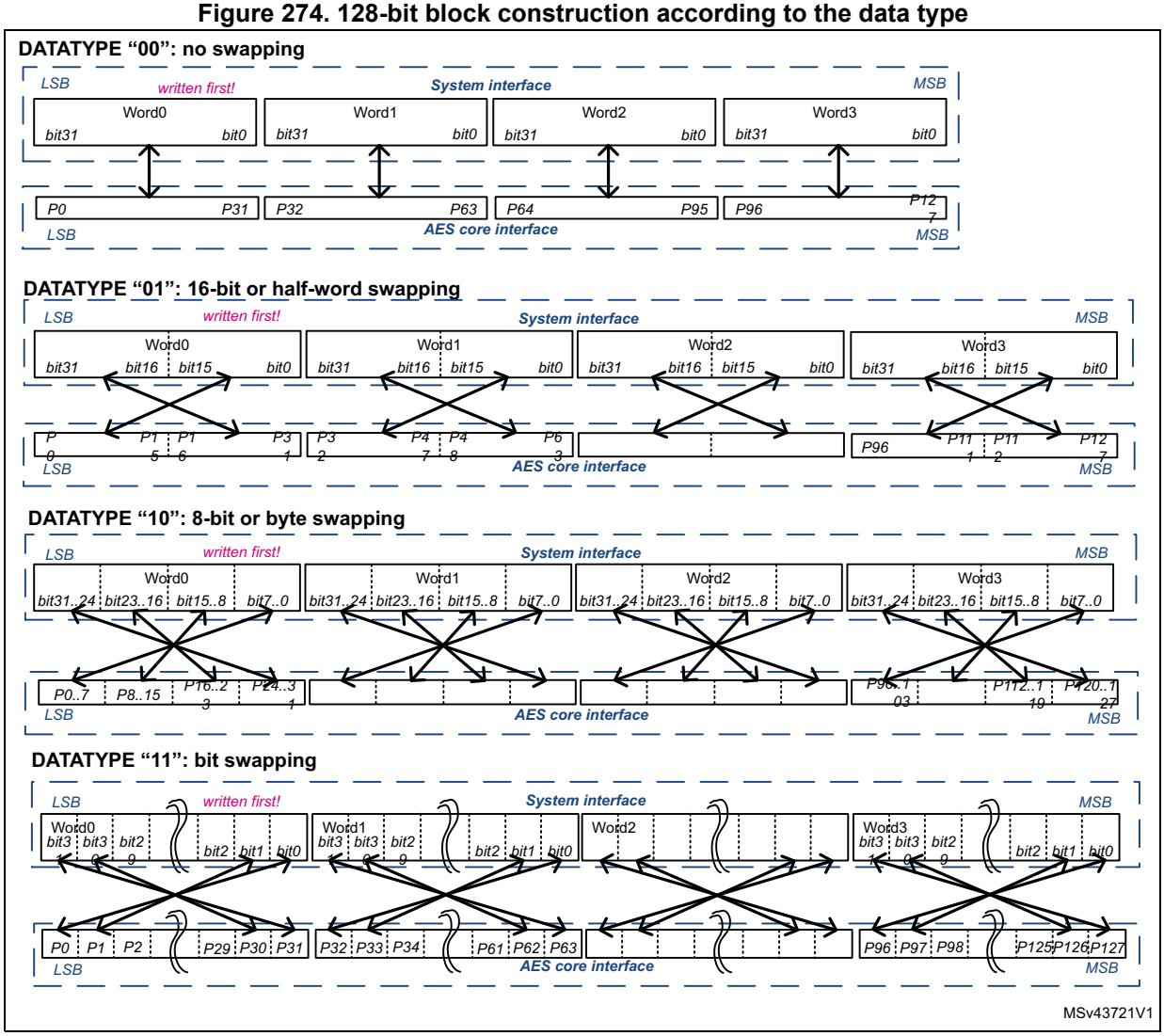
AES data swapping feature

Depending on the type of data to be processed (e.g. byte swapping when data are ASCII text stream), a bit, byte, half-word or no swapping operation must be done on data read from the input FIFO before entering the little-endian AES processing core. The same swapping must be performed on the data produced by the little-endian AES processing core before they are written to the output FIFO.

Figure 274 shows how the AES processing core 128-bit data block P0..127 is constructed from four consecutive 32-bit words written by the driver to the CRYP_DIN register. This is done according to the DATATYPE bitfield in the CRYP control register (CRYP_CR).

Note: The same swapping is performed between the CRYP_DIN and the CRYP data block, and between the CRYP data block and the CRYP_DOUT.

Figure 274. 128-bit block construction according to the data type



Note: The swapping operation concerns only the CRYP_DOUT and CRYP_DIN registers. The CRYP_KxL/KxR and CRYP_IVxL/IVxR registers are not sensitive to the swap mode selected.They have a fixed little-endian configuration (refer to Section 34.3.17 and Section 34.3.18).

Typical examples of data swapping are given in *Table 278*.

Table 278. AES data swapping feature

DATATYPE in CRYP_CR	Swapping performed	Data block representation (64-bit) 0x4E6F7720 69732074
		System memory data (little-endian)
0b00	No swapping	Address @: 0x4E6F7720 (LSB, written first) Address @+4: 0x69732074
0b01	Half-word (16-bit) swapping	Address @: 0x77204E6F (swapped LSB, written first) Address @+4: 0x20746973
0b10	Byte (8-bit) swapping	Address @: 0x20776F4E (swapped LSB, written first) Address @+4: 0x74207369
0b11	Bit swapping	LSB data word: 0x4E6F7720 0b0100 1110 0110 1111 0111 0111 0010 0000 MSB data word: 0x69732074 0b0110 1001 0111 0011 0010 0000 0111 0100 Address @: 0x4EEF672 (swapped LSB, written first) Address @+4: 0x2E04CE96

34.3.17 CRYP key registers

The CRYP_Kx registers are used to store the encryption or decryption keys.

They are organized as eight registers in a little-endian configuration, as shown in *Table 279*.

Table 279. Key registers CRYP_KxR/LR endianness (TDES K1/2/3 and AES 128/192/256-bit keys)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	K1[1:32]	K1[33:64]	K2[1:32]	K2[33:64]	K2[1:32]	K2[33:64]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
-	-	-	-	k[0:31]	k[32:63]	k[64:95]	k[96:127]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
-	-	k[0:31]	k[32:63]	k[64:95]	k[96:127]	k[128:159]	k[160:191]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
k[0:31]	k[32:63]	k[64:95]	k[96:127]	k[128:159]	k[160:191]	k[192:223]	k[224:255]

Note: DES/TDES keys include 8-bit parity information that are not used by the cryptographic processor. In other words, bits 8, 16, 24, 32, 40, 48, 56 and 64 of each 64-bit key value Kx[1:64] are not used.

Keys are considered as four 64-bit data items. They therefore do not have the same data format and representation in system memory as plaintext or ciphertext data.

Any write operation to the CRYP_Kx(L/R) registers when the BUSY bit is set to 1 in the CRYP_SR register is disregarded (i.e. register content not modified). Thus, the software must check that the BUSY equals 0 before modifying key registers.

Key registers are not affected by the data swapping feature controlled by DATATYPE value in CRYP_CR register.

Refer to *Section 34.6: CRYP registers* for a detailed description of CRYP_Kx(L/R) registers.

34.3.18 CRYP initialization vector registers

The CRYP_IVxL/IVxR registers are used to store the initialization vector or the nonce, depending on the chaining mode selected. When used, these registers are updated by the core after each computation round of the TDES or AES core.

They are organized as four registers in a little-endian configuration, as shown in *Table 280*.

Table 280. Initialization vector registers CRYP_IVxR endianness

CRYP_IV1R[31:0]	CRYP_IV1L[31:0]	CRYP_IV0R[31:0]	CRYP_IV0L[31:0]
IV[96:127]	IV[64:95]	IV[32:63]	IV[0:31]

Initialization vector registers are considered as two 64-bit data items. They therefore do not have the same data format and representation in system memory as plaintext or ciphertext data.

Any write operation to the CRYP_IV0...1(L/R) registers when the BUSY bit is set to 1 in the CRYP_SR register is disregarded (i.e. register content not modified). Therefore, the software must check that the BUSY equals 0 in the CRYP_SR register before modifying initialization vectors.

Reading the CRYP_IV0...1(L/R) register returns the latest counter value (useful for managing suspend mode) except for CCM/GCM.

Note: In DES/TDES mode, only CRYP_IV0x are used.

Initialization vector registers are not affected by the data swapping feature controlled by DATATYPE value in CRYP_CR register.

Refer to *Section 34.6: CRYP registers* for a detailed description of CRYP_IVxL/IVxR registers.

34.3.19 CRYP DMA interface

The cryptographic processor provides an interface to connect to the DMA (Direct Memory Access) controller. The DMA operation is controlled through the CRYP DMA control register (CRYP_DMACR).

Data input using DMA

DMA can be enabled for writing data into the cryptographic peripheral by setting the DIEN bit in the CRYP_DMACR register. When this bit is set, the cryptographic processor initiates a DMA request during the INPUT phase each time it requires a word to be written to the CRYP_DIN register.

Table 281 shows the recommended configuration to transfer data from memory to cryptographic processor through the DMA controller.

Table 281. Cryptographic processor configuration for memory-to-peripheral DMA transfers

DMA channel control register field	Programming recommendation
Transfer size	Message length, multiple of 128-bit. This 128-bit granularity corresponds to two blocks for DES, one block for AES. According to the algorithm and the mode selected, special padding/ ciphertext stealing might be required. Refer to *Section 34.3.8: CRYP stealing and data padding for details*.
Source burst size (memory)	CRYP FIFO_size /2 /transfer_width = **4**
Destination burst size (peripheral)	CRYP FIFO_size /2 /transfer_width = **4** (FIFO_size= 8x32-bit, transfer_width= 32-bit)
DMA FIFO size	CRYP FIFO_size /2 = 16 bytes
Source transfer width (memory)	32-bit words
Destination transfer width (peripheral)	32-bit words
Source address increment (memory)	Yes, after each 32-bit transfer.
Destination address increment (peripheral)	Fixed address of CRYP_DIN shall be used (no increment).

Data output using DMA

To enable the DMA for reading data from AES peripheral, set the DOEN bit in the CRYP_DMACR register. When this bit is set, the cryptographic processor initiates a DMA request during the OUTPUT phase each time it requires a word to be read from the CRYP_DOUT register.

Table 282 shows the recommended configuration to transfer data from cryptographic processor to memory through the DMA controller.

Table 282. Cryptographic processor configuration for peripheral to memory DMA transfers

DMA channel control register field	Programming recommendation
Transfer size	Message length, multiple of 128-bit. This 128-bit granularity corresponds to two blocks for DES, one block for AES. Depending on the algorithm used, extra bits have to be discarded.
Source burst size (peripheral)	When *DES* is used: Single transfer (burst size=**1**) When *AES* is used: CRYP FIFO_size /2 /transfer_width = **4** (FIFO_size= 8x32-bit, transfer_width= 32-bit)
Destination burst size (memory)	CRYP FIFO_size /2 /transfer_width = **4**
DMA FIFO size	CRYP FIFO_size /2 = 16 bytes
Source transfer width (peripheral)	32-bit words
memory transfer width (memory)	32-bit words
Source address increment (peripheral)	Fixed address of CRYP_DOUT shall be used (no increment).
Destination address increment (memory)	Yes, after each 32-bit transfer.

DMA mode

When AES is used, the cryptographic processor manages two DMA transfer requests through `cryp_in_dma` and `cryp_out_dma` internal input/output signals, which are asserted:

- for IN FIFO: every time a block has been read from FIFO by CRYP,
- for OUT FIFO: every time a block has been written into the FIFO by the cryptographic processor.

When DES is used, the cryptographic processor manages two DMA transfer requests through `cryp_in_dma` and `cryp_out_dma` internal input/output signals, which are asserted:

- for IN FIFO: every time two blocks have been read from FIFO by the cryptographic processor
- for OUT FIFO: every time a word has been written into the FIFO by the cryptographic processor (single transfer). Note that a burst transfer is also triggered when two blocks have been written into the FIFO.

All request signals are de-asserted if the cryptographic peripheral is disabled or the DMA enable bit is cleared (DIEN bit for the IN FIFO and DOEN bit for the OUT FIFO in the CRYP_DMACR register).

Caution: It is important that DMA controller empties the cryptographic peripheral output FIFO before filling up the CRYP input FIFO. To achieve it, the DMA controller should be configured so

that the transfer from the peripheral to the memory has a higher priority than the transfer from the memory to the peripheral.

For more detailed information on DMA operations, refer to *Section 34.3.5: CRYP procedure to perform a cipher operation*.

34.3.20 CRYP error management

No error flags are generated by the cryptographic processor.

34.4 CRYP interrupts

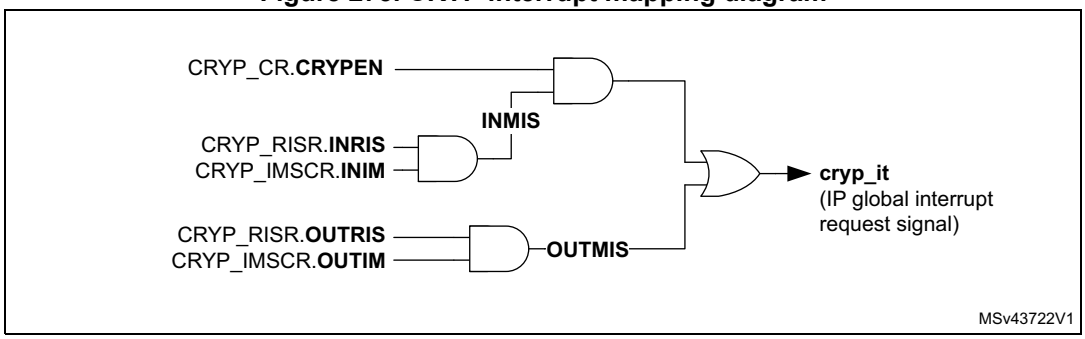
Overview

There are two individual maskable interrupt sources generated by the cryptographic processor to signal the following events:

- Input FIFO empty or not full
- Output FIFO full or not empty

These two sources are combined into a single interrupt signal which is the only interrupt signal from the CRYP peripheral that drives the NVIC (nested vectored interrupt controller). The interrupt logic is summarized on *Figure 275*.

Figure 275. CRYP interrupt mapping diagram



You can enable or disable CRYP interrupt sources individually by changing the mask bits in the CRYP_IMSCR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual maskable interrupt sources can be read either from the CRYP_RISR register, for raw interrupt status, or from the CRYP_MISR register for masked interrupt status. The status of the individual source of event flags can be read from the CRYP_SR register.

Table 283 gives a summary of the available features.

Table 283. CRYP interrupt requests

Interrupt event	Event flag (interrupt status)	Enable control bit	Event flag (source)
Output FIFO full	OUTRIS, OUTMIS	OUTIM and CRYPEN	OFFU
Output FIFO not empty			OFNE
Input FIFO not full	OUTRIS, OUTMIS	INIM and CRYPEN	IFNF
Input FIFO empty			IFEM

Output FIFO service interrupt - OUTMIS

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is when the interrupt follows the state of the output FIFO not empty flag OFNE).

The output FIFO service interrupt OUTMIS is NOT enabled with the CRYP enable bit. Consequently, disabling the CRYP will not force the OUTMIS signal low if the output FIFO is not empty.

Input FIFO service interrupt - INMIS

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is enabled with the CRYP enable bit. Consequently, when CRYP is disabled, the INMIS signal is low even if the input FIFO is empty.

34.5 CRYP processing time

The time required to process a 128-bit block for each mode of operation is summarized below.

Table 284. Processing time (in clock cycle) for ECB, CBC and CTR per 128-bit block

Algorithm / Key size	ECB	CBC	CTR
128b	14	14	14
192b	16	16	16
256b	18	18	18

Table 285. Processing time (in clock cycle) for GCM and CCM per 128-bit block

Algorithm / Key size	GCM					CCM				
	Init	Header	Payload	Tag	Total	Init	Header	Payload	Tag	Total
128b	24	10	14	14	**62**	12	14	25	14	**65**
192b	28	10	16	16	**70**	14	16	29	16	**75**
256b	32	10	18	18	**78**	16	18	33	18	**85**

34.6 CRYP registers

The cryptographic core is associated with several control and status registers, eight key registers and four initialization vectors registers.

34.6.1 CRYP control register (CRYP_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				ALGOMODE3	Res.	GCM_CCMPH[1:0]	
												rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE[1:0]		DATATYPE[1:0]		ALGOMODE[2:0]			ALGODIR	Res.	Res.
rw	w					rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:20 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **GCM_CCMPH[1:0]**: GCM or CCM Phase selection
This bitfield has no effect if GCM, GMAC or CCM algorithm is not selected in ALGOMODE field.
00: Init phase
01: Header phase
10: Payload phase
11: Final phase

Bit 15 **CRYPEN:** CRYP processor Enable
0: Cryptographic processor peripheral is disabled
1: Cryptographic processor peripheral is enabled
This bit is automatically cleared by hardware when the key preparation process ends (ALGOMODE= 0b111) or after GCM/GMAC or CCM init phase.

Bit 14 **FFLUSH:** CRYP FIFO Flush
0: No FIFO flush
1: FIFO flush enabled
When CRYPEN = 0, writing this bit to 1 flushes the IN and OUT FIFOs (i.e. read and write pointers of the FIFOs are reset). Writing this bit to 0 has no effect.
When CRYPEN = 1, writing this bit to 0 or 1 has no effect.
Reading this bit always returns 0.
FFLUSH bit has to be set only when BUSY=0. If not, the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a non-empty FIFO condition.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:8 **KEYSIZE[1:0]**: Key Size selection (AES mode only)

This bitfield defines the bit-length of the key used for the AES cryptographic core. This bitfield is ‘don’t care’ in the DES or TDES modes.

00: 128-bit key length
01: 192-bit key length
10: 256-bit key length
11: Reserved, do not use this value

Writing KEYSIZE bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.

Bits 7:6 **DATATYPE[1:0]**: Data Type selection

This bitfield defines the format of data written in CRYP_DIN or read from CRYP_DOUT registers. For more details refer to *Section 34.3.16: CRYP data registers and data swapping*).

00: 32-bit data. No swapping for each word. First word pushed into the IN FIFO (or popped off the OUT FIFO) forms bits 1...32 of the data block, the second word forms bits 33...64 etc.
01: 16-bit data, or half-word. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 2 half-words, which are swapped with each other.
10: 8-bit data, or bytes. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 4 bytes, which are swapped with each other.
11: bit data, or bit-string. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 32 bits (1st bit of the string at position 0), which are swapped with each other.

Writing DATATYPE bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.

Bits 19, 5:3 **ALGOMODE[3:0]:** Algorithm mode

Below definition includes the bit 19:
0000: TDES-ECB (triple-DES Electronic Codebook).
0001: TDES-CBC (triple-DES Cipher Block Chaining).
0010: DES-ECB (simple DES Electronic Codebook).
0011: DES-CBC (simple DES Cipher Block Chaining).
0100: AES-ECB (AES Electronic Codebook).
0101: AES-CBC (AES Cipher Block Chaining).
0110: AES-CTR (AES Counter Mode).
0111: AES key preparation for ECB or CBC decryption.
1000: AES-GCM (Galois Counter Mode) and AES-GMAC (Galois Message Authentication Code mode).
1001: AES-CCM (Counter with CBC-MAC).

Writing ALGOMODE bits while BUSY=1 has no effect. These bits can only be configured when BUSY=0.

Bit 2 **ALGODIR:** Algorithm Direction

0: Encrypt
1: Decrypt

Writing ALGODIR bit while BUSY=1 has no effect. It can only be configured when BUSY=0.

Bits 1:0 Reserved, must be kept at reset value.

34.6.2 CRYP status register (CRYP_SR)

Address offset: 0x04

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **BUSY:** Busy bit

0: The CRYP core is not processing any data. The reason is:

– either that the CRYP core is disabled (CRYPEN=0 in the CRYP_CR register) and the last processing has completed,

– or the CRYP core is waiting for enough data in the input FIFO or enough free space in the output FIFO (that is in each case at least 2 words in the DES, 4 words in the AES).

1: The CRYP core is currently processing a block of data or a key preparation is ongoing (AES ECB or CBC decryption only).

Bit 3 **OFFU:** Output FIFO full flag

0: Output FIFO is not full

1: Output FIFO is full

Bit 2 **OFNE:** Output FIFO not empty flag

0: Output FIFO is empty

1: Output FIFO is not empty

Bit 1 **IFNF:** Input FIFO not full flag

0: Input FIFO is full

1: Input FIFO is not full

Bit 0 **IFEM:** Input FIFO empty flag

0: Input FIFO is not empty

1: Input FIFO is empty

34.6.3 CRYP data input register (CRYP_DIN)

Address offset: 0x08

Reset value: 0x0000 0000

The CRYP_DIN register is the data input register. It is 32-bit wide. It is used to enter into the input FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to *Section 34.3.16: CRYP data registers and data swapping* for more details.

When CRYP_DIN register is written to the data are pushed into the input FIFO.

- If CRYPEN = 1, when at least two 32-bit words in the DES/TDES mode have been pushed into the input FIFO (four words in the AES mode), and when at least two words are free in the output FIFO (four words in the AES mode), the CRYP engine starts an encrypting or decrypting process.

When CRYP_DIN register is read:

- If CRYPEN = 0, the FIFO is popped, and then the data present in the Input FIFO are returned, from the oldest one (first reading) to the newest one (last reading). The IFEM flag must be checked before each read operation to make sure that the FIFO is not empty.
- if CRYPEN = 1, an undefined value is returned.

Note: After the CRYP_DIN register has been read once or several times, the FIFO must be flushed by setting the FFLUSH bit prior to processing new data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATAIN[31:0]:** Data Input

On read FIFO is popped (last written value is returned), and its value is returned if CRYPEN=0. If CRYPEN=1 DATAIN register returns an undefined value.
On write current register content is pushed inside the FIFO.

34.6.4 CRYP data output register (CRYP_DOUT)

Address offset: 0x0C

Reset value: 0x0000 0000

The CRYP_DOUT register is the data output register. It is read-only and 32-bit wide. It is used to retrieve from the output FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to *Section 34.3.16: CRYP data registers and data swapping* for more details.

When CRYP_DOUT register is read, the last data entered into the output FIFO (pointed to by the read pointer) is returned.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DATAOUT[31:0]:** Data Output

On read returns output FIFO content (pointed to by read pointer), else returns an undefined value.

On write, no effect.

34.6.5 CRYP DMA control register (CRYP_DMACR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DOEN:** DMA Output Enable

When this bit is set, DMA requests are automatically generated by the peripheral during the output data phase.

0: DMA for outgoing data transfer is disabled

1: DMA for outgoing data transfer is enabled

Bit 0 **DIEN:** DMA Input Enable

When this bit is set, DMA requests are automatically generated by the peripheral during the input data phase.

0: DMA for incoming data transfer is disabled

1: DMA for incoming data transfer is enabled

34.6.6 CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14

Reset value: 0x0000 0000

The CRYP_IMSCR register is the interrupt mask set or clear register. It is a read/write register. When a read operation is performed, this register gives the current value of the mask applied to the relevant interrupt. Writing 1 to the particular bit sets the mask, thus enabling the interrupt to be read. Writing 0 to this bit clears the corresponding mask. All the bits are cleared to 0 when the peripheral is reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTIM:** Output FIFO service interrupt mask
0: Output FIFO service interrupt is masked
1: Output FIFO service interrupt is not masked

Bit 0 **INIM:** Input FIFO service interrupt mask
0: Input FIFO service interrupt is masked
1: Input FIFO service interrupt is not masked

34.6.7 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18

Reset value: 0x0000 0001

The CRYP_RISR register is the raw interrupt status register. It is a read-only register. When a read operation is performed, this register gives the current raw status of the corresponding interrupt, i.e. the interrupt information without taking CRYP_IMSCR mask into account. Write operations have no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTRIS:** Output FIFO service raw interrupt status
This bit gives the output FIFO interrupt information without taking CRYP_IMSCR corresponding mask into account.
0: Raw interrupt not pending
1: Raw interrupt pending

Bit 0 **INRIS:** Input FIFO service raw interrupt status
This bit gives the input FIFO interrupt information without taking CRYP_IMSCR corresponding mask into account.
0: Raw interrupt not pending
1: Raw interrupt pending

34.6.8 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C

Reset value: 0x0000 0000

The CRYP_MISR register is the masked interrupt status register. It is a read-only register. When a read operation is performed, this register gives the current masked status of the corresponding interrupt, i.e. the interrupt information taking CRYP_IMSCR mask into account. Write operations have no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTMIS:** Output FIFO service masked interrupt status

This bit gives the output FIFO interrupt information without taking into account the corresponding CRYP_IMSCR mask.

0: Interrupt not pending

1: Interrupt pending

Bit 0 **INMIS:** Input FIFO service masked interrupt status

This bit gives the input FIFO interrupt information without taking into account the corresponding CRYP_IMSCR mask.

0: Interrupt not pending

1: Interrupt pending when CRYPEN= 1

34.6.9 CRYP key register 0L (CRYP_K0LR)

Address offset: 0x20

Reset value: 0x0000 0000

CRYP key registers contain the cryptographic keys.

- In DES/TDES mode, the keys are 64-bit binary values (number from left to right, that is the leftmost bit is bit 1) and named K1, K2 and K3 (K0 is not used). Each key consists of 56 information bits and 8 parity bits.
- In AES mode, the key is considered as a single 128, 192 or 256 bits long sequence $K_0K_1K_2...K_{127/191/255}$. The AES key is entered into the registers as follows:
  - for AES-128: $K_0..K_{127}$ corresponds to $b_{127}..b_0$ ($b_{255}..b_{128}$ are not used),
  - for AES-192: $K_0..K_{191}$ corresponds to $b_{191}..b_0$ ($b_{255}..b_{192}$ are not used),
  - for AES-256: $K_0..K_{255}$ corresponds to $b_{255}..b_0$.

In all cases key bit K_0 is the leftmost bit in CRYP inner memory and register bit b_0 is the rightmost bit in corresponding CRYP_KxLR key register.

For more information refer to *Section 34.3.17: CRYP key registers*.

Note: Write accesses to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K255	K254	K253	K252	K251	K250	K249	K248	K247	K246	K245	K244	K243	K242	K241	K240
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K239	K238	K237	K236	K235	K234	K233	K232	K231	K230	K229	K228	K227	K226	K225	K224
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[255:224]**: AES key bit x (x= 224 to 255)
Note: This register is not used in DES mode

34.6.10 CRYP key register 0R (CRYP_K0RR)

Address offset: 0x24

Reset value: 0x0000 0000

Refer to *Section 34.6.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K223	K222	K221	K220	K219	K218	K217	K216	K215	K214	K213	K212	K211	K210	K209	K208
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K207	K206	K205	K204	K203	K202	K201	K200	K199	K198	K197	K196	K195	K194	K193	K192
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[223:192]**: AES key bit x (x= 192 to 223)
Note: This register is not used in DES mode

34.6.11 CRYP key register 1L (CRYP_K1LR)

Address offset: 0x28

Reset value: 0x0000 0000

Refer to *Section 34.6.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K191	K190	K189	K188	K187	K186	K185	K184	K183	K182	K181	K180	K179	K178	K177	K176
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K175	K174	K173	K172	K171	K170	K169	K168	K167	K166	K165	K164	K163	K162	K161	K160
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[191:160]**: AES key bit x (x= 160 to 191)
In DES mode, K192 corresponds to key K1 bit 1 and K160 corresponds to key K1 bit 32.

34.6.12 CRYP key register 1R (CRYP_K1RR)

Address offset: 0x2C

Reset value: 0x0000 0000

Refer to *Section 34.6.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K159	K158	K157	K156	K155	K154	K153	K152	K151	K150	K149	K148	K147	K146	K145	K144
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K143	K142	K141	K140	K139	K138	K137	K136	K135	K134	K133	K132	K131	K130	K129	K128
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[159:128]**: AES key bit x (x= 128 to 159)
In DES mode K159 corresponds to key K1 bit 33 and K128 corresponds to key K1 bit 64.

34.6.13 CRYP key register 2L (CRYP_K2LR)

Address offset: 0x30

Reset value: 0x0000 0000

Refer to *Section 34.6.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K127	K126	K125	K124	K123	K122	K121	K120	K119	K118	K117	K116	K115	K114	K113	K112
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K111	K110	K109	K108	K107	K106	K105	K104	K103	K102	K101	K100	K99	K98	K97	K96
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[127:96]**: AES key bit x (x= 96 to 127)
In DES mode K127 corresponds to key K2 bit 1 and K96 corresponds to key K2 bit 32.

34.6.14 CRYP key register 2R (CRYP_K2RR)

Address offset: 0x34

Reset value: 0x0000 0000

Refer to *Section 34.6.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K95	K94	K93	K92	K91	K90	K89	K88	K87	K86	K85	K84	K83	K82	K81	K80
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K79	K78	K77	K76	K75	K74	K73	K72	K71	K70	K69	K68	K67	K66	K65	K64
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[95:64]**: AES key bit x (x= 64 to 95)
In DES mode K95 corresponds to key K2 bit 33 and K64 corresponds to key K2 bit 64.

34.6.15 CRYP key register 3L (CRYP_K3LR)

Address offset: 0x38

Reset value: 0x0000 0000

Refer to *Section 34.6.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K63	K62	K61	K60	K59	K58	K57	K56	K55	K54	K53	K52	K51	K50	K49	K48
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K47	K46	K45	K44	K43	K42	K41	K40	K39	K38	K37	K36	K35	K34	K33	K32
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[63:32]**: AES key bit x (x= 32 to 63)
In DES mode K63 corresponds to key K3 bit 1 and K32 corresponds to key K3 bit 32.

34.6.16 CRYP key register 3R (CRYP_K3RR)

Address offset: 0x3C

Reset value: 0x0000 0000

Refer to *Section 34.6.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K31	K30	K29	K28	K27	K26	K25	K24	K23	K22	K21	K20	K19	K18	K17	K16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K15	K14	K13	K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1	K0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[31:0]**: AES key bit x (x= 0 to 31)
In DES mode K31 corresponds to key K3 bit 33 and K0 corresponds to key K3 bit 64.

34.6.17 CRYP initialization vector register 0L (CRYP_IV0LR)

Address offset: 0x40

Reset value: 0x0000 0000

The CRYP_IV0...1(L/R)R are the left-word and right-word registers for the initialization vector (64 bits for DES/TDES and 128 bits for AES). For more information refer to *Section 34.3.18: CRYP initialization vector registers*.

IV0 is the leftmost bit whereas IV63 (DES, TDES) or IV127 (AES) are the rightmost bits of the initialization vector. IV1(L/R)R is used only in the AES. Only CRYP_IV0(L/R) is used in DES/TDES.

Note: Write access to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV0	IV1	IV2	IV3	IV4	IV5	IV6	IV7	IV8	IV9	IV10	IV11	IV12	IV13	IV14	IV15
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV16	IV17	IV18	IV19	IV20	IV21	IV22	IV23	IV24	IV25	IV26	IV27	IV28	IV29	IV30	IV31
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IV[0:31]**: Initialization vector bit x (x= 0 to 31)

34.6.18 CRYP initialization vector register 0R (CRYP_IV0RR)

Address offset: 0x44

Reset value: 0x0000 0000

Refer to *Section 34.6.17: CRYP initialization vector register 0L (CRYP_IV0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV32	IV33	IV34	IV35	IV36	IV37	IV38	IV39	IV40	IV41	IV42	IV43	IV44	IV45	IV46	IV47
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV48	IV49	IV50	IV51	IV52	IV53	IV54	IV55	IV56	IV57	IV58	IV59	IV60	IV61	IV62	IV63
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IV[32:63]**: Initialization vector bit x (x= 32 to 63)

34.6.19 CRYP initialization vector register 1L (CRYP_IV1LR)

Address offset: 0x48

Reset value: 0x0000 0000

Refer to *Section 34.6.17: CRYP initialization vector register 0L (CRYP_IV0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV64	IV65	IV66	IV67	IV68	IV69	IV70	IV71	IV72	IV73	IV74	IV75	IV76	IV77	IV78	IV79
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV80	IV81	IV82	IV83	IV84	IV85	IV86	IV87	IV88	IV89	IV90	IV91	IV92	IV93	IV94	IV95
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IV[64:95]**: Initialization vector bit x (x= 64 to 95)
Note: This register is not used in DES mode

34.6.20 CRYP initialization vector register 1R (CRYP_IV1RR)

Address offset: 0x4C

Reset value: 0x0000 0000

Refer to *Section 34.6.17: CRYP initialization vector register 0L (CRYP_IV0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV96	IV97	IV98	IV99	IV100	IV101	IV102	IV103	IV104	IV105	IV106	IV107	IV108	IV109	IV110	IV111
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV112	IV113	IV114	IV115	IV116	IV117	IV118	IV119	IV120	IV121	IV122	IV123	IV124	IV125	IV126	IV127
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IV[96:127]**: Initialization vector bit x (x= 96 to 127)
Note: This register is not used in DES mode

34.6.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)

Address offset: 0x050 + x* 0x4 (x=0 to 7)

Reset value: 0x0000 0000

These registers contain the complete internal register states of the CRYP processor when the GCM/GMAC or CCM algorithm is selected. They are useful when a context swap has to be performed because a high-priority task needs the cryptographic processor while it is already in use by another task.

When such an event occurs, the CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R (in GCM/GMAC mode) or CRYP_CSGCMCCM0..7R (in CCM mode) registers have to be read and the values retrieved have to be saved in the system memory space. The cryptographic processor can then be used by the preemptive task. Then when the cryptographic computation is complete, the saved context can be read from memory and written back into the corresponding context swap registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRYP_CSGCMCCMxR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYP_CSGCMCCMxR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRYP_CSGCMCCMxR[31:0]**: CRYP internal state registers for GCM, GMAC and CCM modes.

Note: This register is not used in DES/TDES or other AES modes than the ones indicated

34.6.22 CRYP context swap GCM registers (CRYP_CSGCMxR)

Address offset: 0x070 + x* 0x4 (x=0 to 7)

Reset value: 0x0000 0000

Please refer to *Section 34.6.21: CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRYP_CSGCMxR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYP_CSGCMxR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRYP_CSGCMxR[31:0]**: CRYP internal state registers for GCM and GMAC modes.

Note: This register is not used in DES/TDES or other AES modes than the ones indicated

34.6.23 CRYP register map

Table 286. CRYP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 0x00	CRYP_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				ALGOMODE[3]	Res.	GCM_CCMPH		CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Res.	Res.
	Reset value									0	0	0	0	0		0	0	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
	Reset value																															0	1
0x1C	CRYP_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
	Reset value																															0	0
0x20	CRYP_K0LR	CRYP_K0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	CRYP_K0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
... ...																																	
0x38	CRYP_K3LR	CRYP_K3LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	CRYP_K3RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	CRYP_IV0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 286. CRYP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44	CRYP_IV0RR	CRYP_IV0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	CRYP_IV1LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	CRYP_IV1RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	CRYP_CSGCMCCM0R	CRYP_CSGCMCCM0R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	CRYP_CSGCMCCM1R	CRYP_CSGCMCCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	CRYP_CSGCMCCM2R	CRYP_CSGCMCCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	CRYP_CSGCMCCM3R	CRYP_CSGCMCCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	CRYP_CSGCMCCM4R	CRYP_CSGCMCCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	CRYP_CSGCMCCM5R	CRYP_CSGCMCCM5R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	CRYP_CSGCMCCM6R	CRYP_CSGCMCCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	CRYP_CSGCMCCM7R	CRYP_CSGCMCCM7R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	CRYP_CSGCM0R	CRYP_CSGCM0R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	CRYP_CSGCM1R	CRYP_CSGCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78	CRYP_CSGCM2R	CRYP_CSGCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	CRYP_CSGCM3R	CRYP_CSGCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	CRYP_CSGCM4R	CRYP_CSGCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	CRYP_CSGCM5R	CRYP_CSGCM5R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 286. CRYP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x88	CRYP_CSGCM6R	CRYP_CSGCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	CRYP_CSGCM7R	CRYP_CSGCM7R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

35 Hash processor (HASH)

35.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-224, SHA-256), the MD5 (message-digest algorithm 5) hash algorithm and the HMAC (keyed-hash message authentication code) algorithm suitable for a variety of applications. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of up to (261 – 1) bits. It also computes 128 bits digests for the MD5 algorithm.

35.2 HASH main features

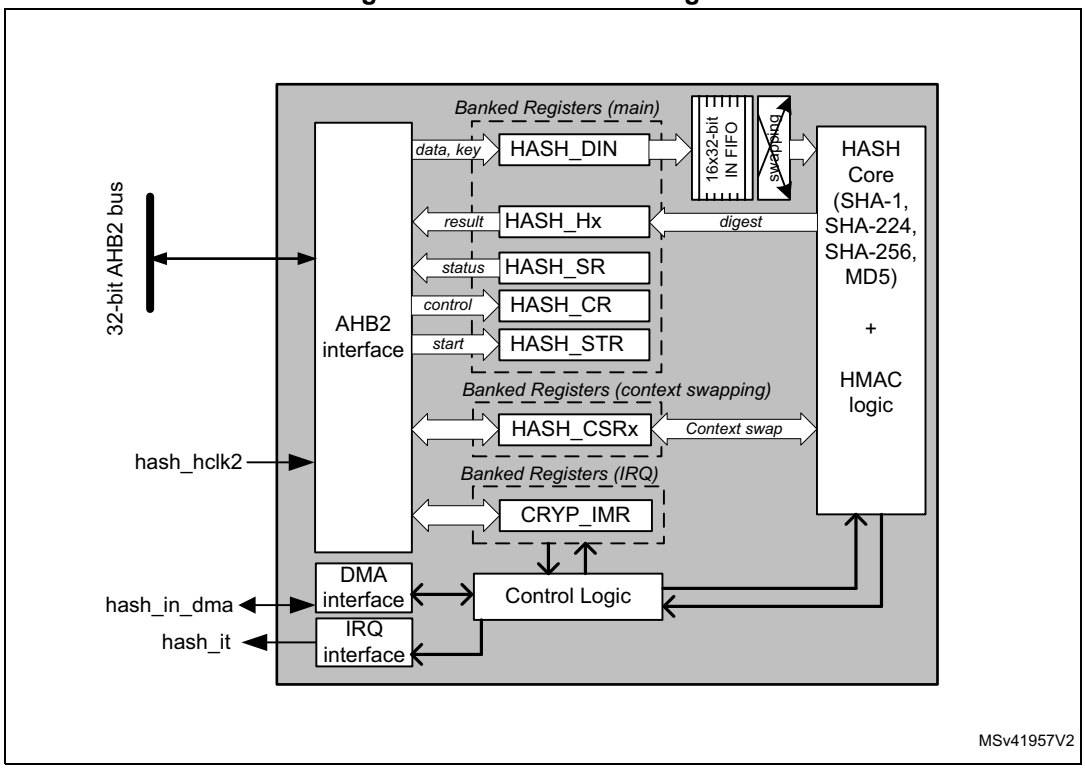
- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Internet Engineering Task Force (IETF) Request For Comments RFC 1321 *MD5 Message-Digest Algorithm*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104 *HMAC: Keyed-Hashing for Message Authentic*ation, and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Word swapping supported: bits, bytes, half-words and 32-bit words
- Automatic padding to complete the input bit string to fit digest minimum block size of 512 bits (16 × 32 bits)
- Single 32-bit input register associated to an internal input FIFO of sixteen 32-bit words, corresponding to one block size
- Fast computation of SHA-1, SHA-224, SHA-256, and MD5
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 66 clock cycles for processing one 512-bit block of data using MD5 algorithm
- AHB slave peripheral, accessible through 32-bit word accesses only (else an AHB error is generated)
- 8 × 32-bit words (H0 to H7) for output message digest
- Automatic data flow control with support of direct memory access (DMA) using one channel. Fixed burst of 4 supported.
- Interruptible message digest computation, on a per-32-bit word basis
  - Re-loadable digest registers
  - Hashing computation suspend/resume mechanism, including using DMA

35.3 HASH functional description

35.3.1 HASH block diagram

Figure 276 shows the block diagram of the hash processor.

Figure 276. HASH block diagram



35.3.2 HASH internal signals

Table 287 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 287. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk2	digital input	AHB2 bus clock
hash_it	digital output	Hash processor global interrupt request
hash_in_dma	digital input/output	DMA burst request/ acknowledge

35.3.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard and the IETF RFC1321 publication (MD5).

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message of any length below 2^{64} bits is provided on input, the SHA-1, SHA-224, SHA-256 and MD5 processing core produces respectively a 160-bit, 224 bit, 256 bit and 128-bit output string called a message digest. The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” because it is computationally infeasible to find a message that corresponds to a given message digest (SHA-1 is no more qualified as secure since February 2017), or to find two different messages that produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

35.3.4 Message data feeding

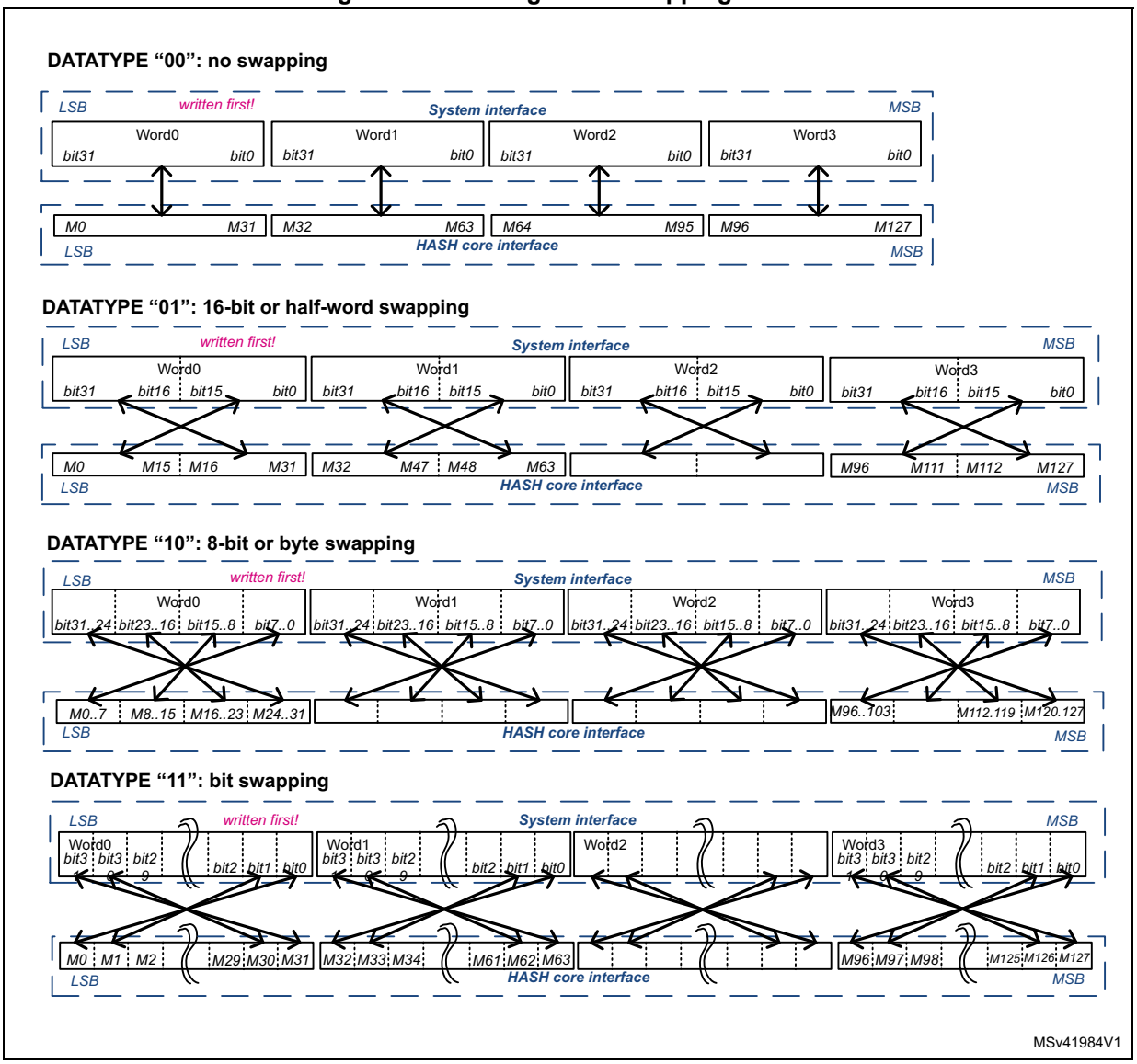
The message (or data file) to be processed by the HASH should be considered as a bit string. Per FIPS PUB 180-1 and 180-2 standards this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “`01100001 01100010 01100011`” is represented by a 32-bit word `0x00636261`, and 8-bit words `0x61626300`.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO (IN FIFO) each time the register is written with new data. Hence HASH_DIN and the input FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (e.g. byte swapping when data are ASCII text stream) there must be a bit, byte, half-word or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 277* shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into IN FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR).

HASH_DIN data endianness when bit swapping is disabled (DATATYPE=”00”) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 277. Message data swapping feature



35.3.5 Message digest computing

The hash processor sequentially processes 512-bit blocks when computing the message digest. Thus, each time 16 × 32-bit words (= 512 bits) have been written to the hash processor by the DMA or the CPU, the HASH automatically starts computing the message digest. This operation is known as ‘partial digest computation’.

As described in *Section 35.3.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO. In order to perform the hash computation on this data below sequence shall be used by the application.

1. Initialize the hash processor using the HASH_CR register:
   – Select the right algorithm using ALGO field. If needed program the correct swapping operation on the message input words using DATATYPE bitfield in HASH_CR.
   – Set MODE=1 and select the key length using LKEY if HMAC mode has been selected.
   – Update NBLW to define the number of valid bits in last word if it is different from 32 bits. If it is the case automatic padding could be applied by the HASH.
2. Complete the initialization by setting to 1 the INIT bit in HASH_CR. Also set the bit DMAE to 1 if data are transferred via DMA.

Caution: When programming step 2, it is important to set up before or at the same time the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW).

3. Start filling data by writing to HASH_DIN register, unless data are automatically: transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the IN FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   a) When data are filled by software:
   – The partial digest computation is triggered when the software writes an additional word to the HASH_DIN register (actually the first word of the next block). Once the processor is ready again (DINIS=1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.
   – The final digest computation is triggered when the last block is entered and the software writes the DCAL bit to 1. If the message length is not an exact multiple of 512 bits, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 35.3.6* for details).
   b) When data are filled by DMA as a single DMA transfer (MDMAT bit=”0”):
   – The partial digest computation is triggered automatically each time the FIFO is full.
   – The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register (DCAL bit is set to 1 by hardware). If the message length is not an exact multiple of 512 bits, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 35.3.6* for details).
   c) When data are filled using multiple DMA transfers (MDMAT bit=”1”) :
   – The partial digest computations are triggered as for single DMA transfers. However the final digest computation is not triggered automatically when the last block has been transferred to the HASH_DIN register (DCAL bit is not set to 1 by hardware). It allows the hash processor to receive a new DMA transfer as part of

this digest computation. To launch the final digest computation, the software must set MDMAT bit to 0 before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers (see description before).

4. Once computed, the digest can be read from the output registers as described in *Table 288*.

Table 288. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
MD5	HASH_H0 to HASH_H3	HASH_H0[31]	128
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA-256	HASH_H0 to HASH_H7	HASH_H0[31]	256

For more information about HMAC detailed instructions, refer to *Section 35.3.7: HMAC operation*.

35.3.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every 512-bit block) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed padding sequences with DMA is enabled or disabled are described in *Section 35.3.5: Message digest computing*.

Padding example

As specified by Federal Information Processing Standards PUB 180-1 and PUB 180-2, message padding consists in appending a “1” followed by *k* “0”s, itself followed by a 64-bit integer that is equal to the length *L* in bits of the message. These three padding operations generate a padded message of length $L + 1 + k + 64$, which by construction is a multiple of 512 bits.

For the hash processor, the “1” is added to the last word written to the HASH_DIN register at the bit position defined by the NBLW bitfield, and the remaining upper bits are cleared (“0”s).

Example from FIPS PUB180-2

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24:

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, L, in two-word format (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE=10 in HASH_CR), the above message has to be entered by following below the sequence:

1. `0xUU636261` is written to the HASH_DIN register (where ‘U’ means don’t care).
2. `0x18` is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits).
3. `0x10` is written to the HASH_STR register to start the message padding (described above) and then perform the digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0...4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_H0 = 0xA9993E36
HASH_H1 = 0x4706816A
HASH_H2 = 0xBA3E2571
HASH_H3 = 0x7850C26C
HASH_H4 = 0x9CD0D89D
```

35.3.7 HMAC operation

Overview

As specified by Internet Engineering Task Force *RFC2104, HMAC: keyed-hashing for message authentication,* the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message) = Hash((key | pad) XOR [0x5C]n
          | Hash((key | pad) XOR [0x36]n | message))
```

where:

- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the size of the underlying hash function data block that is 512 bits for SHA-1, SHA224, SHA-256, MD5 hash algorithms (i.e. n=64).
- `pad` is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n,* the application shall first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

HMAC processing

Four different steps are required to compute the HMAC:

1. The block is initialized by writing the INIT bit to 1 with the MODE bit at 1 and the ALGO bits set to the value corresponding to the desired algorithm. The LKEY bit must also be set to 1 if the key being used is longer than 64 bytes. In this case, as required by HMAC specifications, the hash processor will use the hash of the key instead of the real key.
2. The key to be used for the inner hash function must be provided to the hash processor:

   The key loading operation follows the same mechanism as the message bit string loading, i.e. write key data into HASH_DIN and complete the transfer by writing to HASH_STR register.

Note: Endianness details can be found in Section 35.3.4: Message data feeding.

3. Once the last key word has been entered and computation has started, the hash processor elaborates the inner key material. Once this operation has completed, it is ready to accept the message bit string as described in *Section 35.3.4: Message data feeding*.
4. After the final hash round, the hash processor returns “ready” to indicate that it is ready to receive the key to be used for the outer hash function (normally, this key is the same as the one used for the inner hash function). When the last word of the key is entered and computation starts, the HMAC result can be found in the HASH_H0...HASH_H7 registers.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 35.5: HASH processing time.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO=”00” and MODE=”1” in HASH_CR) as specified by NIST.

Let us assume that the original message is the ASCII binary-coded form of “`Sample message for keylen=blocklen`”, of length L = 34 bytes. If the HASH is programmed in

no swapping mode (DATATYPE=00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (lenght=64, i.e. no padding), specified by NIST. As key lenght=64, LKEY bit is set to 0 in HASH_CR register

   ```
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F
   ```

2. **Message** input (lenght=34, i.e. padding required). HASH_STR must be set to `0x20` to start message padding and inner hash computation (see 'U' as don't care)

   ```
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU
   ```

3. **Outer hash key** input (lenght=64, i.e. no padding). A key identical to the inner hash key is entered here.

4. **Final outer hash computing** is then performed by the HASH. The HMAC SHA-1 digest result is available in the HASH_HRx registers (x = 0...4), as shown below:

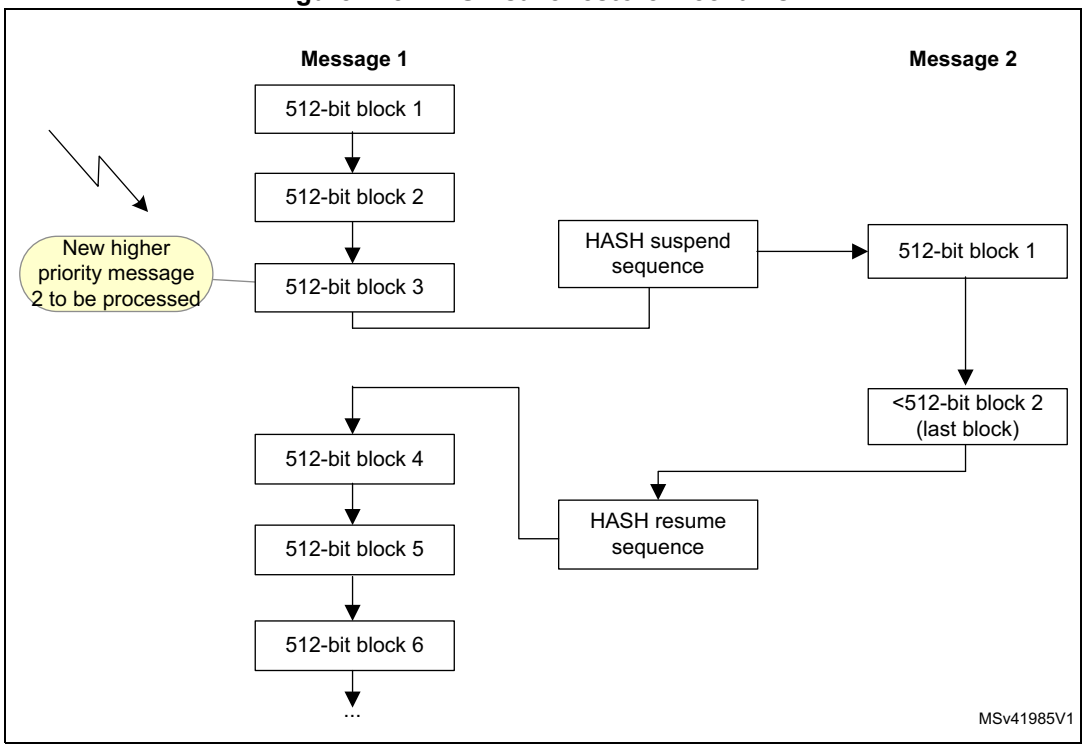
   ```
   HASH_H0 = 0x5FD596EE
   HASH_H1 = 0x78D5553C
   HASH_H2 = 0x8FF4E72D
   HASH_H3 = 0x266DFD19
   HASH_H4 = 0x2366DA29
   ```

35.3.8 Context swapping

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 278*.

Figure 278. HASH save/restore mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described below.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing. This means that the user application must wait until DINIS = 1 (last block processed and input FIFO empty) or NBW ≠ 0 (FIFO not full and no processing ongoing). The detailed procedure is described below.

- **Current context saving**

  Before interrupting the current message digest calculation, the application must store the contents of the following registers into memory:

  – HASH_IMR
  – HASH_STR
  – HASH_CR
  – HASH_CSR0 to HASH_CSR53

- **Current context restoring**

  To resume processing the interrupted message, the application must respect the following steps:

  a) Write the following registers with the values saved in memory: HASH_IMR, HASH_STR and HASH_CR.
  b) Initialize the hash processor by setting the INIT bit in the HASH_CR register.
  c) Write the HASH_CSR0 to HASH_CSR53 registers with the values saved in memory.
  d) Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is not possible to predict if a DMA transfer is ongoing. The user application must thus stop DMA transfers, then wait until the hash processor is ready before interrupting the current message digest calculation. The detailed procedure is described below.

- **Current context saving**

  Before interrupting the current message digest calculation using DMA, the application must respect the following steps:

  a) Clear the DMAE bit to disable the DMA interface.
  b) Wait until the current DMA transfer is complete (wait for DMAS = 0 in the HASH_SR register). Note that the block may or may not have been totally transferred to the HASH.
  c) Disable the corresponding channel in the DMA controller.
  d) Wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1

- **Current context restoring**

  To resume processing the interrupted message using DMA, the application must respect the following steps:

  a) Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
  b) Restart the processing from the point where it was interrupted by setting the DMAE bit.

Note: If the context swapping does not involve HMAC operations, the HASH_CSR38 to HASH_CSR53 registers do not need to be saved and restored.

If the context swapping occurs between two blocks (the last block was completely processed and the next block has not yet been pushed into the IN FIFO, NBW = 000 in the HASH_CR register), the HASH_CSR22 to HASH_CSR37 registers do not need to be saved and restored.

35.3.9 HASH DMA interface

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH asserts the burst request signal to the DMA controller when there is enough free words in the FIFO to support a burst of four words.

Once four 32-bit words have been received, the HASH automatically restarts this process, checks the FIFO size, and asserts a new request if the FIFO status allow a burst reception. For more information refer to *Section 35.3.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that will be copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

```
NBLW = Len(Message)% 32
```

where "`x%32`" gives the remainder of x divided by 32.

DMAS bit in HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared to 0 and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

35.3.10 HASH error management

No error flags are generated by the HASH hardware.

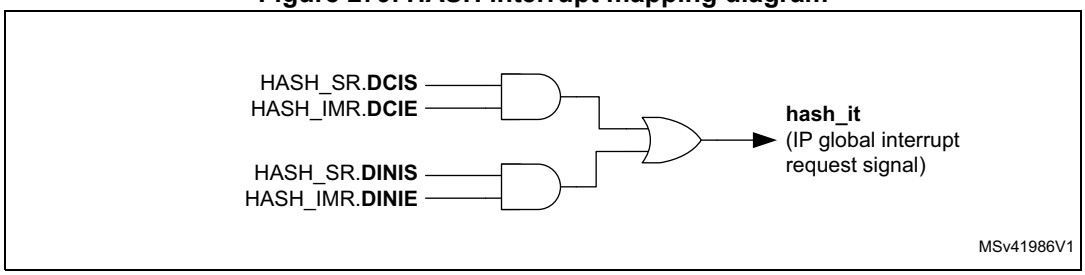
35.4 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal, as shown on *Figure 279*.

Figure 279. HASH interrupt mapping diagram



The above interrupt sources can be enabled or disabled individually by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual interrupt events can be read from the HASH_SR register. *Table 289* gives a summary of the available features.

Table 289. HASH interrupt requests

Interrupt event	Event flag	Enable control bit
Digest computation completed flag	DCIS	DCIE
Data input buffer ready to get a new block flag	DINIS	DINIE

35.5 HASH processing time

Table 290 summarizes the time required to process a 512-bit intermediate block for each mode of operation.

Table 290. Processing time (in clock cycle)

Mode of operation	FIFO load(1)	Computation phase	Total
MD5	16	50	**66**
SHA-1	16	66	**82**
SHA-224	16	50	**66**
SHA-256			

1. The time required to load the 16 words of the block into the processor must be added to this value.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

35.6 HASH registers

The HASH core is associated with several control and status registers and five message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB2 error is generated.

35.6.1 HASH control register (HASH_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[1]	Res.	LKEY
													rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				ALGO[0]	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r	rw	rw	rw	rw	rw	w		

Bits 31:19 Reserved, must be kept at reset value.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LKEY:** Long key selection

This bit selects between short key (≤ 64 bytes) or long key (> 64 bytes) in HMAC mode.

0: Short key (≤ 64 bytes)

1: Long key (> 64 bytes)

Note: This selection is only taken into account when the INIT bit is set and MODE= 1. Changing this bit during a computation has no effect.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT:** Multiple DMA Transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE:** DIN not empty

This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is written to 1.

0: No data are present in the data input buffer

1: The input buffer contains at least one word of data

Bits 11:8 **NBW[3:0]:** Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO. NBW increments (+1) when a write access is performed to the HASH_DIN register while DINNE = 1.

It goes to zero when the INIT bit is written to 1 or when a digest calculation starts (DCAL written to 1 or DMA end of transfer).

If the DMA is not used

0000 and DINNE=0: no word has been pushed into the DIN buffer, i.e. both HASH_DIN register and IN FIFO are empty.

0000 and DINNE=1: one word has been pushed into the DIN buffer, i.e. HASH_DIN register contains one word and IN FIFO is empty.

0001: two words have been pushed into the DIN buffer, i.e. HASH_DIN register and the IN FIFO contain one word each.

...

1111: 16 words have been pushed into the DIN buffer.

If the DMA is used

NBW is the exact number of words that have been pushed into the IN FIFO by the DMA.

Bits 18, 7 **ALGO[1:0]:** Algorithm selection

These bits selects the SHA-1, SHA-224, SHA-256 or the MD5 algorithm:

00: SHA-1 algorithm selected

01: MD5 algorithm selected

10: SHA-224 algorithm selected

11: SHA-256 algorithm selected

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bit 6 **MODE:** Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

0: Hash mode selected

1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

Note: This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

These bits define the format of the data entered into the HASH_DIN register:

00: 32-bit data. The data written to HASH_DIN are directly used by the hash processing, without reordering.

01: 16-bit data or half-word. The data written to HASH_DIN are considered as two half-words, and are swapped before being used by the hash processing.

10: 8-bit data or bytes. The data written to HASH_DIN are considered as four bytes, and are swapped before being used by the hash processing.

11: bit data or bit-string. The data written to HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the hash processing (first bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

0: DMA transfers disabled

1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

After this bit is set it is cleared by hardware while the last data of the message is written to the hash processor.

Setting this bit to 0 while a DMA transfer is on-going is not aborting this current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is written to 1.

Setting INIT bit to 1 does not clear DMAE bit.

Bit 2 **INIT:** Initialize message digest calculation

Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.

Writing this bit to 0 has no effect. Reading this bit always return 0.

Bits 1:0 Reserved, must be kept at reset value.

35.6.2 HASH data input register (HASH_DIN)

Address offset: 0x04

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks of 512 bits. When the HASH_DIN register is programmed, the value presented on the AHB databus is 'pushed' into the hash core and the register takes the new value presented on the AHB databus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a block of 16 words has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing new data into the HASH_DIN register (the first word of the next block) if the DMA is not used (intermediate digest calculation),
- automatically if the DMA is used.

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched:

- by writing the DCAL bit to 1 in the HASH_STR register (final digest calculation),
- automatically if the DMA is used and MDMAT bit is set to 0.

When a digest calculation (intermediate or final) is ongoing and a new write access to the HASH_DIN register is performed, wait-states are inserted on the AHB2 bus until the hash calculation completes.

When the HASH_DIN register is read, the last word written to this location is accessed (zero after reset).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATAIN[31:0]:** Data input

Reading this register returns the current register content.

Writing this register pushes the current register content into the IN FIFO, and the register takes the new value presented on the AHB databus.

35.6.3 HASH start register (HASH_STR)

Address offset: 0x08

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by writing the DCAL bit to 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							w				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

Writing this bit to 1 starts the message padding, using the previously written value of NBLW, and starts the calculation of the final message digest with all data words written to the IN FIFO since the INIT bit was last written to 1.

Reading this bit returns 0.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written in HASH_DIN register, the hash processor takes only the valid bits specified as below, after internal data swapping:

0x00: All 32 bits of the last data written are valid message bits i.e. M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid i.e. M[0]

0x02: Only two bits of the last data written (after swapping) are valid i.e. M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid i.e. M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid i.e. M[30:0]

The above mechanism is valid only if DCAL=0. If NBLW bits are written while DCAL is set to 1, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

35.6.4 HASH digest registers (HASH_HR0..7)

These registers contain the message digest result named as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result

  HASH_HR5 to HASH_HR7 registers are not used, and they are read as zero.
- HASH_HR0, HASH_HR1, HASH_HR2 and HASH_HR3 registers return A, B, C and D (respectively), as defined by MD5.

  HASH_HR4 to HASH_HR7 registers are not used, and they are read as zero.
- HASH_HR0 to HASH_HR6 registers return the SHA-224 digest result.

  HASH_HR7 register is not used, and it is read as zero.
- HASH_HR0 to HASH_HR7 registers return the SHA-256 digest result.

In all cases, the digest most significant bit is stored in HASH_HR0[31] and it is not used.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (that is when the DCAL bit has been written to 1), then the read operation is stalled until the hash calculation completes.

Note: HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 mapping are duplicated in two memory regions.

HASH_HR0

Address offset: 0x0C

Address offset: ALTERNATE: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H0															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H0															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H0**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

HASH_HR1

Address offset: 0x10

Address offset: ALTERNATE: 0x3104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H1															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H1															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H1**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

HASH_HR2

Address offset: 0x14

Address offset: ALTERNATE: 0x3108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H2															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H2**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

HASH_HR3

Address offset: 0x18

Address offset: ALTERNATE: 0x31C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H3															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H3**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

HASH_HR4

Address offset: 0x1C

Address offset: ALTERNATE: 0x320

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H4															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H4															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H4**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

HASH_HR5

Address offset: 0x324

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H5															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H5															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H5**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

HASH_HR6

Address offset: 0x328

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H6															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H6															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H6**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

HASH_HR7

Address offset: 0x32C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H7															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H7															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H7**: refer to *Section 35.6.4: HASH digest registers (HASH_HR0..7)* introduction

Note: When starting a digest computation for a new bit stream (by writing the INIT bit to 1), these registers are forced to their reset values.

35.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

35.6.6 HASH status register (HASH_SR)

Address offset: 0x24

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE=0 and no DMA transfer is ongoing. No interrupt is associated with this bit.
0: DMA interface is disabled (DMAE=0) and no transfer is ongoing
1: DMA interface is enabled (DMAE=1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by writing the INIT bit to 1 in the HASH_CR register.
0: No digest available in the HASH_HRx registers
1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the input buffer is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.
0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

35.6.7 HASH context swap registers (HASH_CSRx)

These registers contain the complete internal register states of the hash processor. They are useful when a context swap has to be done because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task, and when the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSR0

Address offset: 0x0F8

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CS0															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS0															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

HASH_CSRx (x=1 to 53)

Address offset: 0x0F8 + x * 0x4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSx															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSx															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

35.6.8 HASH register map

Table 291 gives the summary HASH register map and reset values.

Table 291. HASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[1]	Res.	LKEY	Res.	Res.	MDMAT	DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Res.	Res.
	Reset value														0		0			0	0	0	0	0	0	0	0	0	0	0	0		
0x04	HASH_DIN	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW				
	Reset value																								0				0	0	0	0	0
0x0C	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x24	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value																													0	0	0	1
0xF8	HASH_CSR0	CSR0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0xFC	HASH_CSR1	CSR1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x314	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x318	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x31C	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x320	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x324	HASH_HR5	H5																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x328	HASH_HR6	H6																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x32C	HASH_HR7	H7																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

36 High-Resolution Timer (HRTIM)

36.1 Introduction

The high-resolution timer can generate up to 10 digital signals with highly accurate timings. It is primarily intended to drive power conversion systems such as switch mode power supplies or lighting systems, but can be of general purpose usage, whenever a very fine timing resolution is expected.

Its modular architecture allows to generate either independent or coupled waveforms. The wave-shape is defined by self-contained timings (using counters and compare units) and a broad range of external events, such as analog or digital feedbacks and synchronization signals. This allows to produce a large variety of control signal (PWM, phase-shifted, constant Ton,...) and address most of conversion topologies.

For control and monitoring purposes, the timer has also timing measure capabilities and links to built-in ADC and DAC converters. Last, it features light-load management mode and is able to handle various fault schemes for safe shut-down purposes.

36.2 Main features

- Multiple timing units
  - Full-resolution available on all outputs, possibility to adjust duty-cycle, frequency and pulse width in triggered one-pulse mode
  - 6 16-bit timing units (each one with an independent counter and 4 compare units)
  - 10 outputs that can be controlled by any timing unit, up to 32 set/reset sources per channel
  - Modular architecture to address either multiple independent converters with 1 or 2 switches or few large multi-switch topologies
- Up to 10 external events, available for any timing unit
  - Programmable polarity and edge sensitivity
  - 5 events with a fast asynchronous mode
  - 5 events with a programmable digital filter
  - Spurious events filtering with blanking and windowing modes
- Multiple links to built-in analog peripherals
  - 4 triggers to ADC converters
  - 3 triggers to DAC converters
  - 3 comparators for analog signal conditioning
- Versatile protection scheme
  - 5 fault inputs can be combined and associated to any timing unit
  - Programmable polarity, edge sensitivity, and programmable digital filter
  - dedicated delayed protections for resonant converters
- Multiple HRTIM instances can be synchronized with external synchronization inputs/outputs
- Versatile output stage
  - Full-resolution Deadtime insertion
  - Programmable output polarity
  - Chopper mode
- Burst mode controller to handle light-load operation synchronously on multiple converters
- 7 interrupt vectors, each one with up to 14 sources
- 6 DMA requests with up to 14 sources, with a burst mode for multiple registers update

36.3 Functional description

36.3.1 General description

The HRTIM can be partitioned into several sub entities:

- The master timer
- The timing units (Timer A to Timer E)
- The output stage
- The burst mode controller
- An external event and fault signal conditioning logic that is shared by all timers
- The system interface

The master timer is based on a 16-bit up counter. It can set/reset any of the 10 outputs via 4 compare units and it provides synchronization signals to the 5 timer units. Its main purpose is to have the timer units controlled by a unique source. An interleaved buck converter is a typical application example where the master timer manages the phase-shifts between the multiple units.

The timer units are working either independently or coupled with the other timers including the master timer. Each timer contains the controls for two outputs. The outputs set/reset events are triggered either by the timing units compare registers or by events coming from the master timer, from the other timers or from external events.

The output stage has several duties

- Addition of deadtime when the 2 outputs are configured in complementary PWM mode
- Addition of a carrier frequency on top of the modulating signal
- Management of fault events, by asynchronously asserting the outputs to a predefined safe level

The burst mode controller can take over the control of one or multiple timers in case of light-load operation. The burst length and period can be programmed, as well as the idle state of the outputs.

The external event and fault signal conditioning logic includes:

- The input selection MUXes (for instance for selecting a digital input or an on-chip source for a given external event channel)
- Polarity and edge-sensitivity programming
- Digital filtering (for 5 channels out of 10)

The system interface allows the HRTIM to interact with the rest of the MCU:

- Interrupt requests to the CPU
- DMA controller for automatic accesses to/from the memories, including an HRTIM specific burst mode
- Triggers for the ADC and DAC converters

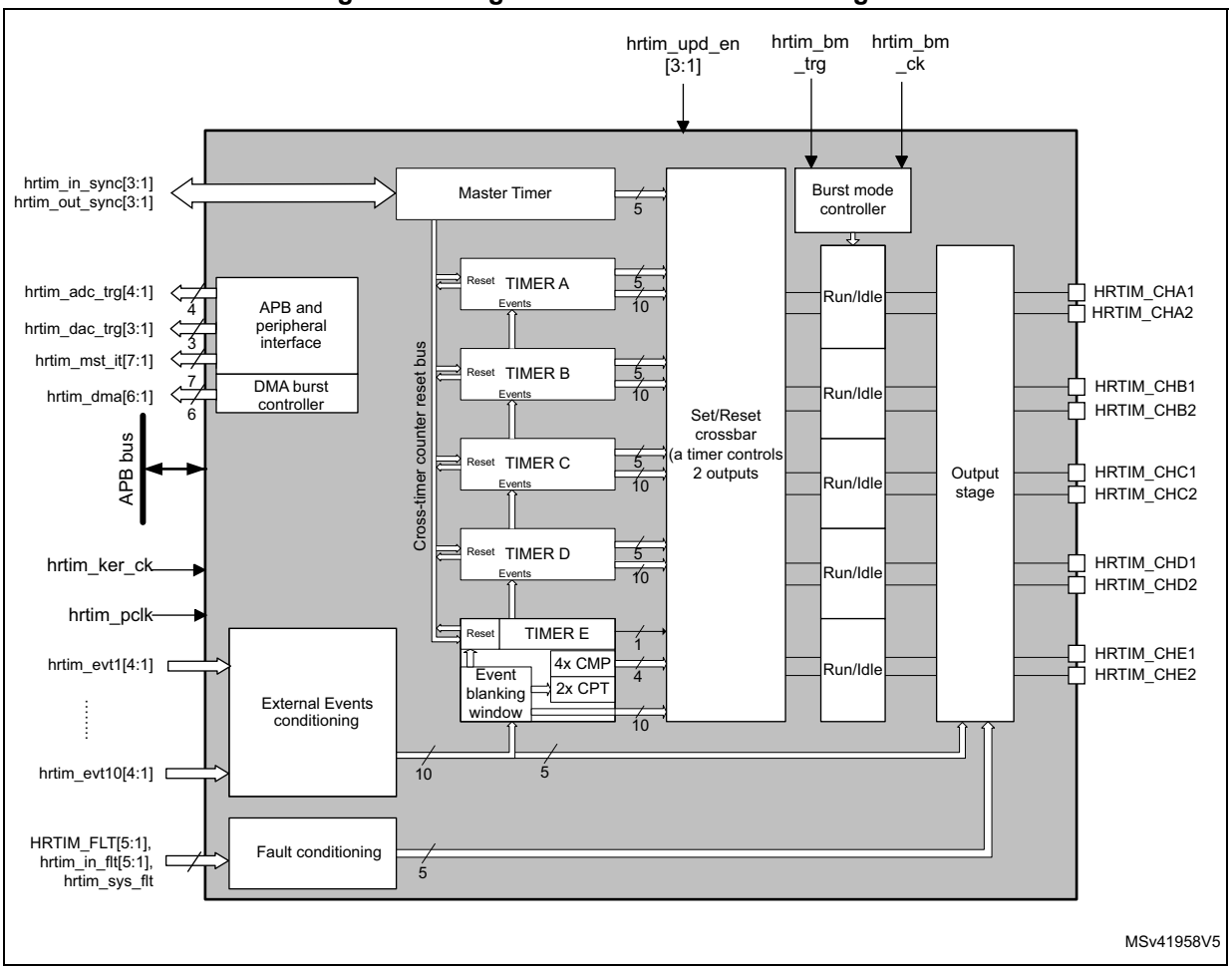
The HRTIM registers are split into 7 groups:

- Master timer registers
- Timer A to Timer E registers
- Common registers for features shared by all timer units

Note: As a writing convention, references to the 5 timing units in the text and in registers are generalized using the "x" letter, where x can be any value from A to E.

The block diagram of the timer is shown in *Figure 280*.

Figure 280. High-resolution timer block diagram



36.3.2 HRTIM pins and internal signals

The table here below summarizes the HRTIM inputs and outputs, both on-chip and off-chip.

Table 292. HRTIM Input/output summary

Signal name	Signal type	Description
HRTIM_CHA1, HRTIM_CHA2, HRTIM_CHB1, HRTIM_CHB2, HRTIM_CHC1, HRTIM_CHC2, HRTIM_CHD1, HRTIM_CHD2, HRTIM_CHE1, HRTIM_CHE2	Outputs	Main HRTIM timer outputs. They can be coupled by pairs (HRTIM_CHx1 & HRTIM_CHx2) with deadtime insertion or work independently.
HRTIM_FLT[5:1], hrtim_in_flt[5:1]	Digital input	Fault inputs: immediately disable the HRTIM outputs when asserted (5 on-chip inputs and 5 off-chip HRTIM_FLTx inputs).
hrtim_sys_flt	Digital input	System fault gathering MCU internal fault events (Clock security system, SRAM parity error, Cortex®-M7 lockup (HardFault), PVD output).
hrtim_in_sync[3:1]	Digital Input	Synchronization inputs to synchronize the whole HRTIM with other internal or external timer resources: hrtim_in_sync1: reserved hrtim_in_sync2: the source is a regular TIMx timer (via on-chip interconnect) hrtim_in_sync3: the source is an external HRTIM (via the HRTIM_SCIN input pins)
hrtim_out_sync[2:1]	Digital output	The purpose of this output is to cascade or synchronize several HRTIM instances, either on-chip or off-chip: hrtim_out_sync1: reserved hrtim_out_sync2: the destination is an off-chip HRTIM or peripheral (via HRTIM_SCOUT output pins)
hrtim_evt1[4:1]	Digital input	External events. Each of the 10 events can be selected among 4 sources, either on-chip (from other built-in peripherals: comparator, ADC analog watchdog, TIMx timers, trigger outputs) or off-chip (HRTIM_EEVx input pins)
hrtim_evt2[4:1]		
hrtim_evt3[4:1]		
hrtim_evt4[4:1]		
hrtim_evt5[4:1]		
hrtim_evt6[4:1]		
hrtim_evt7[4:1]		
hrtim_evt8[4:1]		
hrtim_evt9[4:1]		
hrtim_evt10[4:1]		
hrtim_upd_en[3:1]	Digital input	HRTIM register update enable inputs (on-chip interconnect) trigger the transfer from shadow to active registers

Table 292. HRTIM Input/output summary (continued)

Signal name	Signal type	Description
hrtim_bm_trg	Digital input	Burst mode trigger event (on-chip interconnect)
hrtim_bm_ck[4:1]	Digital input	Burst mode clock (on-chip interconnect)
hrtim_adc_trg[4:1]	Digital output	ADC start of conversion triggers
hrtim_dac_trg[3:1]	Digital output	DAC conversion update triggers
hrtim_mst_it[7:1]	Digital output	Interrupt requests
hrtim_dma[6:1]	Digital output	DMA requests
hrtim_pclk	Digital input	APB clock
hrtim_ker_ck	Digital input	HRTIM kernel clock (hereafter mentioned as f_{HRTIM}).

36.3.3 Clocks

The HRTIM must be supplied by the t_{HRTIM} system clock to offer a full resolution. All clocks present in the HRTIM are derived from this reference clock.

Definition of terms

f_{HRTIM}: main HRTIM clock (hrtim_ker_ck). All subsequent clocks are derived and synchronous with this source.

f_{DTG}: deadtime generator clock. For convenience, only the t_{DTG} period ($t_{DTG} = 1/f_{DTG}$) is used in this document.

f_{CHPFRQ}: chopper stage clock source.

f_{1STPW}: clock source defining the length of the initial pulse in chopper mode. For convenience, only the t_{1STPW} period ($t_{1STPW} = 1/f_{1STPW}$) is used in this document.

f_{BRST}: burst mode controller counter clock.

$f_{SAMPLING}$: clock needed to sample the fault or the external events inputs.

f_{FLTS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter fault events.

f_{EEVS}: clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter external events.

f_{pclk} (hrtim_pclk): APB bus clock, needed for register read/write accesses

Timer clock and prescaler

Each timer in the HRTIM has its own individual clock prescaler, which allows you to adjust the timer resolution. (See *Table 293*).

Table 293. Timer resolution and min. PWM frequency for f_{HRTIM} = 400 MHz

CKPSC[2:0](1)	Prescaling ratio	$f_{COUNTER}$	Resolution	Min PWM frequency
101	1	400 MHz	2.5 ns	6.1 kHz
110	2	400/2 MHz = 200 MHz	5 ns	3.05 kHz
111	4	400/4 MHz = 100MHz	10 ns	1.5 kHz

1. CKPSC[2:0] values from 000 to 100 are reserved.

The Full-resolution is available for edge positioning, PWM period adjustment and externally triggered pulse duration.

Initialization

At start-up, it is mandatory to initialize first the prescaler bitfields before writing the compare and period registers. Once the timer is enabled (MCEN or TxCEN bit set in the HRTIM_MCR register), the prescaler cannot be modified.

When multiple timers are enabled, the prescalers are synchronized with the prescaler of the timer that was started first.

Warning: **It is possible to have different prescaling ratios in the master and TIMA..E timers only if the counter and output behavior does not depend on other timers' information and signals. It is mandatory to configure identical prescaling ratios in these timers when one of the following events is propagated from one timing unit (or master timer) to another: output set/reset event, counter reset event, update event, external event filter or capture triggers. Prescaler factors not equal will yield to unpredictable results.**

Deadtime generator clock

The deadtime prescaler is supplied by f_{HRTIM} / 8 / $2^{(DTPRSC[2:0])}$, programmed with DTPRSC[2:0] bits in the HRTIM_DTxR register.

t_{DTG} ranges from 2.5 ns to 20 ns for f_{HRTIM} = 400 MHz.

Chopper stage clock

The chopper stage clock source f_{CHPFRQ} is derived from f_{HRTIM} with a division factor ranging from 16 to 256, so that 1.56 MHz <= f_{CHPFRQ} <= 25 MHz for f_{HRTIM} = 400 MHz.

t_{1STPW} is the length of the initial pulse in chopper mode, programmed with the STRPW[3:0] bits in the HRTIM_CHPxR register, as follows:

t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.

It uses f_{HRTIM} / 16 as clock source (25 MHz for f_{HRTIM}= 400 MHz).

Burst Mode Prescaler

The burst mode controller counter clock f_{BRST} can be supplied by several sources, among which one is derived from f_{HRTIM}.

In this case, f_{BRST} ranges from f_{HRTIM} to f_{HRTIM} / 32768 (12.2 kHz for f_{HRTIM} = 400 MHz).

Fault input sampling clock

The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{FLTS}.

f_{FLTS} is derived from f_{HRTIM} and ranges from 400 MHz to 50 MHz for f_{HRTIM} = 400 MHz.

External Event input sampling clock

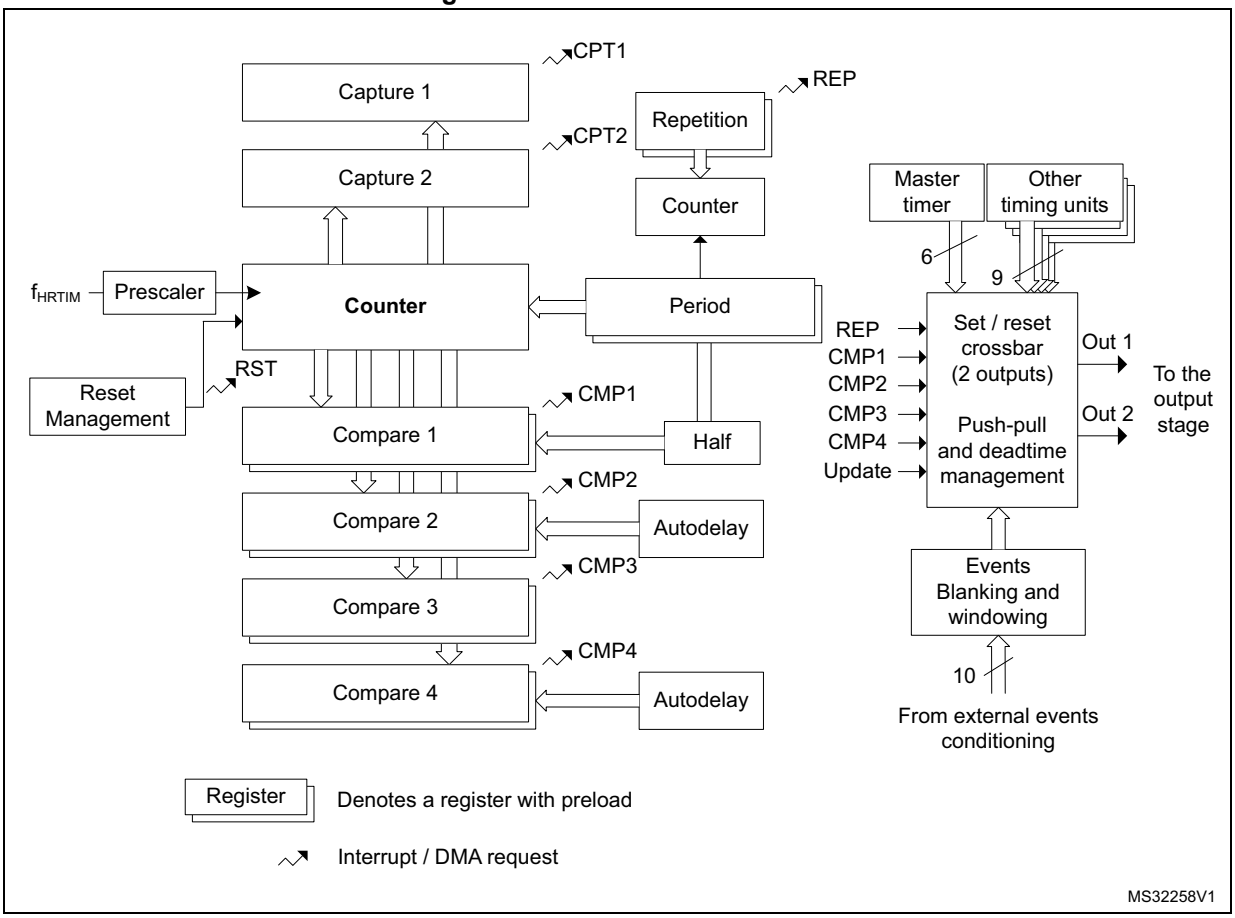
The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{EEVS}.

f_{EEVS} is derived from f_{HRTIM} and ranges from 400 MHz to 50 MHz for f_{HRTIM} = 400 MHz.

36.3.4 Timer A..E timing units

The HRTIM embeds 5 identical timing units made of a 16-bit up-counter with an auto-reload mechanism to define the counting period, 4 compare and 2 capture units, as per *Figure 281*. Each unit includes all control features for 2 outputs, so that it can operate as a standalone timer.

Figure 281. Timer A..E overview



The period and compare values must be within a lower and an upper limit related to the high-resolution implementation and listed in *Table 294*:

- The minimum value must be greater than or equal to 3 periods of the f_{HRTIM} clock
- The maximum value must be less than or equal to 0xFFFF - 1 periods of the f_{HRTIM} clock

Table 294. Period and Compare registers min and max values

CKPSC[2:0] value(1)	Min	Max
≥ 5	0x0003	0xFFFD

1. CKPSC[2:0] values < 5 are reserved.

Note: A compare value greater than the period register value will not generate a compare match event.

Counter operating mode

Timer A..E can operate in continuous (free-running) mode or in single-shot manner where counting is started by a reset event, using the CONT bit in the HRTIM_TIMxCR control register. An additional RETRIG bit allows you to select whether the single-shot operation is retriggerable or non-retriggerable. Details of operation are summarized on *Table 295* and on *Figure 282* and *Figure 283*.

Table 295. Timer operating modes

CONT	RETRIG	Operating mode	Start / Stop conditions Clocking and event generation
0	0	Single-shot Non-retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A first reset event starts the counting and any subsequent reset is ignored until the counter reaches the PER value. The PER event is then generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
0	1	Single-shot Retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A reset event starts the counting if the counter is stopped, otherwise it clears the counter. When the counter reaches the PER value, the PER event is generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
1	X	Continuous mode	Setting the TxEN bit enables the timer and starts the counter simultaneously. When the counter reaches the PER value, it rolls-over to 0x0000 and resumes counting. The counter can be reset at any time.

The TxEN bit can be cleared at any time to disable the timer and stop the counting.

Figure 282. Continuous timer operation

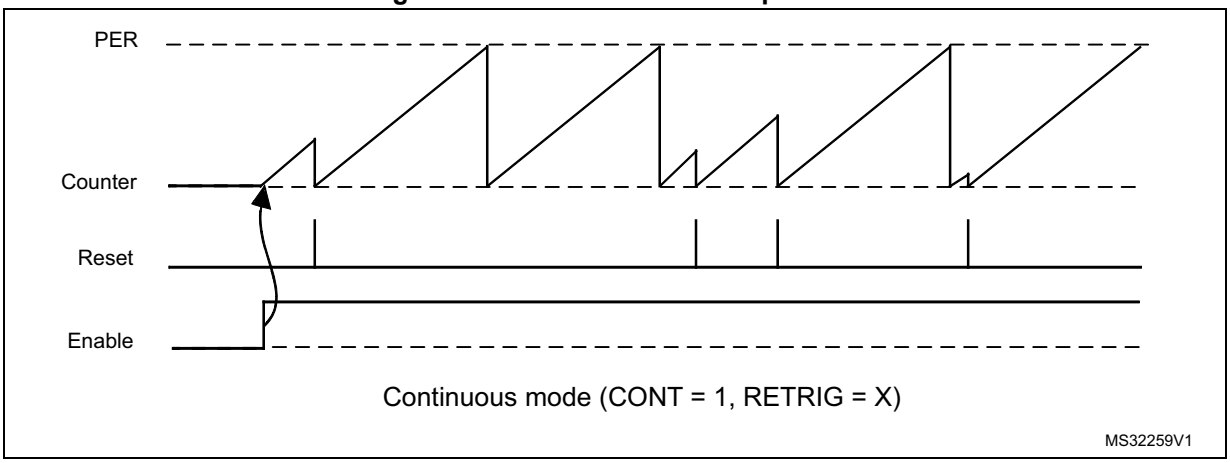
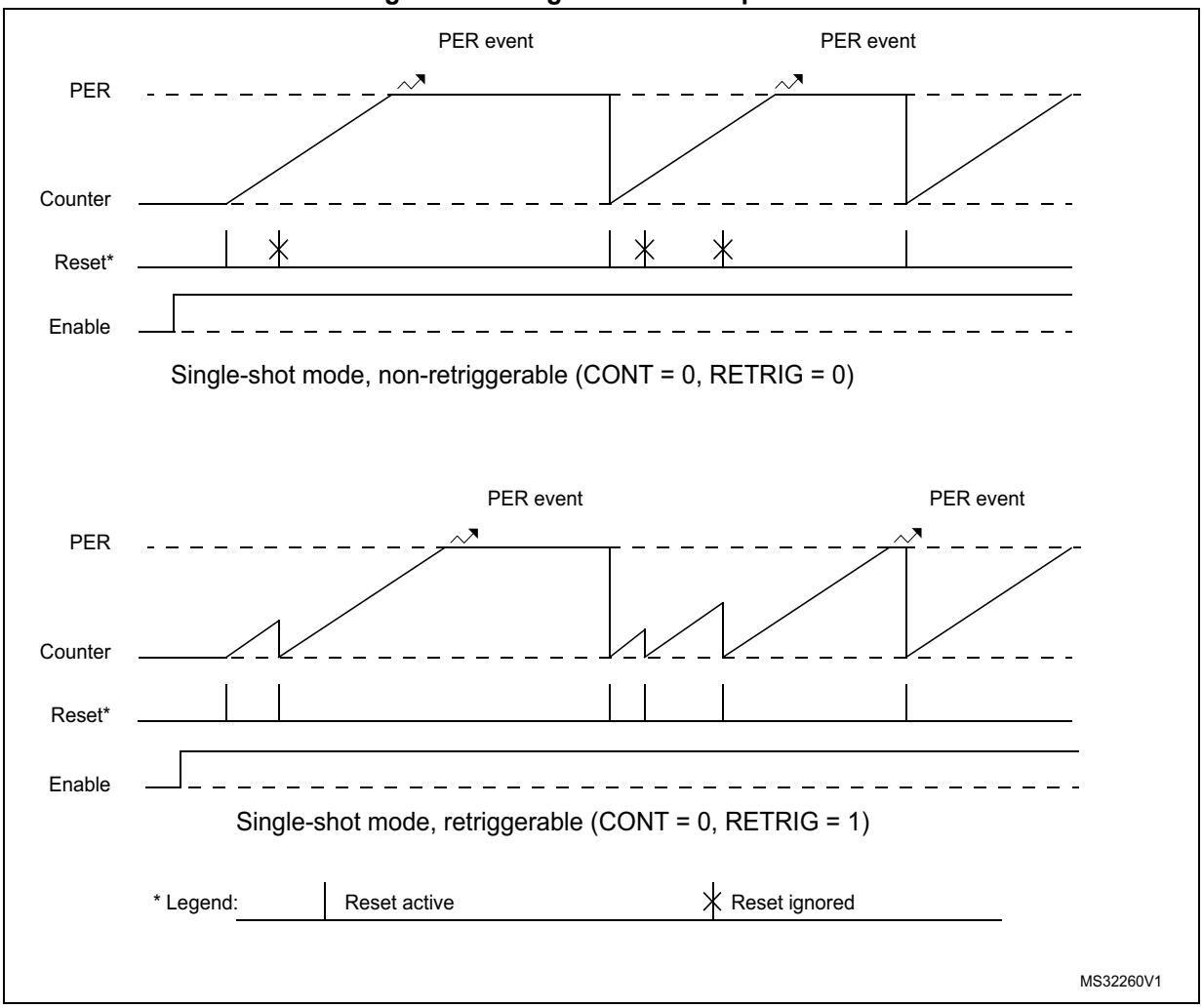


Figure 283. Single-shot timer operation



Roll-over event

A counter roll-over event is generated when the counter goes back to 0 after having reached the period value set in the HRTIM_PERxR register in continuous mode.

This event is used for multiple purposes in the HRTIM:

- To set/reset the outputs
- To trigger the register content update (transfer from preload to active)
- To trigger an IRQ or a DMA request
- To serve as a burst mode clock source or a burst start trigger
- as an ADC trigger
- To decrement the repetition counter

If the initial counter value is above the period value when the timer is started, or if a new period is set while the counter is already above this value, the counter is not reset: it will overflow at the maximum period value and the repetition counter will not decrement.

Timer reset

The reset of the timing unit counter can be triggered by up to 30 events that can be selected simultaneously in the HRTIM_RSTxR register, among the following sources:

- The timing unit: Compare 2, Compare 4 and Update (3 events)
- The master timer: Reset and Compare 1..4 (5 events)
- The external events EXTEVNT1..10 (10 events)
- All other timing units (e.g. Timer B..E for timer A): Compare 1, 2 and 4 (12 events)

Several events can be selected simultaneously to handle multiple reset sources. In this case, the multiple reset requests are ORed. When 2 counter reset events are generated within the same f_{HRTIM} clock cycle, the last counter reset is taken into account.

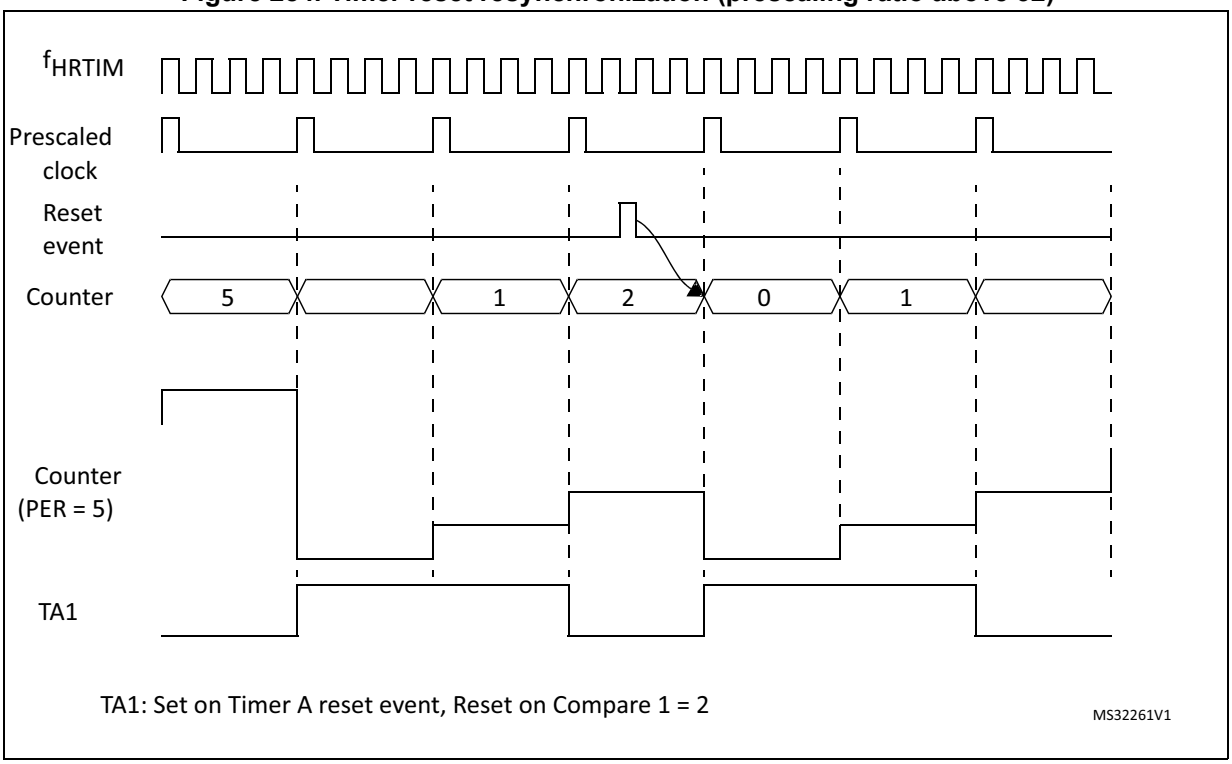
Additionally, it is possible to do a software reset of the counter using the TxRST bits in the HRTIM_CR2 register. These control bits are grouped into a single register to allow the simultaneous reset of several counters.

The reset requests are taken into account only once the related counters are enabled (TxCEN bit set).

When the f_{HRTIM} clock prescaling ratio is above 1, the counter reset event is delayed to the next active edge of the prescaled clock. This allows to maintain a jitterless waveform generation when an output transition is synchronized to the reset event (typically a constant Ton time converter).

Figure 284 shows how the reset is handled for a clock prescaling ratio of 4 (f_{HRTIM} divided by 4).

Figure 284. Timer reset resynchronization (prescaling ratio above 32)



Repetition counter

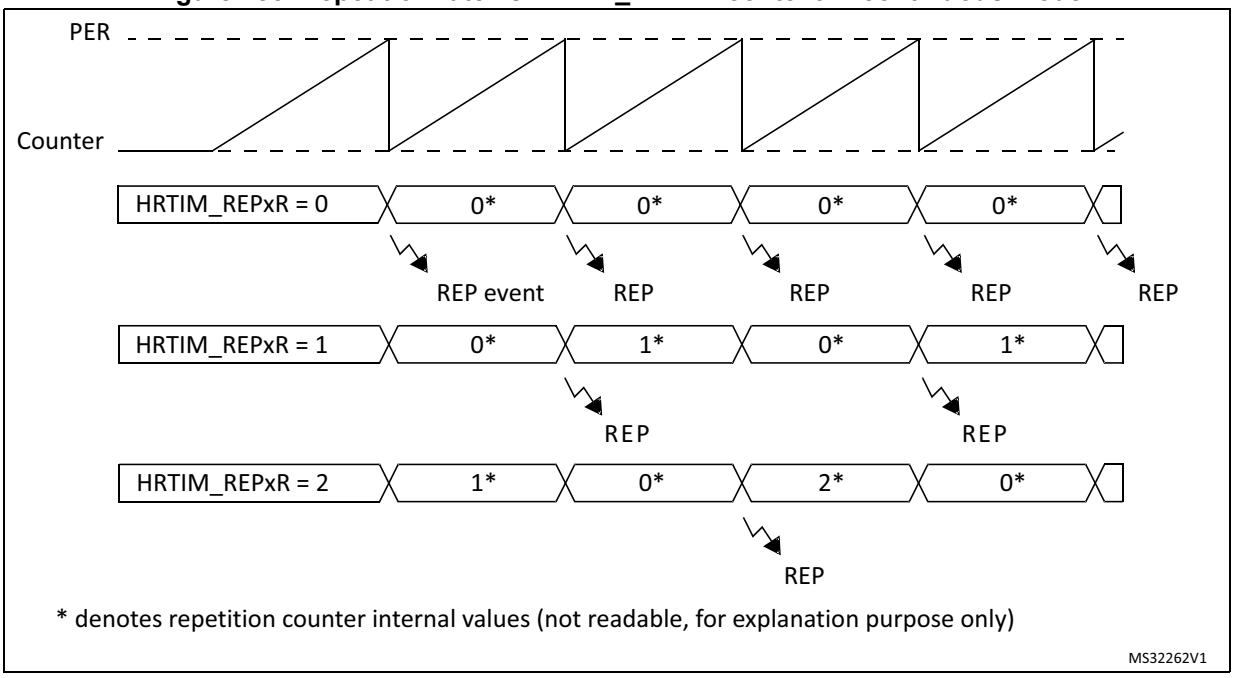
A common software practice is to have an interrupt generated when the period value is reached, so that the maximum amount of time is left for processing before the next period begins. The main purpose of the repetition counter is to adjust the period interrupt rate and off-load the CPU by decoupling the switching frequency and the interrupt frequency.

The timing units have a repetition counter. This counter cannot be read, but solely programmed with an auto-reload value in the HRTIM_REPxR register.

The repetition counter is initialized with the content of the HRTIM_REPxR register when the timer is enabled (TXCEN bit set). Once the timer has been enabled, any time the counter is cleared, either due to a reset event or due to a counter roll-over, the repetition counter is decreased. When it reaches zero, a REP interrupt or a DMA request is issued if enabled (REPIE and REPDE bits in the HRTIM_DIER register).

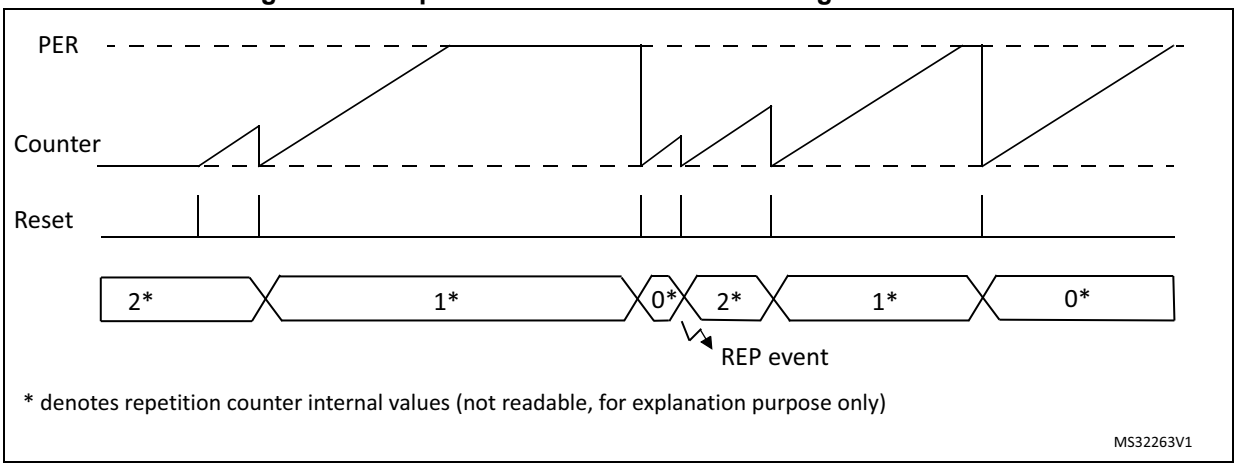
If the HRTIM_REPxR register is set to 0, an interrupt is generated for each and every period. For any value above 0, a REP interrupt is generated after (HRTIM_REPxR + 1) periods. *Figure 285* presents the repetition counter operation for various values, in continuous mode.

Figure 285. Repetition rate vs HRTIM_REPxR content in continuous mode



The repetition counter can also be used when the counter is reset before reaching the period value (variable frequency operation) either in continuous or in single-shot mode (*Figure 286* here-below). The reset causes the repetition counter to be decremented, at the exception of the very first start following counter enable (TxCEN bit set).

Figure 286. Repetition counter behavior in single-shot mode



A reset or start event from the hrtim_in_sync[3:1] source causes the repetition to be decremented as any other reset. However, in SYNCIN-started single-shot mode (SYNCSTRTx bit set in the HRTIM_TIMxCR register), the repetition counter will be decremented only on the 1st reset event following the period. Any subsequent reset will not alter the repetition counter until the counter is re-started by a new request on hrtim_in_sync[3:1] inputs.

Set / reset crossbar

A “set” event correspond to a transition to the output active state, while a “reset” event corresponds to a transition to the output inactive state.
The polarity of the waveform is defined in the output stage to accommodate positive or negative logic external components: an active level corresponds to a logic level 1 for a positive polarity (POLx = 0), and to a logic level 0 for a negative polarity (POLx = 1).

Each of the timing units handles the set/reset crossbar for two outputs. These 2 outputs can be set, reset or toggled by up to 32 events that can be selected among the following sources:

- The timing unit: Period, Compare 1..4, register update (6 events)
- The master timer: Period, Compare 1..4, HRTIM synchronization (6 events)
- All other timing units (e.g. Timer B..E for timer A): TIMEVNT1..9 (9 events described in *Table 296*)
- The external events EXTEVNT1..10 (10 events)
- A software forcing (1 event)

The event sources are ORed and multiple events can be simultaneously selected.

Each output is controlled by two 32-bit registers, one coding for the set (HRTIM_SETxyR) and another one for the reset (HRTIM_RSTxyR), where x stands for the timing unit: A..E and y stands for the output 1or 2 (e.g. HRTIM_SETA1R, HRTIM_RSTC2R,...).

If the same event is selected for both set and reset, it will toggle the output. It is not possible to toggle the output state more than one time per t_{HRTIM} period: in case of two consecutive toggling events within the same cycle, only the first one is considered.

The set and reset requests are taken into account only once the counter is enabled (TxCEN bit set), except if the software is forcing a request to allow the prepositioning of the outputs at timer start-up.

Table 296 summarizes the events from other timing units that can be used to set and reset the outputs. The number corresponds to the timer events (such as TIMEVNTx) listed in the register, and empty locations are indicating non-available events.

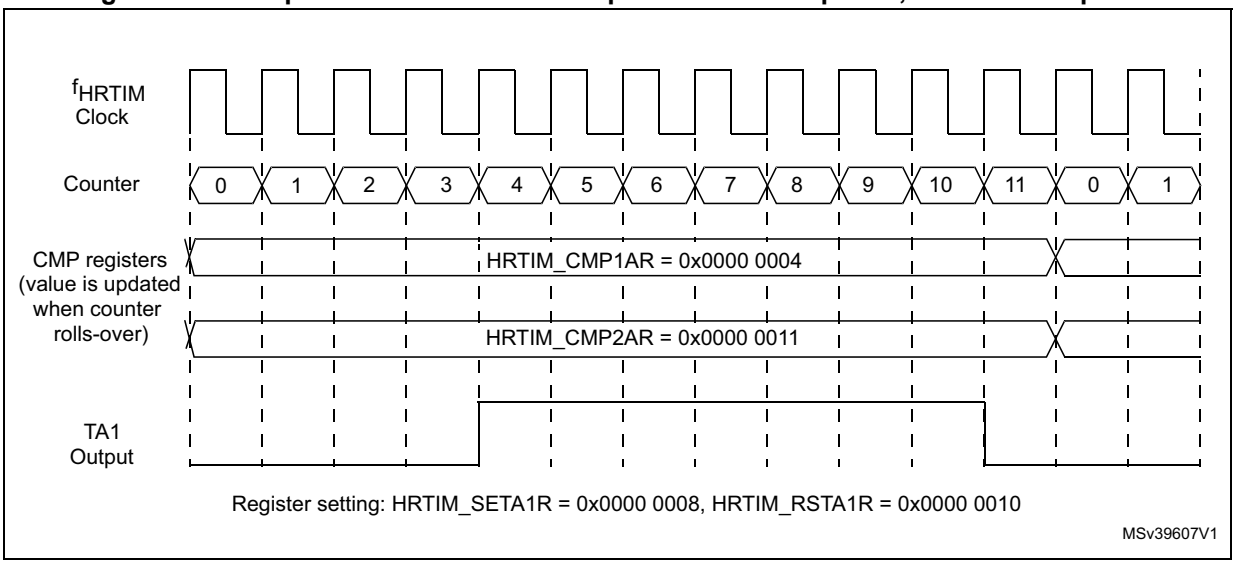
For instance, Timer A outputs can be set or reset by the following events: Timer B Compare1, 2 and 4, Timer C Compare 2 and 3,... and Timer E Compare 3 will be listed as TIMEVNT8 in HRTIM_SETA1R.

Table 296. Events mapping across Timer A to E

Source		Timer A				Timer B				Timer C				Timer D				Timer E			
		CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4
Destination	Timer A	-	-	-	-	1	2	-	3	-	4	5	-	6	7	-	-	-	-	8	9
	Timer B	1	2	-	3	-	-	-	-	-	-	4	5	-	-	6	7	8	9	-	-
	Timer C	-	1	2	-	-	3	4	-	-	-	-	-	-	5	-	6	-	7	8	9
	Timer D	1	-	-	2	-	3	-	4	5	-	6	7	-	-	-	-	8	-	-	9
	Timer E	-	-	1	2	-	-	3	4	5	6	-	-	7	8	-	9	-	-	-	-

Figure 287 represents how a PWM signal is generated using two compare events.

Figure 287. Compare events action on outputs: set on compare 1, reset on compare 2



Set/Reset on Update events

Half mode

This mode aims at generating square signal with fixed 50% duty cycle and variable frequency (typically for converters using resonant topologies). It allows to have the duty cycle automatically forced to half of the period value when a new period is programmed.

This mode is enabled by writing HALF bit to 1 in the HRTIM_TIMxCR register. When the HRTIM_PERxR register is written, it causes an automatic update of the Compare 1 value with HRTIM_PERxR/2 value.

The output on which a square wave is generated must be programmed to have one transition on CMP1 event, and one transition on the period event, as follows:

– HRTIM_SETxyR = 0x0000 0008, HRTIM_RSTxyR = 0x0000 0004, or

– HRTIM_SETxyR = 0x0000 0004, HRTIM_RSTxyR = 0x0000 0008

The HALF mode overrides the content of the HRTIM_CMP1xR register. The access to the HRTIM_PERxR register only causes Compare 1 internal register to be updated. The user-accessible HRTIM_CMP1xR register is not updated with the HRTIM_PERxR / 2 value.

When the preload is enabled (PREEN = 1, MUDIS, TxUDIS), Compare 1 active register is refreshed on the Update event. If the preload is disabled (PREEN= 0), Compare 1 active register is updated as soon as HRTIM_PERxR is written.

The period must be greater than or equal to 6 periods of the f_{HRTIM} clock when the HALF mode is enabled.

Capture

The timing unit has the capability to capture the counter value, triggered by internal and external events. The purpose is to:

- measure events arrival timings or occurrence intervals
- update Compare 2 and Compare 4 values in auto-delayed mode (see *Auto-delayed mode*).

The capture is done with f_{HRTIM} resolution.

The timer has 2 capture registers: HRTIM_CPT1xR and HRTIM_CPT2xR. The capture triggers are programmed in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers.

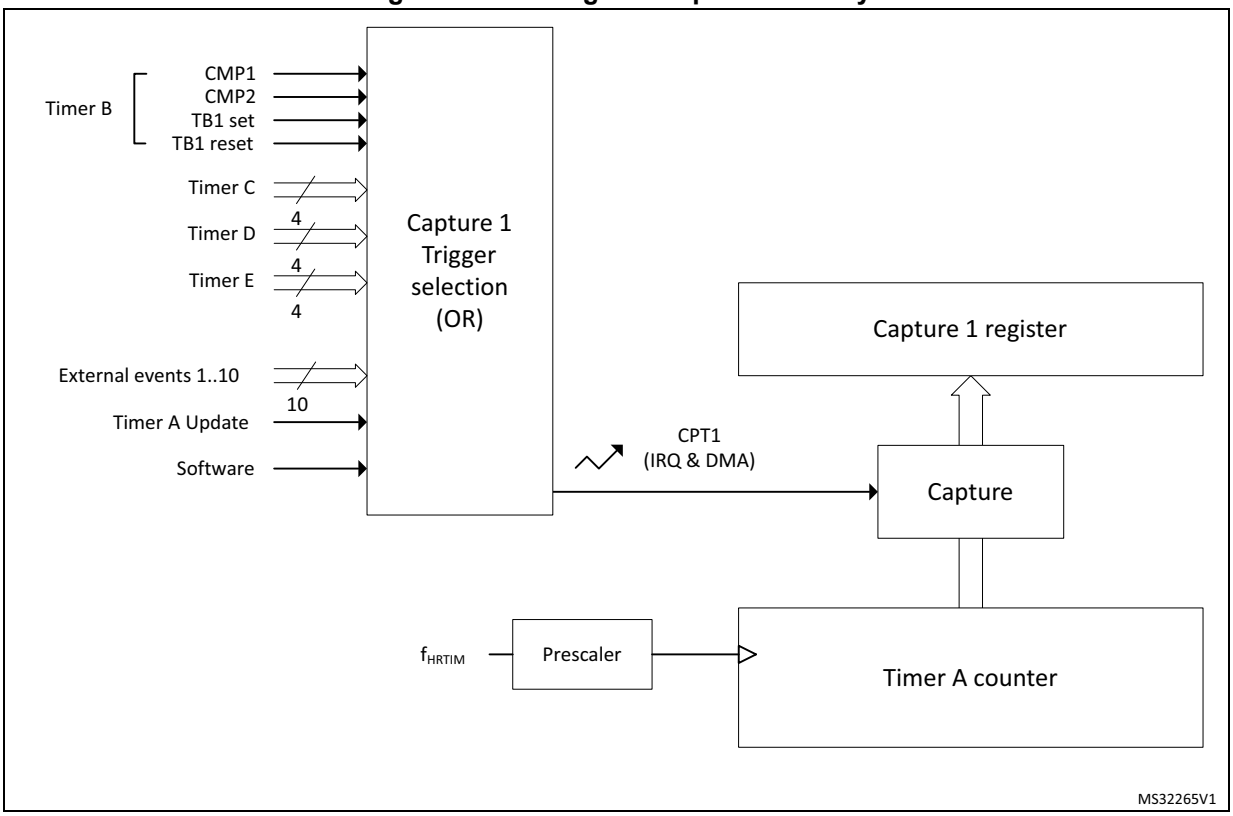
The capture of the timing unit counter can be triggered by up to 28 events that can be selected simultaneously in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers, among the following sources:

- The external events, EXTEVNT1..10 (10 events)
- All other timing units (e.g. Timer B..E for timer A): Compare 1, 2 and output 1 set/reset events (16 events)
- The timing unit: Update (1 event)
- A software capture (1 event)

Several events can be selected simultaneously to handle multiple capture triggers. In this case, the concurrent trigger requests are ORed. The capture can generate an interrupt or a DMA request when CPTxIE and CPTxDE bits are set in the HRTIM_TIMxDIER register.

Over-capture is not prevented by the circuitry: a new capture is triggered even if the previous value was not read, or if the capture flag was not cleared.

Figure 288. Timing unit capture circuitry



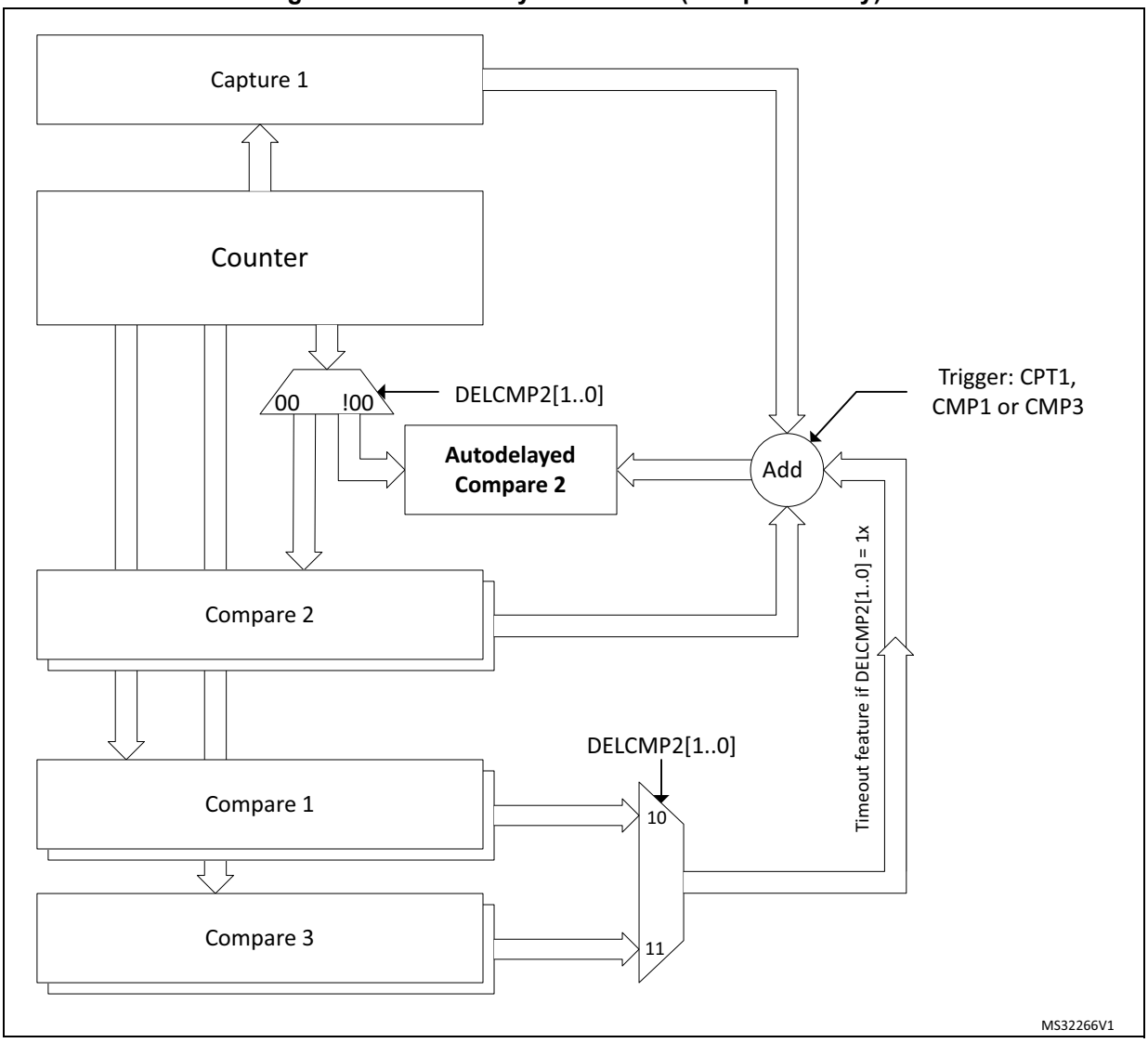
Auto-delayed mode

This mode allows to have compare events generated relatively to capture events, so that for instance an output change can happen with a programmed timing following a capture. In this case, the compare match occurs independently from the timer counter value. It enables the generation of waveforms with timings synchronized to external events without the need of software computation and interrupt servicing.

As long as no capture is triggered, the content of the HRTIM_CMPxR register is ignored (no compare event is generated when the counter value matches the Compare value. Once the capture is triggered, the compare value programmed in HRTIM_CMPxR is summed with the captured counter value in HRTIM_CPTxyR, and it updates the internal auto-delayed compare register, as seen on *Figure 289*. The auto-delayed compare register is internal to the timing unit and cannot be read. The HRTIM_CMPxR preload register is not modified after the calculation.

This feature is available only for Compare 2 and Compare 4 registers. Compare 2 is associated with capture 1, while Compare 4 is associated with capture 2. HRTIM_CMP2xR and HRTIM_CMP4xR Compares cannot be programmed with a value below 3 f_{HRTIM} clock periods, as in the regular mode.

Figure 289. Auto-delayed overview (Compare 2 only)



The auto-delayed Compare is only valid from the capture up to the period event: once the counter has reached the period value, the system is re-armed with Compare disabled until a capture occurs.

DELCMP2[1:0] and DELCMP4[1:0] bits in HRTIM_TIMxCR register allow to configure the auto-delayed mode as follows:

- 00
  Regular compare mode: HRTIM_CMP2xR and HRTIM_CMP4xR register contents are directly compared with the counter value.
- 01
  Auto-delayed mode: Compare 2 and Compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event.

- 1X
  Auto-delayed mode with timeout: Compare 2 and Compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event or after a Compare 1 match (DELCMPx[1:0]= 10) or a Compare 3 match (DELCMPx[1:0]= 11) to have a timeout function if capture 1/2 event is missing.

When the capture occurs, the comparison is done with the (HRTIM_CMP2/4xR + HRTIM_CPT1/2xR) value. If no capture is triggered within the period, the behavior depends on the DELCMPx[1:0] value:

- DELCMPx[1:0] = 01: the compare event is not generated
- DELCMPx[1:0] = 10 or 11: the comparison is done with the sum of the 2 compares (for instance HRTIM_CMP2xR + HRTIM_CMP1xR). The captures are not taken into account if they are triggered after CMPx + CMP1 (resp. CMPx + CMP3).

The captures are enabled again at the beginning of the next PWM period.

If the result of the auto-delayed summation is above 0xFFFF (overflow), the value is ignored and no compare event will be generated until a new period is started.

Note: DELCMPx[1:0] bitfield must be reset when reprogrammed from one value to the other to re-initialize properly the auto-delayed mechanism, for instance:

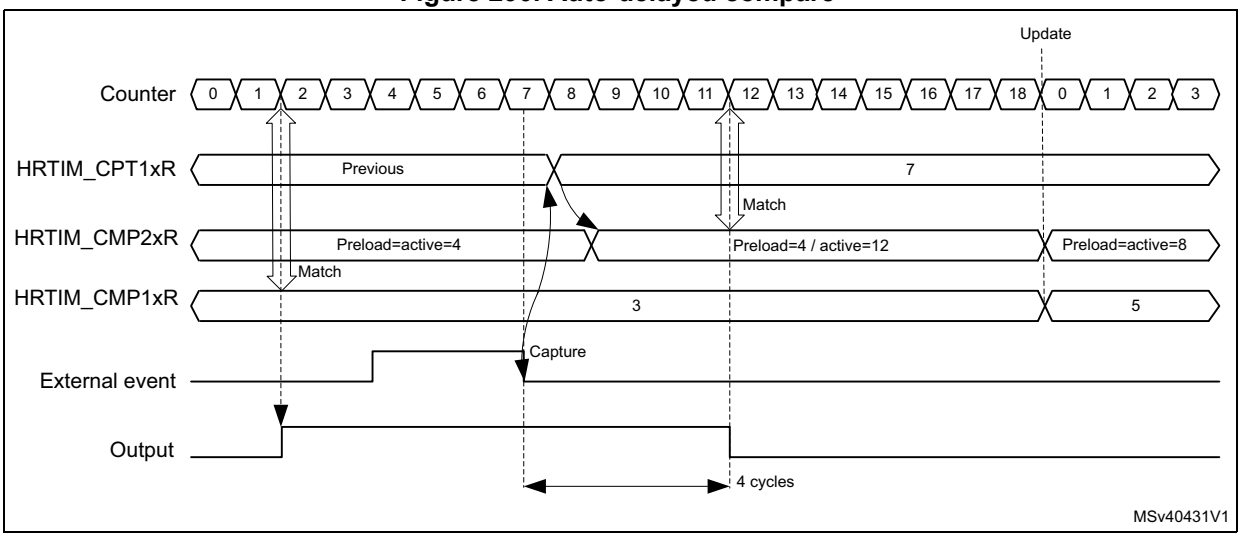
- DELCMPx[1:0] = 10
- DELCMPx[1:0] = 00
- DELCMPx[1:0] = 11

As an example, *Figure 290* shows how the following signal can be generated:

- Output set when the counter is equal to Compare 1 value
- Output reset 4 cycles after a falling edge on a given external event

Note: To simplify the figure, the external event signal is shown without any resynchronization delay: practically, there is a delay of 1 to 2 f_{HRTIM} clock periods between the falling edge and the capture event due to an internal resynchronization stage which is necessary to process external input signals.

Figure 290. Auto-delayed compare



A regular compare channel (e.g. Compare 1) is used for the output set: as soon as the counter matches the content of the compare register, the output goes to its active state.

A delayed compare is used for the output reset: the compare event can be generated only if a capture event has occurred. No event is generated when the counter matches the delayed compare value (counter = 4). Once the capture event has been triggered by the external event, the content of the capture register is summed to the delayed compare value to have the new compare value. In the example, the auto-delayed value 4 is summed to the capture equal to 7 to give a value of 12 in the auto-delayed compare register. From this time on, the compare event can be generated and will happen when the counter is equal to 12, causing the output to be reset.

Overcapture management in auto-delayed mode

Overcapture is prevented when the auto-delayed mode is enabled (DELCMPx[1:0] = 01, 10, 11).

When multiple capture requests occur within the same counting period, only the first capture is taken into account to compute the auto-delayed compare value. A new capture is possible only:

- Once the auto-delayed compare has matched the counter value (compare event)
- Once the counter has rolled over (period)
- Once the timer has been reset

Changing auto-delayed compare values

When the auto-delayed compare value is preloaded (PREEN bit set), the new compare value is taken into account on the next coming update event (for instance on the period event), regardless of when the compare register was written and if the capture occurred (see *Figure 290*, where the delay is changed when the counter rolls over).

When the preload is disabled (PREEN bit reset), the new compare value is taken into account immediately, even if it is modified after the capture event has occurred, as per the example below:

1. At t1, DELCMP2 = 1.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, a capture event occurs capturing the value CPTR1 = 0x20. => comparison enabled, compare value = 0x60
4. At t4, CMP2_act = 0x100 (before the counter reached value CPTR1 + 0x40) => comparison still enabled, new compare value = 0x120
5. At t5, the counter reaches the period value => comparison disabled, cmp2_act = 0x100

Similarly, if the CMP1(CMP3) value changes while DELCMPx = 10 or 11, and preload is disabled:

1. At t1, DELCMP2 = 2.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, CMP3 event occurs - CMP3_act = 0x50 before capture 1 event occurs => comparison enabled, compare value = 0x90
4. At t4, CMP3_act = 0x100 (before the counter reached value 0x90) => comparison still enabled, Compare 2 event will occur at = 0x140

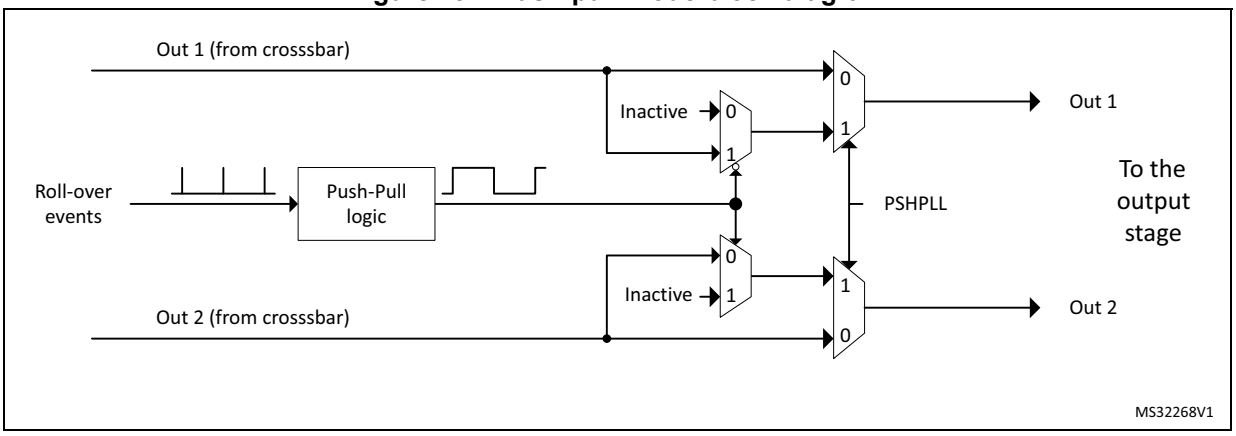
Push-pull mode

This mode primarily aims at driving converters using push-pull topologies. It also needs to be enabled when the delayed idle protection is required, typically for resonant converters (refer to *Section 36.3.9: Delayed Protection*).

The push-pull mode is enabled by setting PSHPLL bit in the HRTIM_TIMxCR register.

It applies the signals generated by the crossbar to output 1 and output 2 alternatively, on the period basis, maintaining the other output to its inactive state. The redirection rate (push-pull frequency) is defined by the timer's period event, as shown on *Figure 291*. The push-pull period is twice the timer counting period.

Figure 291. Push-pull mode block diagram



The push-pull mode is only available when the timer operates in continuous mode: the counter must not be reset once it has been enabled (TxCEN bit set). It is necessary to disable the timer to stop a push-pull operation and to reset the counter before re-enabling it.

The signal shape is defined using HRTIM_SETxyR and HRTIM_RSTxyR for both outputs. It is necessary to have HRTIM_SETx1R = HRTIM_SETx2R and HRTIM_RSTx1R = HRTIM_RSTx2R to have both outputs with identical waveforms and to achieve a balanced operation. Still, it is possible to have different programming on both outputs for other uses.

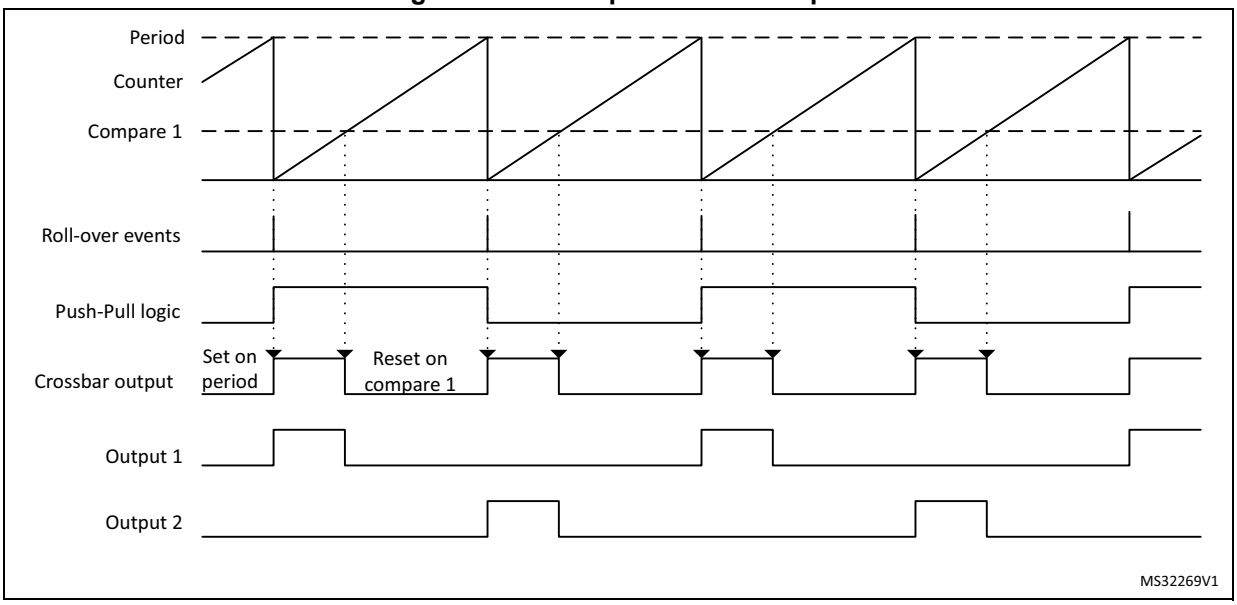
Note: The push-pull operation cannot be used when a deadtime is enabled (mutually exclusive functions).

The CPPSAT status bit in HRTIM_TIMxISR indicates on which output the signal is currently active. CPPSTAT is reset when the push-pull mode is disabled.

In the example given on *Figure 292*, the timer internal waveform is defined as follows:

- Output set on period event
- Output reset on Compare 1 match event

Figure 292. Push-pull mode example



Deadtime

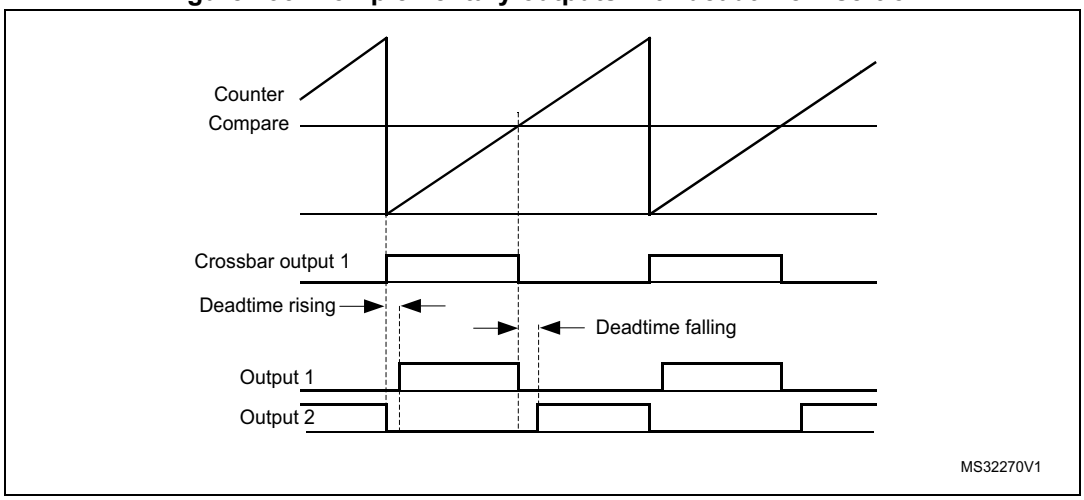
A deadtime insertion unit allows to generate a couple of complementary signals from a single reference waveform, with programmable delays between active state transitions. This is commonly used for topologies using half-bridges or full bridges. It simplifies the software: only 1 waveform is programmed and controlled to drive two outputs.

The Dead time insertion is enabled by setting DTEN bit in HRTIM_OUTxR register. The complementary signals are built based on the reference waveform defined for output 1, using HRTIM_SETx1R and HRTIM_RSTx1R registers: HRTIM_SETx2R and HRTIM_RSTx2R registers are not significant when DTEN bit is set.

Note: The deadtime cannot be used simultaneously with the push-pull mode.

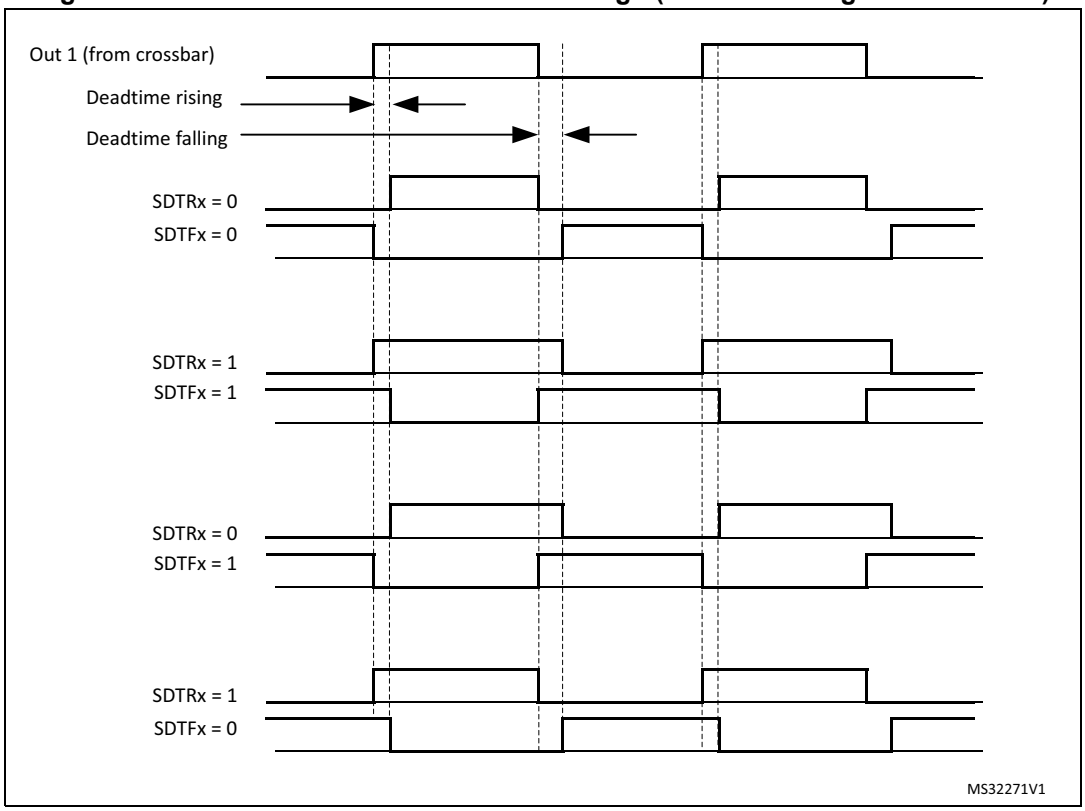
Two deadtimes can be defined in relationship with the rising edge and the falling edge of the reference waveform, as in *Figure 293*.

Figure 293. Complementary outputs with deadtime insertion



Negative deadtime values can be defined when some control overlap is required. This is done using the deadtime sign bits (SDTFx and SDTRx bits in HRTIM_DTxR register). *Figure 294* shows complementary signal waveforms depending on respective signs.

Figure 294. Deadtime insertion vs deadtime sign (1 indicates negative deadtime)



The deadtime values are defined with DTFx[8:0] and DTRx[8:0] bitfields and based on a specific clock prescaled according to DTPRSC[2:0] bits, as follows:

t_{DTx} = +/- DTx[8:0] x t_{DTG}

where x is either R or F and $t_{DTG} = (2^{(DTPRSC[2:0])}) \times t_{HRTIM}$.

Table 297 gives the resolution and maximum absolute values depending on the prescaler value.

Table 297. Deadtime resolution and max absolute values

DTPRSC[2:0](1)	t_{DTG}	t_{DTx} max	f_{HRTIM}= 400 MHz	
			t_{DTG} (ns)	$\|t_{DTx}\|$ max (µs)
011	t_{HRTIM}	511 * t_{DTG}	2.5	1.28
100	2 * t_{HRTIM}		5	2.56
101	4 * t_{HRTIM}		10	5.11
110	8 * t_{HRTIM}		20	10.22
111	16 * t_{HRTIM}		40	20.44

1. DTPRSC[2:0] values 000, 001, 010 are reserved.

Figure 295 to *Figure 298* present how the deadtime generator behaves for reference waveforms with pulsewidth below the deadtime values, for all deadtime configurations.

Figure 295. Complementary outputs for low pulse width (SDTRx = SDTFx = 0)

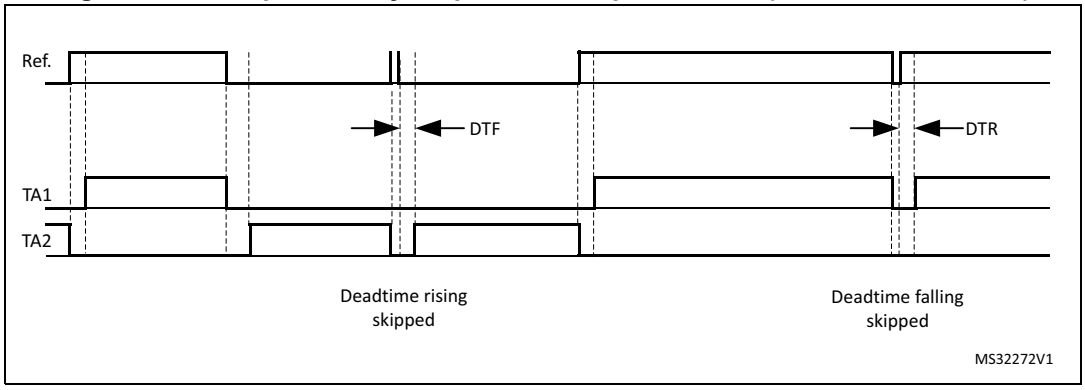


Figure 296. Complementary outputs for low pulse width (SDTRx = SDTFx = 1)

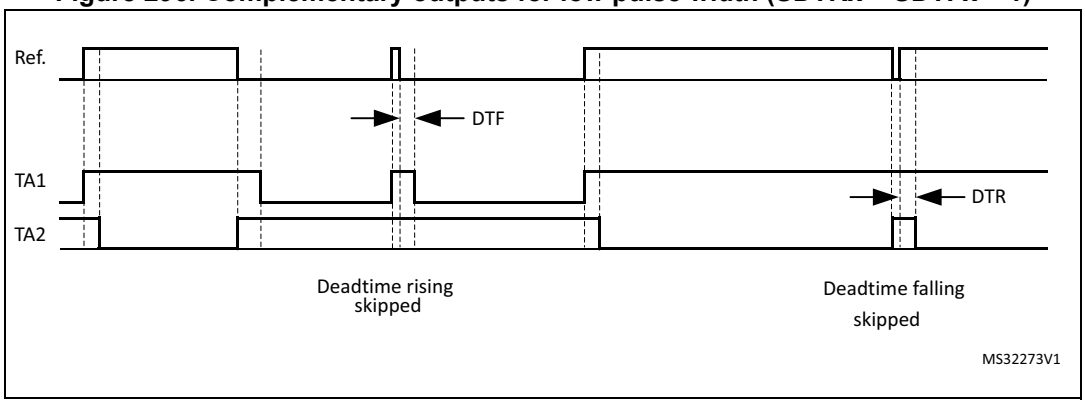


Figure 297. Complementary outputs for low pulse width (SDTRx = 0, SDTFx = 1)

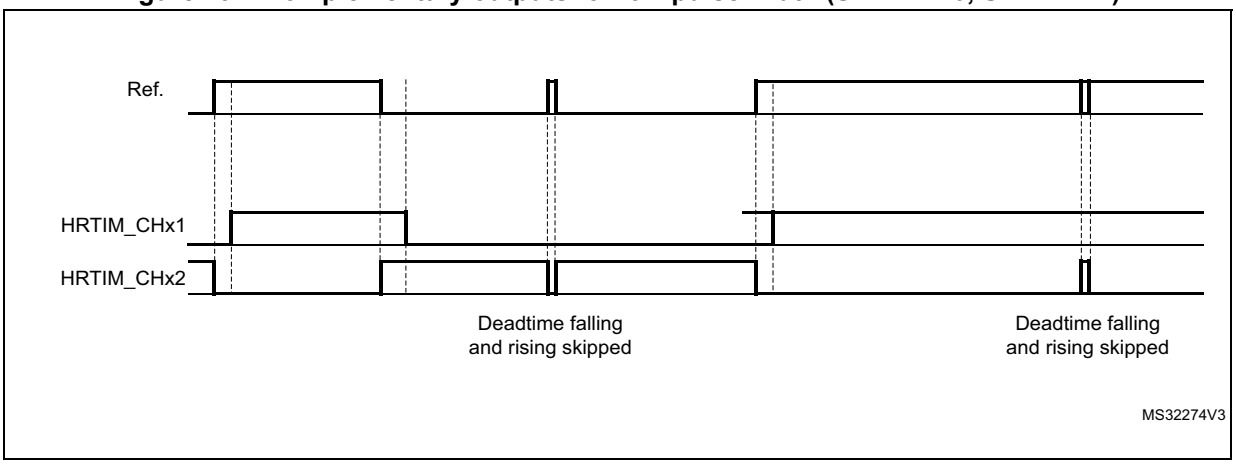
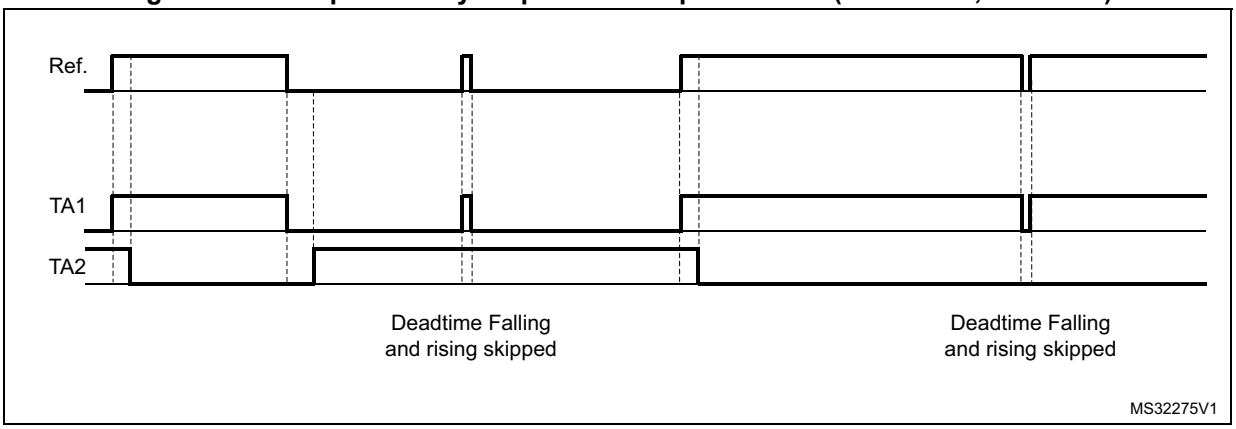


Figure 298. Complementary outputs for low pulse width (SDTRx = 1, SDTFx=0)



For safety purposes, it is possible to prevent any spurious write into the deadtime registers by locking the sign and/or the value of the deadtime using DTFLKx, DTRLKx, DTFSLKx and DTRSLKx. Once these bits are set, the related bits and bitfields are becoming read only until the next system reset.

Caution: DTEN bit must not be changed in the following cases:
- When the timer is enabled (TxEN bit set)
- When the timer outputs are set/reset by another timer (while TxEN is reset)

Otherwise, an unpredictable behavior would result.
It is therefore necessary to disable the timer (TxCEN bit reset) and have the corresponding outputs disabled.

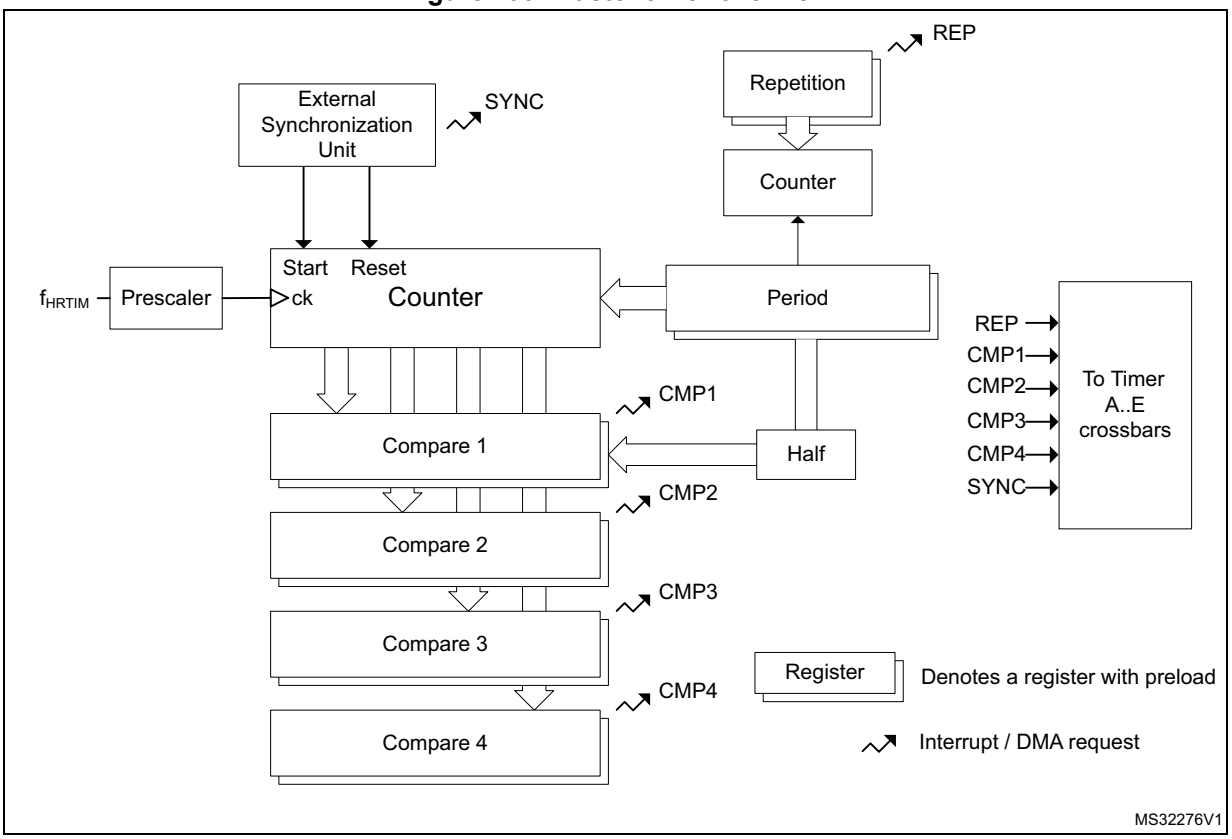
For the particular case where DTEN must be set while the burst mode is enabled with a deadtime upon entry (BME = 1, DIDL = 1, IDLEM = 1), it is necessary to force the two outputs in their IDLES state by software commands (SST, RST bits) before setting DTEN bit. This is to avoid any side effect resulting from a burst mode entry that would happen immediately before a deadtime enable.

36.3.5 Master timer

The main purpose of the master timer is to provide common signals to the 5 timing units, either for synchronization purpose or to set/reset outputs. It does not have direct control over any outputs, but still can be used indirectly by the set/reset crossbars.

Figure 299 provides an overview of the master timer.

Figure 299. Master timer overview



The master timer is based on the very same architecture as the timing units, with the following differences:

- It does not have outputs associated with, nor output related control
- It does not have its own crossbar unit, nor push-pull or deadtime mode
- It can only be reset by the external synchronization circuitry
- It does not have a capture unit, nor the auto-delayed mode
- It does not include external event blanking and windowing circuitry
- It has a limited set of interrupt / DMA requests: Compare 1..4, repetition, register update and external synchronization event.

The master timer control register includes all the timer enable bits, for the master and Timer A..E timing units. This allows to have all timer synchronously started with a single write access.

It also handles the external synchronization for the whole HRTIM timer (see *Section 36.3.17: Synchronizing the HRTIM with other timers or HRTIM instances*), with both MCU internal and external (inputs/outputs) resources.

Master timer control registers are mapped with the same offset as the timing units' registers.

36.3.6 Set/reset events priorities and narrow pulses management

This section describes how the output waveform is generated when several set and/or reset requests are occurring within 3 consecutive t_{HRTIM} periods.

An arbitration is performed during each t_{HRTIM} period, in 2 steps:

1. For each active event, the desired output transition is determined (set, reset or toggle).
2. A predefined arbitration is performed among the active events (from highest to lowest priority CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1 ⟶ PER, see *Concurrent set request / Concurrent reset requests*.

When set and reset requests from two different sources are simultaneous, the reset action has the highest priority.

Concurrent set request / Concurrent reset requests

When multiple sources are selected for a set event, an arbitration is performed when the set requests occur within the same f_{HRTIM} clock period.

In case of multiple requests from adjacent timers (TIMEVNT1..9), the request which occurs first is taken into account. The arbitration is done in 2 steps, depending on the source (from the highest to the lowest priority): CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1.

If multiple requests from the master timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and a single request will be taken into account (from the highest to the lowest priority):

MSTCMP4 → MSTCMP3 → MSTCMP2 → MSTCMP1 → MSTCMPER

When multiple requests internal to the timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and the requests are taken with the following priority, whatever the effective timing (from highest to lowest):

CMP4 → CMP3 → CMP2 → CMP1 → PER

Note: Practically, this is of a primary importance only when using auto-delayed Compare 2 and Compare 4 simultaneously (i.e. when the effective set/reset cannot be determined a priori because it is related to an external event). In this case, the highest priority signal must be affected to the CMP4 event.

Last, the highest priority is given to non timing-related: EXTEVNT1..10, RESYNC (coming from SYNC event if SYNCRSTx or SYNCSTRTx is set or from a software reset), update and software set (SST).

As a summary, in case of simultaneous events, the effective set (reset) event will be arbitrated between:

- Any TIMEVNT1..9 event
- A single source from the master (as per the fixed arbitration given above)
- A single source from the timer
- The "non timing-related events".

The same arbitration principle applies for concurrent reset requests. In this case, the reset request has the highest priority.

A set or reset event occurring within the prescaler clock cycle is delayed to the next active edge of the prescaled clock (as for a counter reset), even if the arbitration is still performed every t_{HRTIM} cycle.

If a reset event is followed by a set event within the same prescaler clock cycle, the latest event will be considered.

36.3.7 External events global conditioning

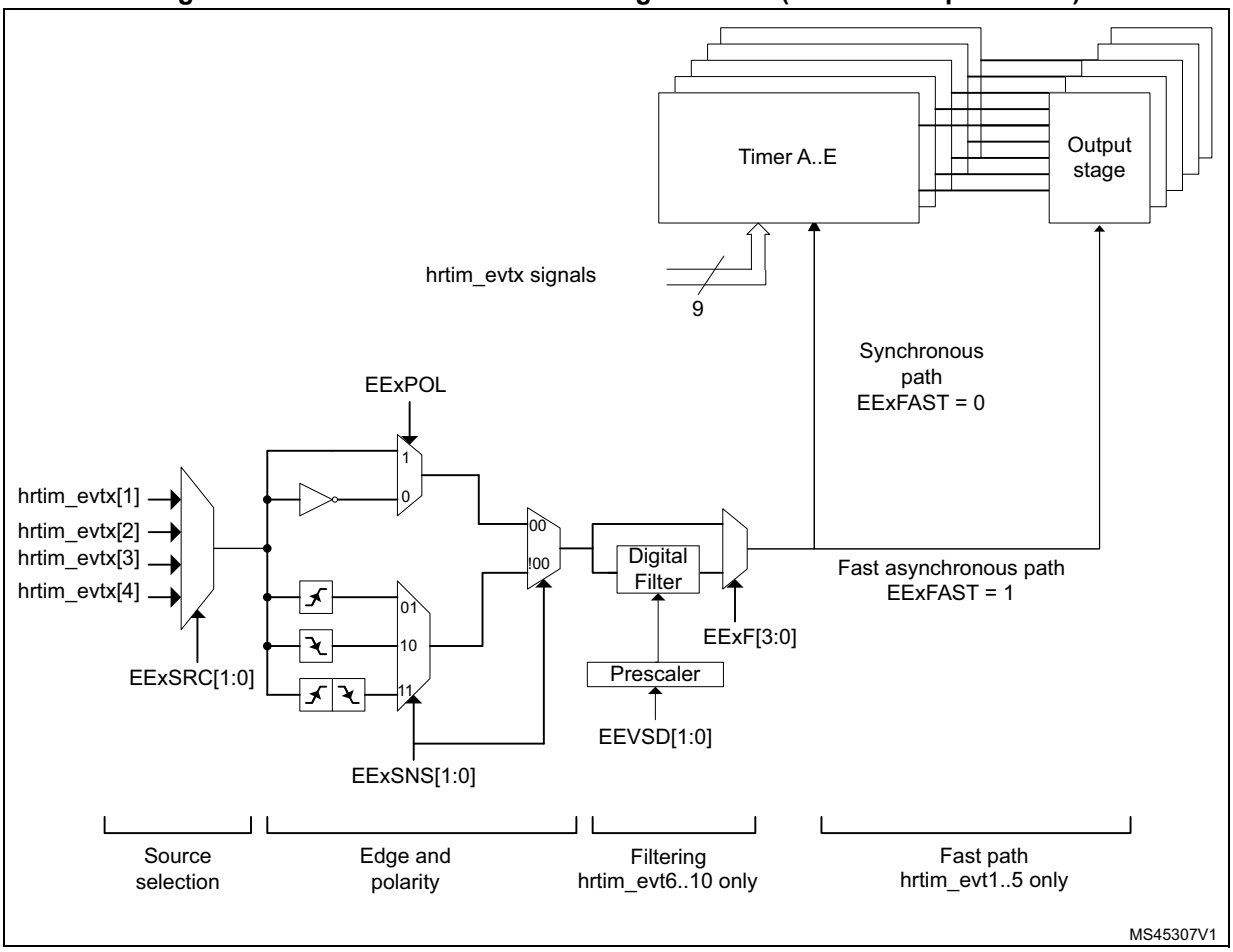
The HRTIM timer can handle events not generated within the timer, referred to as “external event”. These external events come from multiple sources, either on-chip or off-chip:

- built-in comparators,
- digital input pins (typically connected to off-chip comparators and zero-crossing detectors),
- on-chip events for other peripheral (ADC’s analog watchdogs and general purpose timer trigger outputs).

The external events conditioning circuitry allows to select the signal source for a given channel (with a 4:1 multiplexer) and to convert it into an information that can be processed by the crossbar unit (for instance, to have an output reset triggered by a falling edge detection on an external event channel).

Up to 10 external event channels can be conditioned and are available simultaneously for any of the 5 timers. This conditioning is common to all timers, since this is usually dictated by external components (such as a zero-crossing detector) and environmental conditions (typically the filter set-up will be related to the applications noise level and signature). *Figure 300* presents an overview of the conditioning logic for a single channel.

Figure 300. External event conditioning overview (1 channel represented)



The 10 external events are initialized using the HRTIM_EECR1 and HRTIM EECR2 registers:

- to select up to 4 sources with the EExSRC[1:0] bits,
- to select the sensitivity with EExSNS[1:0] bits, to be either level-sensitive or edge-sensitive (rising, falling or both),
- to select the polarity, in case of a level sensitivity, with EExPOL bit,
- to have a low latency mode, with EExFAST bits (see *Latency to external events*), for external events 1 to 5.

Note: The external events used as triggers for reset, capture, burst mode, ADC triggers and delayed protection are edge-sensitive even if EESNS bit is reset (level-sensitive selection): if POL = 0 the trigger is active on external event rising edge, while if POL = 1 the trigger is active on external event falling edge.

The external events are discarded as long as the counters are disabled (TxCEN bit reset) to prevent any output state change and counter reset, except if they are used as ADC triggers.

Additionally, it is possible to enable digital noise filters, for external events 6 to 10, using EExF[3:0] bits in the HRTIM_EECR3 register.

A digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has

reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected).

The sampling clock is either the f_{HRTIM} clock or a specific prescaled clock f_{EEVS} derived from f_{HRTIM}, defined with EEVSD[1:0] bits in HRTIM_EECR3 register.

Table 298 summarizes the available sources and features associated with each of the 10 external events channels.

Table 298. External events mapping and associated features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E	Src1	Src 2	Src3	Src4
1	Yes	-	-	-	PC10	COMP1	TIM1_TRGO	ADC1_AWD1
2	Yes	-	-	-	PC12	COMP2	TIM2_TRGO	ADC1_AWD2
3	Yes	-	-	-	PD5	-	TIM3_TRGO	ADC1_AWD3
4	Yes	-	-	-	PG11	OPAMP1(1)	TIM7_TRGO	ADC2_AWD1
5	Yes	-	-	-	PG12	-	LPTIM1 OUT	ADC2_AWD2
6	-	Yes	Yes	-	PB4	COMP1	TIM6_TRGO	ADC2_AWD3
7	-	Yes	Yes	-	PB5	COMP2	TIM7_TRGO	-
8	-	Yes	-	Yes	PB6	-	TIM6_TRGO	TTCAN_TMP
9	-	Yes	-	Yes	PB7	OPAMP1(1)	TIM15_TRGO	TTCAN_RTP
10	-	Yes	-	-	PG13	-	LPTIM2 OUT	TTCAN_SOC

1. OPAMP1_VOUT can be used as High-resolution timer internal event source. In this case, OPAMP1_VOUT (PC4) pin must be configured in input mode. The data from the GPIO pin is redirect to the HRTIM external events through the pin Schmitt trigger. If OPAMP1 is disabled, PC4 pin, configured in input mode, can be used as HRTIM external events.

Latency to external events

The external event conditioning gives the possibility to adjust the external event processing time (and associated latency) depending on performance expectations:

- A regular operating mode, in which the external event is resampled with the clock before acting on the output crossbar. This adds some latency but gives access to all crossbar functionalities. It enables the generation of an externally triggered high-resolution pulse.
- A fast operating mode, in which the latency between the external event and the action on the output is minimized. This mode is convenient for ultra-fast over-current protections, for instance.

EExFAST bits in the HRTIM_EECR1 register allow to define the operating for channels 1 to 5. This influences the latency and the jitter present on the output pulses, as summarized in the table below.

Table 299. Output set/reset latency and jitter vs external event operating mode

EExFAST	Response time latency	Response time jitter	Jitter on output pulse (counter reset by ext. event)
0	5 to 6 cycles of f_{HRTIM} clock	1 cycles of f_{HRTIM} clock	No jitter, pulse width maintained with high-resolution
1	Minimal latency (depends whether the comparator or digital input is used)	Minimal jitter	1 cycle of f_{HRTIM} clock jitter pulse width resolution down to t_{HRTIM}

The EExFAST mode is only available with level-sensitive programming (EExSNS[1:0] = 00); the edge-sensitivity cannot be programmed.

It is possible to apply event filtering to external events (both blanking and windowing with EExFLTR[3:0] != 0000, see *Section 36.3.8*). In this case, EExLTCHx bit must be reset: the postponed mode is not supported, neither the windowing timeout feature.

Note: The external event configuration (source and polarity) must not be modified once the related EExFAST bit is set.

A fast external event cannot be used to toggle an output: if must be enabled either in HRTIM_SETxyR or HRTIM_RSTxyR registers, not in both.

When a set and a reset event - from 2 independent fast external events - occur simultaneously, the reset has the highest priority in the crossbar and the output becomes inactive.

When EExFAST bit is set, the output cannot be changed during the 11 f_{HRTIM} clock periods following the external event.

Figure 301 and *Figure 302* give practical examples of the reaction time to external events, for output set/reset and counter reset.

Figure 301. Latency to external events falling edge (counter reset and output set)

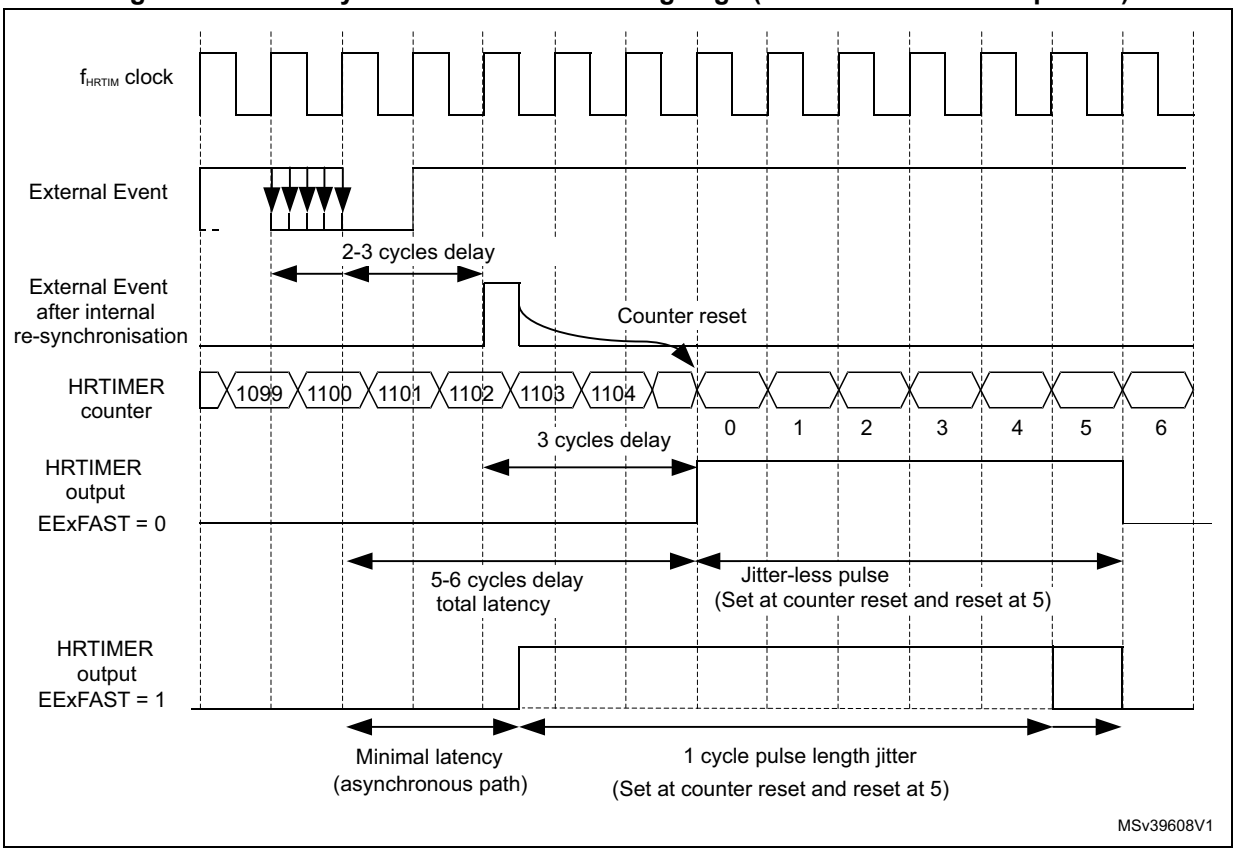
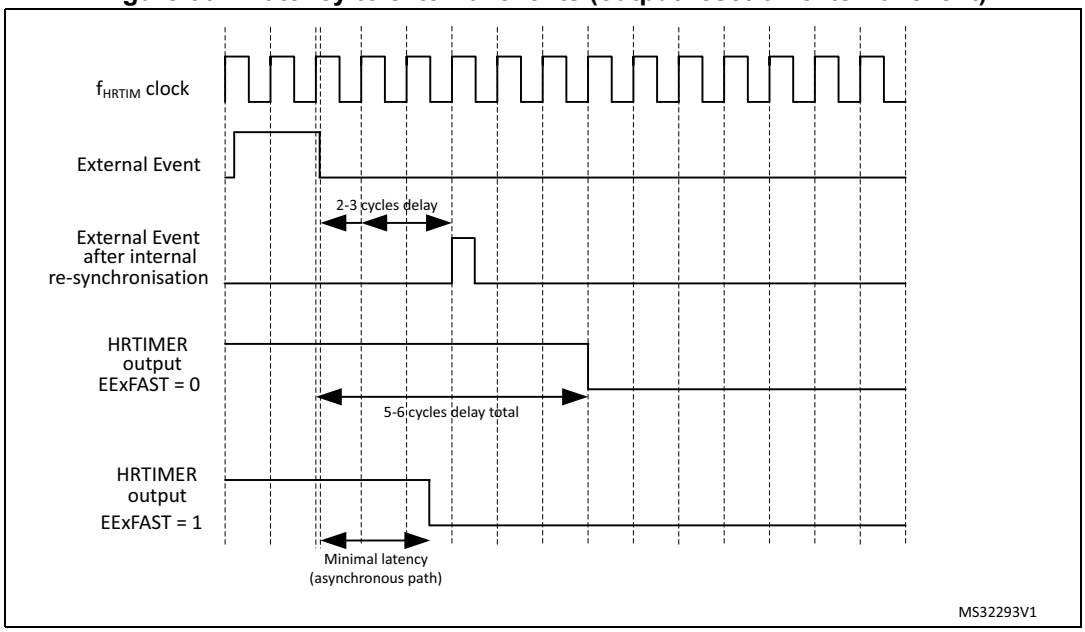


Figure 302. Latency to external events (output reset on external event)



36.3.8 External event filtering in timing units

Once conditioned, the 10 external events are available for all timing units.

They can be used directly and are active as soon as the timing unit counter is enabled (TxCEN bit set).

They can also be filtered to have an action limited in time, usually related to the counting period. Two operations can be performed:

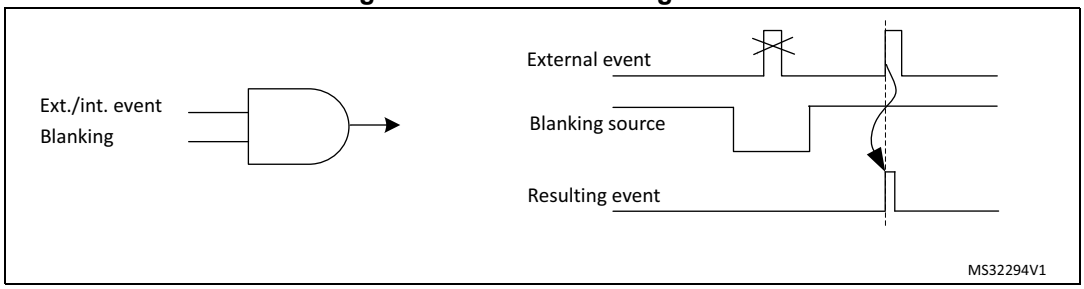
- blanking, to mask external events during a defined time period,
- windowing, to enable external events only during a defined time period.

These modes are enabled using HRTIM_EExFLTR[3:0] bits in the HRTIM_EEFxR1 and HRTIM_EEFxR2 registers. Each of the 5 TimerA..E timing units has its own programmable filter settings for the 10 external events.

Blanking mode

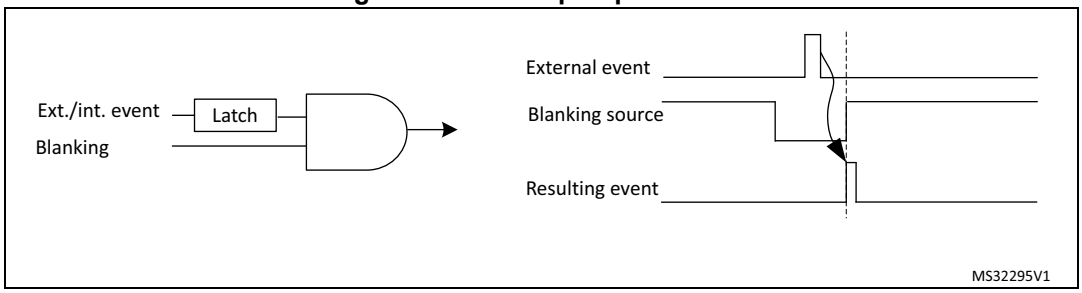
In event blanking mode (see *Figure 303*), the external event is ignored if it happens during a given blanking period. This is convenient, for instance, to avoid a current limit to trip on switching noise at the beginning of a PWM period. This mode is active for EExFLTR[3:0] bitfield values ranging from 0001 to 1100.

Figure 303. Event blanking mode



In event postpone mode, the external event is not taken into account immediately but is memorized (latched) and generated as soon as the blanking period is completed, as shown on *Figure 304*. This mode is enabled by setting EExLTCH bit in HRTIM_EEFxR1 and HRTIM_EEFxR2 registers.

Figure 304. Event postpone mode



The blanking signal comes from several sources:

- the timer itself: the blanking lasts from the counter reset to the compare match (EExFLTR[3:0] = 0001 to 0100 for Compare 1 to Compare 4)
- from other timing units (EExFLTR[3:0] = 0101 to 1100): the blanking lasts from the selected timing unit counter reset to one of its compare match, or can be fully programmed as a waveform on Tx2 output. In this case, events are masked as long as the Tx2 signal is inactive (it is not necessary to have the output enabled, the signal is taken prior to the output stage).

The EEXFLTR[3:0] configurations from 0101 to 1100 are referred to as TIMFLTR1 to TIMFLTR8 in the bit description, and differ from one timing unit to the other. *Table 300* gives the 8 available options per timer: CMPx refers to blanking from counter reset to compare match, Tx2 refers to the timing unit TIMx output 2 waveform defined with HRTIM_SETx2 and HRTIM_RSTx2 registers. For instance, Timer B (TIMFLTR6) is Timer C output 2 waveform.

Table 300. Filtering signals mapping per time

	Source	Timer A				Timer B				Timer C				Timer D				Timer E			
		CMP 1	CMP 2	CMP 4	TA2	CMP 1	CMP 2	CMP 4	TB2	CMP 1	CMP 2	CMP 4	TC2	CMP 1	CMP 2	CMP 4	TD2	CMP 1	CMP 2	CMP 4	TE2
Destination	Timer A	-	-	-	-	1	-	2	3	4	-	5	6	7	-	-	-	-	8	-	-
	Timer B	1	-	2	3	-	-	-	-	4	5	-	6	-	7	-	-	8	-	-	-
	Timer C	-	1	-	-	2	-	3	4	-	-	-	-	5	-	6	7	-	-	8	-
	Timer D	1	-	-	-	-	2	-	-	3	4	-	5	-	-	-	-	6	-	7	8
	Timer E	-	1	-	-	2	-	-	-	3	-	4	5	6	-	7	8	-	-	-	-

Figure 305 and *Figure 306* give an example of external event blanking for all edge and level sensitivities, in regular and postponed modes.

Figure 305. External trigger blanking with edge-sensitive trigger

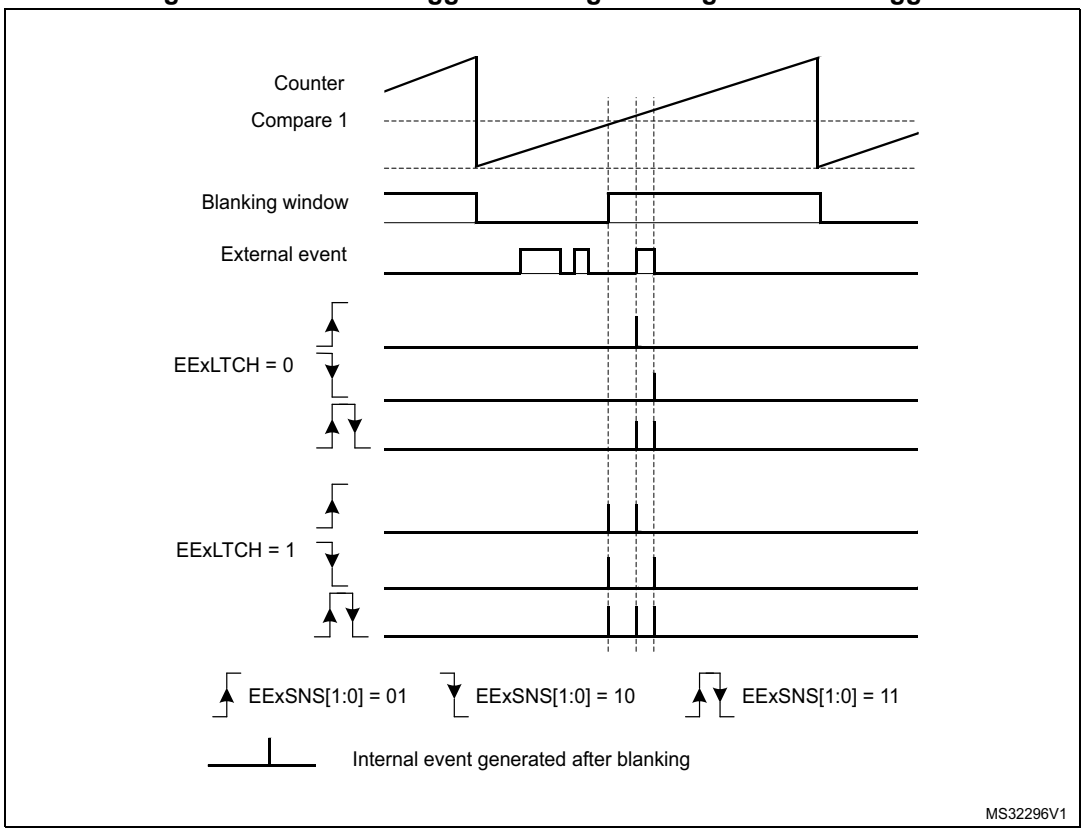
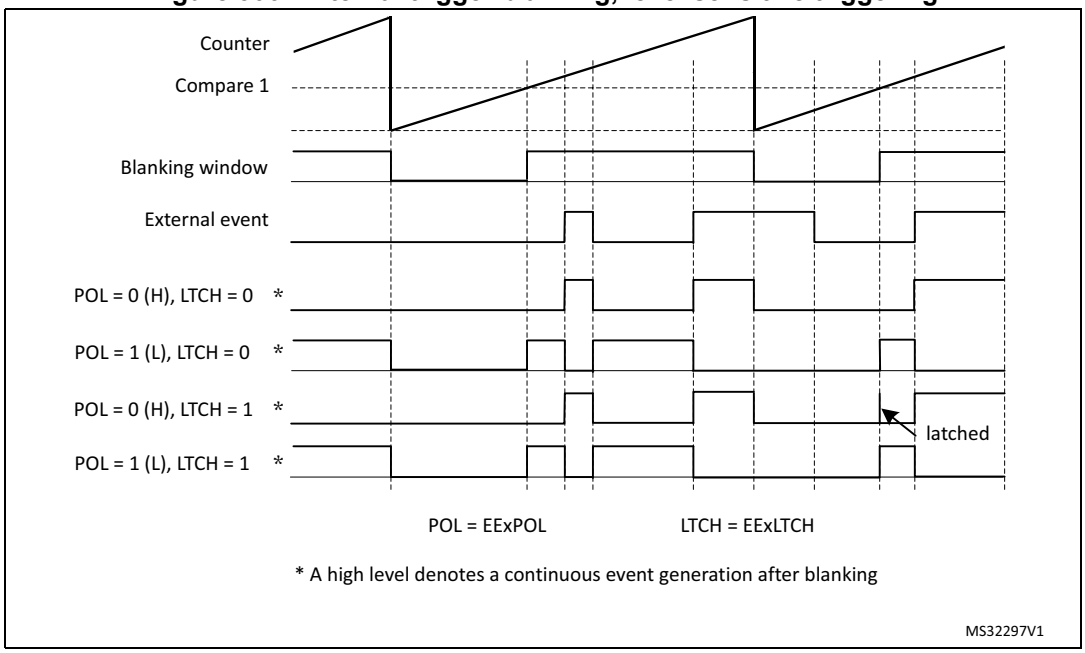


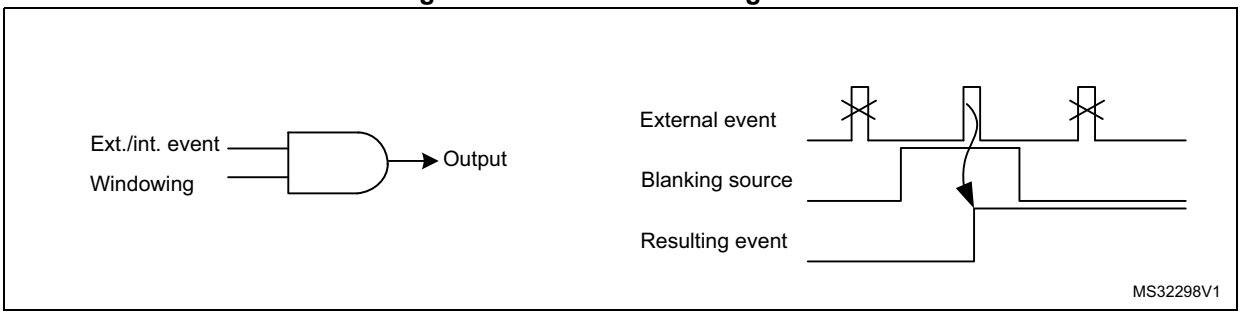
Figure 306. External trigger blanking, level sensitive triggering



Windowing mode

In event windowing mode, the event is taken into account only if it occurs within a given time window, otherwise it is ignored. This mode is active for EExFLTR[3:0] ranging from 1101 to 1111.

Figure 307. Event windowing mode



EExLTCH bit in EEFxR1 and EEFxR2 registers allows to latch the signal, if set to 1: in this case, an event is accepted if it occurs during the window but is delayed at the end of it.

- If EExLTCH bit is reset and the signal occurs during the window, it is passed through directly.
- If EExLTCH bit is reset and no signal occurs, a timeout event is generated at the end of the window.

A use case of the windowing mode is to filter synchronization signals. The timeout generation allows to force a default synchronization event, when the expected synchronization event is lacking (for instance during a converter start-up).

There are 3 sources for each external event windowing, coded as follows:

- 1101 and 1110: the windowing lasts from the counter reset to the compare match (respectively Compare 2 and Compare 3)
- 1111: the windowing is related to another timing unit and lasts from its counter reset to its Compare 2 match. The source is described as TIMWIN in the bit description and is given in *Table 301*. As an example, the external events in timer B can be filtered by a window starting from timer A counter reset to timer A Compare 2.

Table 301. Windowing signals mapping per timer (EEFLTR[3:0] = 1111)

Destination	Timer A	Timer B	Timer C	Timer D	Timer E
TIMWIN (source)	Timer B CMP2	Timer A CMP2	Timer D CMP2	Timer C CMP2	Timer D CMP2

Note: The timeout event generation is not supported if the external event is programmed in fast mode.

Figure 308 and *Figure 309* present how the events are generated for the various edge and level sensitivities, as well as depending on EExLTCH bit setting. Timeout events are specifically mentioned for clarity reasons.

Figure 308. External trigger windowing with edge-sensitive trigger

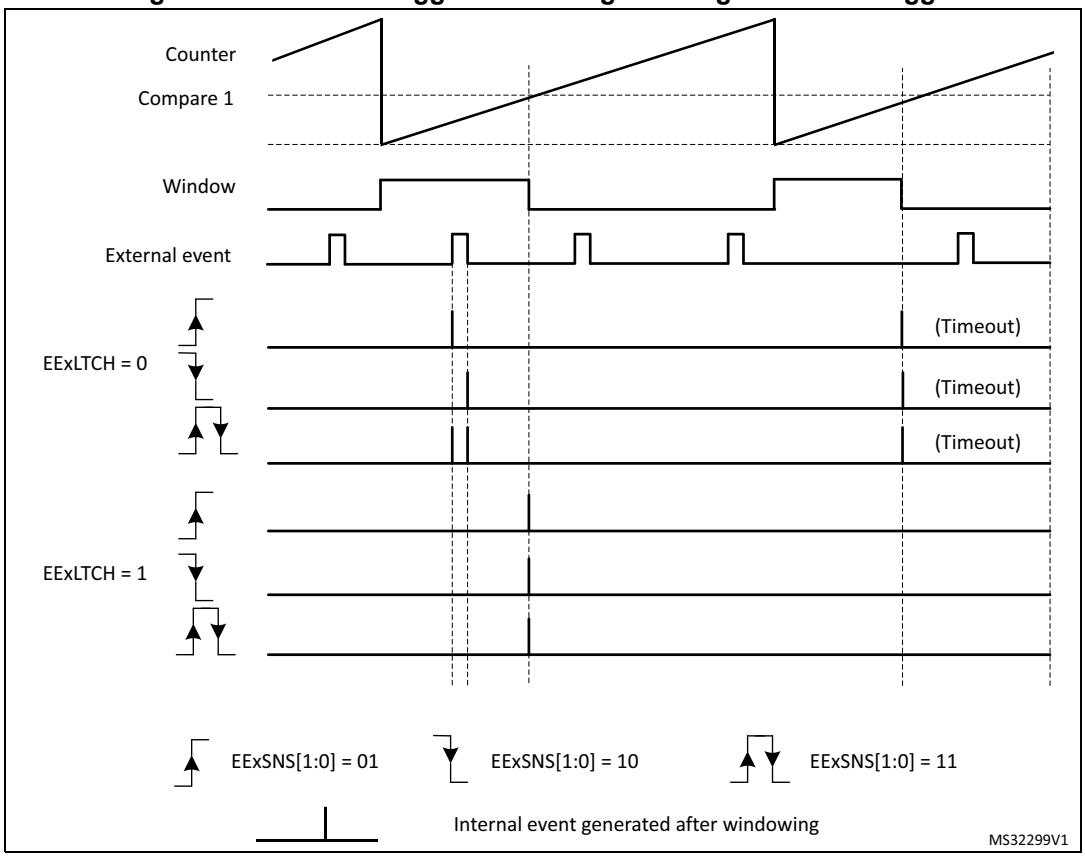
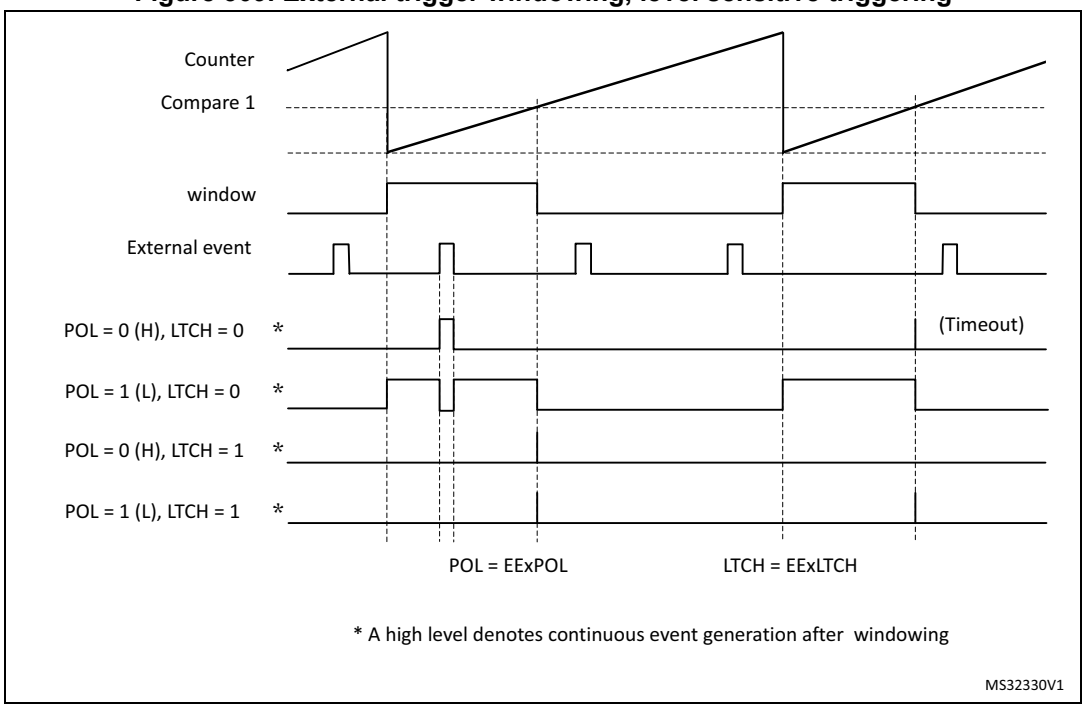


Figure 309. External trigger windowing, level sensitive triggering



36.3.9 Delayed Protection

The HRTIM features specific protection schemes, typically for resonant converters when it is necessary to shut down the PWM outputs in a delayed manner, either once the active pulse is completed or once a push-pull period is completed. These features are enabled with DLYPRTEN bit in the HRTIM_OUTxR register, and are using specific external event channels.

Delayed Idle

In this mode, the active pulse is completed before the protection is activated. The selected external event causes the output to enter in idle mode at the end of the active pulse (defined by an output reset event in HRTIM_RSTx1R or HRTIM_RSTx2R).

Once the protection is triggered, the idle mode is permanently maintained but the counter continues to run, until the output is re-enabled. Tx1OEN and Tx2OEN bits are not affected by the delayed idle entry. To exit from delayed idle and resume operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1. The output state will change on the first transition to an active state following the output enable command.

Note: The delayed idle mode cannot be exited immediately after having been entered, before the active pulse is completed: it is mandatory to make sure that the outputs are in idle state before resuming the run mode. This can be done by waiting up to the next period, for instance, or by polling the O1CPY and/or O2CPY status bits in the TIMxISR register.

The delayed idle mode can be applied to a single output (DLYPRT[2:0] = x00 or x01) or to both outputs (DLYPRT[2:0] = x10).

An interrupt or a DMA request can be generated in response to a Delayed Idle mode entry. The DLYPRT flag in HRTIM_TIMxISR is set as soon as the external event arrives, independently from the end of the active pulse on output.

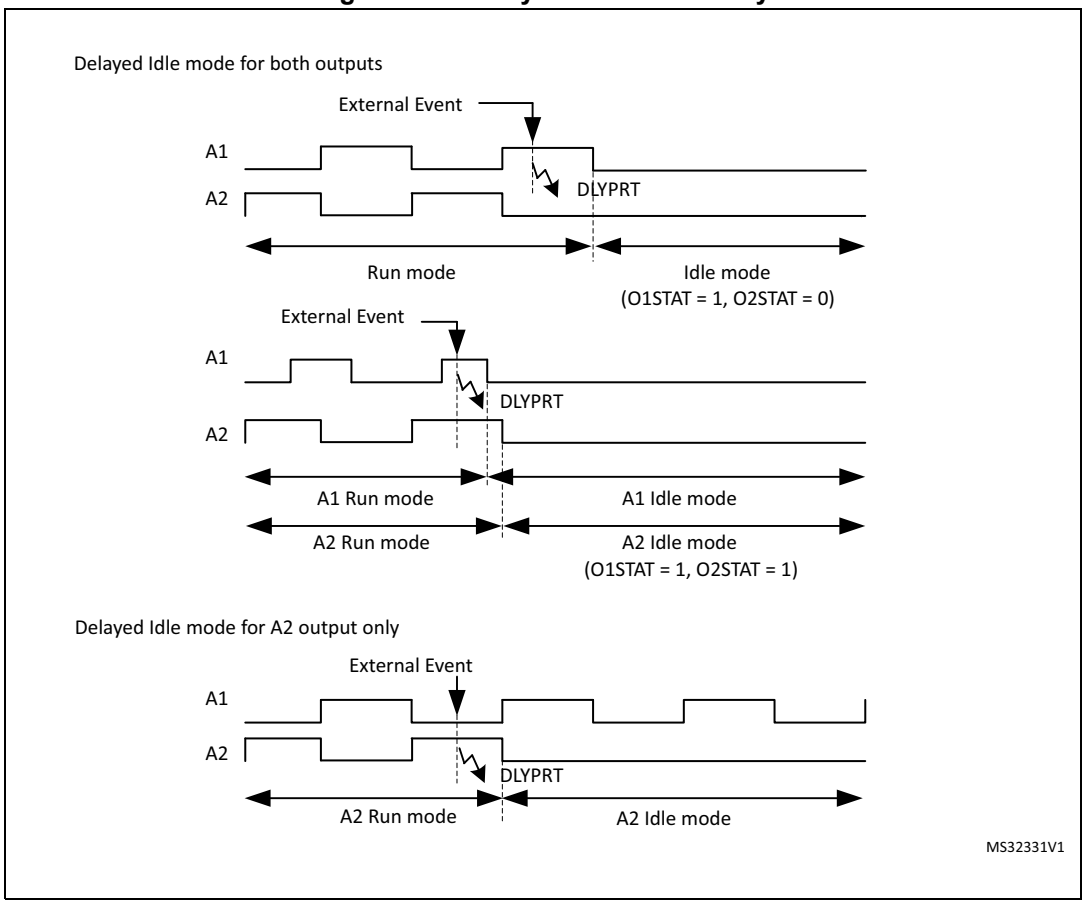
When the Delayed Idle mode is triggered, the output states can be determined using O1STAT and O2STAT in HRTIM_TIMxISR. Both status bits are updated even if the delayed idle is applied to a single output. When the push-pull mode is enabled, the IPPSTAT flag in HRTIM_TIMxISR indicates during which period the delayed protection request occurred.

This mode is available whatever the timer operating mode (regular, push-pull, deadtime). It is available with 2 external events only:

- hrtim_evt6 and hrtim_evt7 for Timer A, B and C
- hrtim_evt8 and hrtim_evt9 for Timer D and E

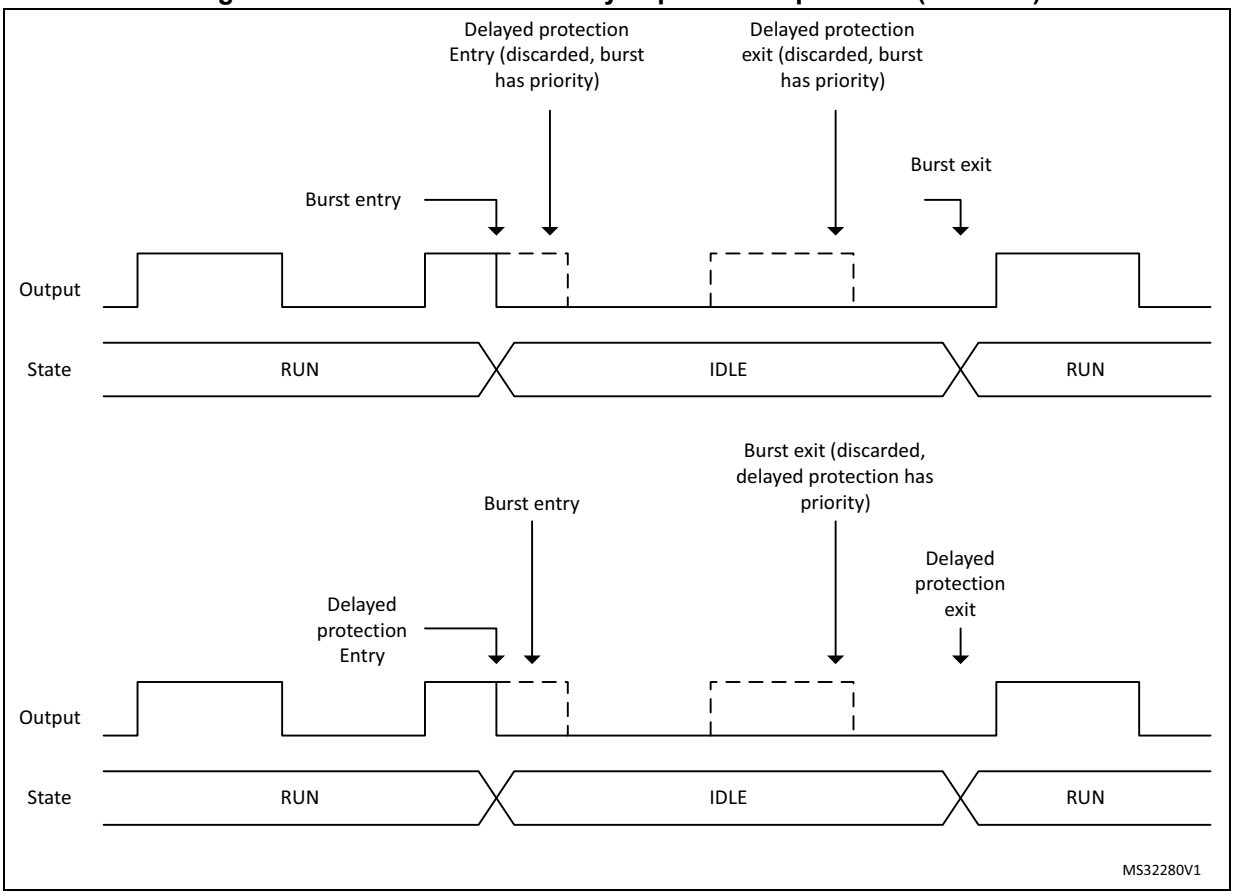
The delayed protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxEN bit is reset, until the TxyOEN bits are set.

Figure 310. Delayed Idle mode entry



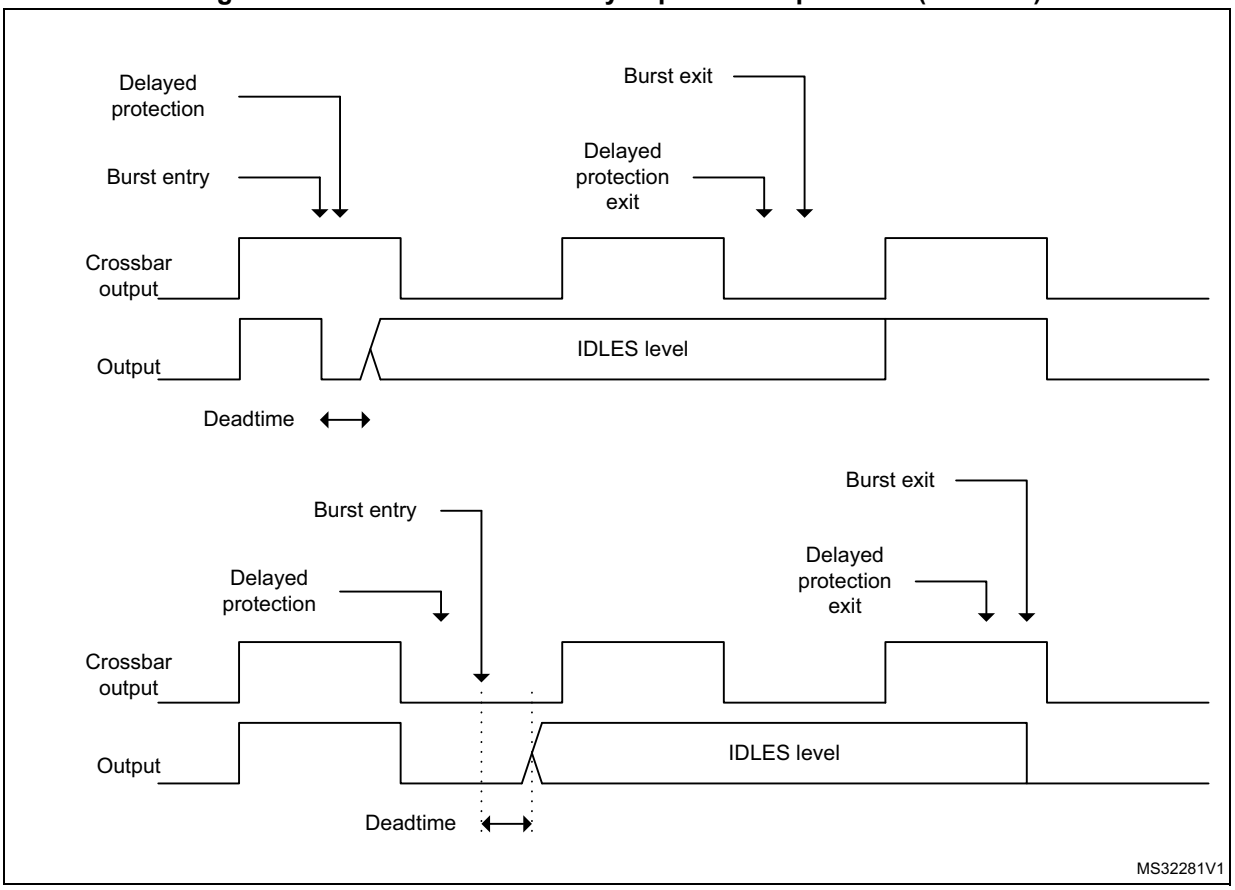
The delayed idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the delayed idle protection has been triggered. On the contrary, If the delayed protection is exited while the burst mode is active, the burst mode will be resumed normally and the output will be maintained in the idle state until the burst mode exits. *Figure 311* gives an overview of these different scenarios.

Figure 311. Burst mode and delayed protection priorities (DIDL = 0)



The same priorities are applied when the delayed burst mode entry is enabled (DIDL bit set), as shown on *Figure 312* below.

Figure 312. Burst mode and delayed protection priorities (DIDL = 1)



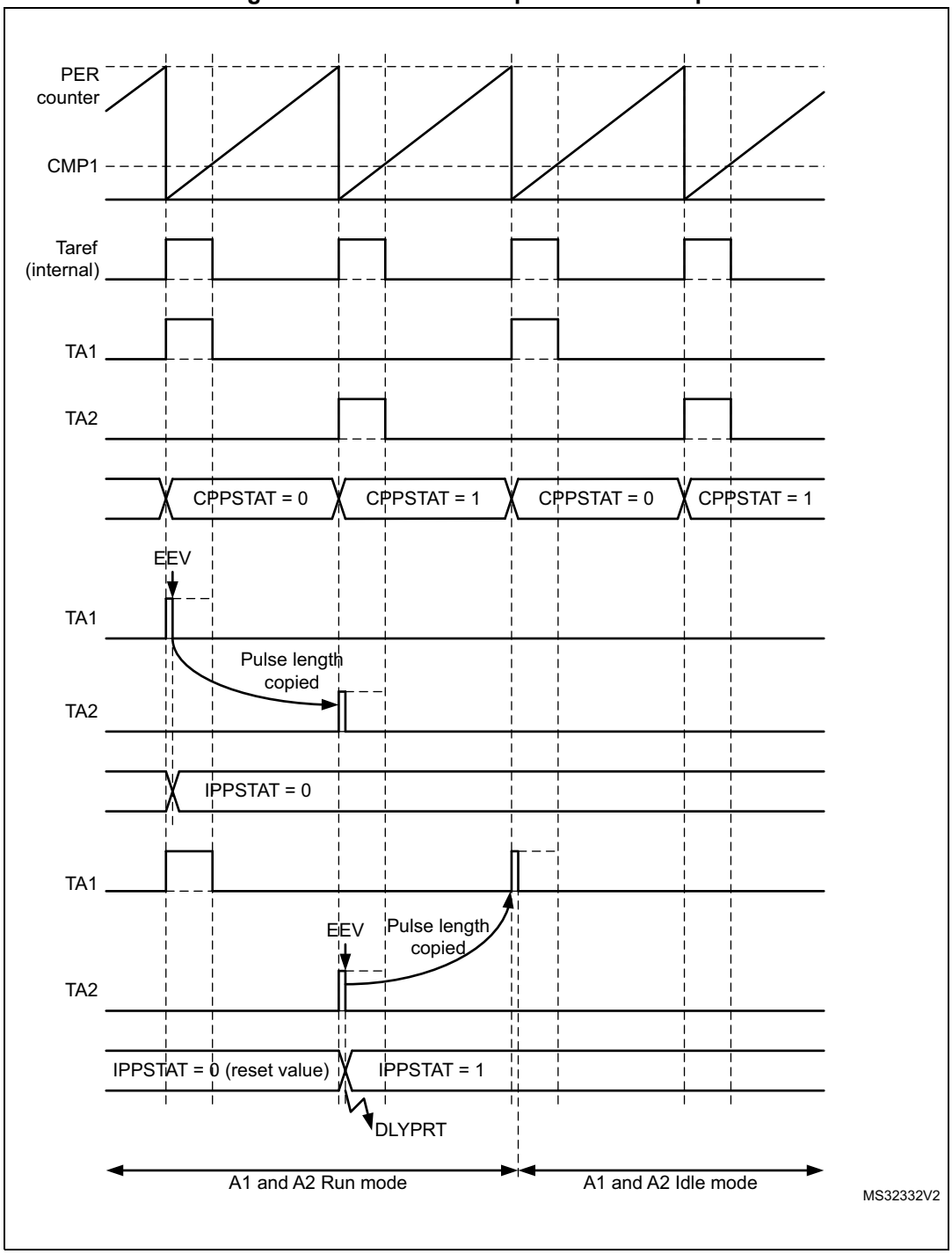
Balanced Idle

Only available in push-pull mode, it allows to have balanced pulsewidth on the two outputs when one of the active pulse is shortened due to a protection. The pulsewidth, which was terminated earlier than programmed, is copied on the alternate output and the two outputs are then put in idle state, until the normal operation is resumed by software. This mode is enabled by writing x11 in DLYPRT[2:0] bitfield in HRTIM_OUTxR.

This mode is available with 2 external events only:

- hrtim_evt6 and hrtim_evt7 for Timer A, B and C
- hrtim_evt8 and hrtim_evt9 for Timer D and E

Figure 313. Balanced Idle protection example



When the balanced Idle mode is enabled, the selected external event triggers a capture of the counter value into the Compare 4 active register (this value is not user-accessible). The push-pull is maintained for one additional period so that the shorten pulse can be repeated: a new output reset event is generated while the regular output set event is maintained.

The Idle mode is then entered and the output takes the level defined by IDLESx bits in the HRTIM_OUTxR register. The balanced idle mode entry is indicated by the DLYPRT flag, while the IPPSTAT flag indicates during which period the external event occurred, to determine the sequence of shorten pulses (HRTIM_CHA1 then HRTIM_CHA2 or vice versa).

The timer operation is not interrupted (the counter continues to run).

To enable the balanced idle mode, it is necessary to have the following initialization:

- timer operating in continuous mode (CONT = 1)
- Push-pull mode enabled
- HRTIM_CMP4xR must be set to 0 and the content transferred into the active register (for instance by forcing a software update)
- DELCMP4[1:0] bit field must be set to 00 (auto-delayed mode disabled)
- DLYPRT[2:0] = x11 (delayed protection enable)

Note: The HRTIM_CMP4xR register must not be written during a balanced idle operation. The CMP4 event is reserved and cannot be used for another purpose.

In balanced idle mode, it is recommended to avoid multiple external events or software-based reset events causing an output reset. If such an event arrives before a balanced idle request within the same period, it will cause the output pulses to be unbalanced (1st pulse length defined by the external event or software reset, while the 2nd pulse is defined by the balanced idle mode entry).

The minimum pulsewidth that can be handled in balanced idle mode is 4 f_{HRTIM} clock periods.

If the capture occurs before the counter has reached this minimum value, the current pulse is extended up to 4 f_{HRTIM} clock periods before being copied into the secondary output. In any case, the pulsewidths are always balanced.

Tx1OEN and Tx2OEN bits are not affected by the balanced idle entry. To exit from balanced idle and resume the operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1 simultaneously. The output state will change on the first active transition following the output enable.

It is possible to resume operation similarly to the delayed idle entry. For instance, if the external event arrives while output 1 is active (delayed idle effective after output 2 pulse), the re-start sequence can be initiated for output 1 first. To do so, it is necessary to poll CPPSTAT bit in the HRTIM_TIMxISR register. Using the above example (IPPSTAT flag equal to 0), the operation will be resumed when CPPSTAT bit is 0.

In order to have a specific re-start sequence, it is possible to poll the CPPSTAT to know which output will be active first. This allows, for instance, to re-start with the same sequence as the idle entry sequence: if EEV arrives during output 1 active, the re-start sequence will be initiated when the output 1 is active (CPPSTAT = 0).

Note: The balanced idle mode must not be disabled while a pulse balancing sequence is on-going. It is necessary to wait until the CMP4 flag is set, thus indicating that the sequence is completed, to reset the DLYPRTEN bit.

The balanced idle protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxCEN bit is reset, until TxyOEN bits are set.

Balanced idle can be used together with the burst mode under the following conditions:

- TxBM bit must be reset (counter clock maintained during the burst, see *Section 36.3.13*),
- No balanced idle protection must be triggered while the outputs are in a burst idle state.

The balanced idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the balanced idle protection has been triggered. On the contrary, if the delayed protection is exited while the burst mode is active, the burst mode will be resumed normally.

Note: Although the output state is frozen in idle mode, a number of events are still generated on the auxiliary outputs (see Section 36.3.16) during the idle period following the delayed protection:
- Output set/reset interrupt or DMA requests
- External event filtering based on output signal
- Capture events triggered by set/reset

36.3.10 Register preload and update management

Most of HRTIM registers are buffered and can be preloaded if needed. Typically, this allows to prevent the waveforms from being altered by a register update not synchronized with the active events (set/reset).

When the preload mode is enabled, accessed registers are shadow registers. Their content is transferred into the active register after an update request, either software or synchronized with an event.

By default, PREEN bits in HRTIM_MCR and HRTIM_TIMxCR registers are reset and the registers are not preloaded: any write directly updates the active registers. If PREEN bit is reset while the timer is running and preload was enabled, the content of the preload registers is directly transferred into the active registers.

Each timing unit and the master timer have their own PREEN bit. If PRREN is set, the preload registers are enabled and transferred to the active register only upon an update event.

There are two options to initialize the timer when the preload feature is needed:

- Enable PREEN bit at the very end of the timer initialization to have the preload registers transferred into the active registers before the timer is enabled (by setting MCEN and TxCEN bits).
- enable PREEN bit at any time during the initialization and force a software update immediately before starting.

Table 302 lists the registers which can be preloaded, together with a summary of available update events.

Table 302. HRTIM preloadable control registers and associated update sources

Timer	Preloadable registers	Preload enable	Update sources
Master Timer	HRTIM_DIER HRTIM_MPER HRTIM_MREP HRTIM_MCMP1R HRTIM_MCMP2R HRTIM_MCMP3R HRTIM_MCMP4R	PREEN bit in HRTIM_MCR	Software Repetition event Burst DMA event Repetition event following a burst DMA event
Timer x x = A..E	HRTIM_TIMxDIER HRTIM_TIMxPER HRTIM_TIMxREP HRTIM_TIMxCMP1R HRTIM_TIMxCMP1CR HRTIM_TIMxCMP2R HRTIM_TIMxCMP3R HRTIM_TIMxCMP4R HRTIM_DTxR HRTIM_SETx1R HRTIM_RSTx1R HRTIM_SETx2R HRTIM_RSTx2R HRTIM_RSTxR	PREEN bit in HRTIM_TIMxCR	Software TIMx Repetition event TIMx Reset Event Burst DMA event Update event from other timers (TIMy, Master) Update event following a burst DMA event Update enable input 1..3 Update event following an update enable input 1..3
HRTIM Common	HRTIM_ADC1R HRTIM_ADC2R HRTIM_ADC3R HRTIM_ADC4R	TIMx or Master timer Update, depending on ADxUSRC[2:0] bits in HRTIM_CR1, if PREEN = 1 in the selected timer	

The master timer has 4 update options:

1. Software: writing 1 into MSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is cancelled.
2. Update done when the master counter rolls over and the master repetition counter is equal to 0. This is enabled when MREPU bit is set in HRTIM_MCR.
3. Update done once Burst DMA is completed (see *Section 36.3.21* for details). This is enabled when BRSTDMA[1:0] = 01 in HRTIM_MCR. It is possible to have both MREPU=1 and BRSTDMA=01.
   Note: The update can take place immediately after the end of the burst sequence if SWU bit is set (i.e. forced update mode). If SWU bit is reset, the update will be done on the next update event following the end of the burst sequence.
4. Update done when the master counter rolls over following a Burst DMA completion. This is enabled when BRSTDMA[1:0] = 10 in HRTIM_MCR.

An interrupt or a DMA request can be generated by the master update event.

Each timer (TIMA..E) can also have the update done as follows:

- By software: writing 1 into TxSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is canceled.
- Update done when the counter rolls over and the repetition counter is equal to 0. This is enabled when TxREPU bit is set in HRTIM_TIMxCR.
- Update done when the counter is reset or rolls over in continuous mode. This is enabled when TxRSTU bit is set in HRTIM_TIMxCR. This is used for a timer operating in single-shot mode, for instance.
- Update done once a Burst DMA is completed. This is enabled when UPDGAT[3:0] = 0001 in HRTIM_TIMxCR.
- Update done on the update event following a Burst DMA completion (the event can be enabled with TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0010 in HRTIM_TIMxCR.
- Update done when receiving a request on the update enable input 1..3. This is enabled when UPDGAT[3:0] = 0011, 0100, 0101 in HRTIM_TIMxCR.
- Update done on the update event following a request on the update enable input 1..3 (the event can be enabled with TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0110, 0111, 1000 in HRTIM_TIMxCR
- Update done synchronously with any other timer or master update (for instance TIMA can be updated simultaneously with TIMB). This is used for converters requiring several timers, and is enabled by setting bits MSTU and TxU in HRTIM_TIMxCR register.

The update enable inputs 1..3 allow to have an update event synchronized with on-chip events coming from the general-purpose timers. These inputs are rising-edge sensitive.

Table 303 lists the connections between update enable inputs and the on-chip sources.

Table 303. Update enable inputs and sources

Update enable input	Update source
Update enable input 1	TIM16_OC
Update enable input 2	TIM17_OC
Update enable input 3	TIM6_TRGO

This allows to synchronize low frequency update requests with high-frequency signals (for instance an update on the counter roll-over of a 100 kHz PWM that has to be done at a 100 Hz rate).

Note: The update events are synchronized to the prescaler clock when CKPSC[2:0] > 5.

An interrupt or a DMA request can be generated by the Timx update event.

MUDIS and TxUDIS bits in the HRTIM_CR1 register allow to temporarily disable the transfer from preload to active registers, whatever the selected update event. This allows to modify several registers in multiple timers. The regular update event takes place once these bits are reset.

MUDIS and TxUDIS bits are all grouped in the same register. This allows the update of multiple timers (not necessarily synchronized) to be disabled and resumed simultaneously.

The following example is a practical use case. A first power converter is controlled with the master, TIMB and TIMC. TIMB and TIMC must be updated simultaneously with the master timer repetition event. A second converter works in parallel with TIMA, TIMD and TIME, and TIMD, TIME must be updated with TIMA repetition event.

First converter

In HRTIM_MCR, MREPU bit is set: the update will occur at the end of the master timer counter repetition period. In HRTIM_TIMBCR and HRTIM_TIMCCR, MSTU bits are set to have TIMB and TIMC timers updated simultaneously with the master timer.

When the power converter set-point has to be adjusted by software, MUDIS, TBUDIS and TCUDIS bits of the HRTIM_CR register must be set prior to write accessing registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, MUDIS, TBUDIS and TCUDIS bits must be reset. The transfer from preload to active registers will be done as soon as the master repetition event occurs.

Second converter

In HRTIM_TIMACR, TAREPU bit is set: the update will occur at the end of the Timer A counter repetition period. In HRTIM_TIMDCR and HRTIM_TIMECR, TAU bits are set to have TIMD and TIME timers updated simultaneously with Timer A.

When the power converter set-point has to be adjusted by software, TAUDIS, TDUDIS and TEUDIS bits of the HRTIM_CR register must be set prior to write accessing the registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, TAUDIS, TDUDIS and TEUDIS bits can be reset: the transfer from preload to active registers will be done as soon as the Timer A repetition event occurs.

36.3.11 Events propagation within or across multiple timers

The HRTIM offers many possibilities for cascading events or sharing them across multiple timing units, including the master timer, to get full benefits from its modular architecture. These are key features for converters requiring multiple synchronized outputs.

This section summarizes the various options and specifies whether and how an event is propagated within the HRTIM.

TIMx update triggered by the Master timer update

The sources listed in *Table 304* are generating a master timer update. The table indicates if the source event can be used to trigger a simultaneous update in any of TIMx timing units.

Operating condition: MSTU bit is set in HRTIM_TIMxCR register.

Table 304. Master timer update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	BRSTDMA[1:0] = 01	No	Must be done in TIMxCR (UPDGAT[3:0] = 0001)
Roll-over event following a Burst DMA end	BRSTDMA[1:0] = 10	Yes	-
Repetition event caused by a counter roll-over	MREPU = 1	Yes	-
Repetition event caused by a counter reset (from HRTIM_SCIN or software)		No	-
Software update	MSWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx update triggered by the TIMy update

The sources listed in *Table 305* are generating a TIMy update. The table indicates if the given event can be used to trigger a simultaneous update in another or multiple TIMx timers.

Operating condition: TyU bit set in HRTIM_TIMxCR register (source = TIMy and destination = TIMx).

Table 305. TIMx update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0001)
Update caused by the update enable input	UPDGAT[3:0] = 0011, 0100, 0101	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0011, 0100, 0101
Master update	MSTU = 1 in HRTIM_TIMyCR	No	Must be done with MSTU = 1 in HRTIM_TIMxCR
Another TIMx update (TIMz>TIMy>TIMx)	TzU=1 in HRTIM_TIMyCR TyU=1 in TIMxCR	No	Must be done with TzU=1 in HRTIM_TIMxCR TzU=1 in HRTIM_TIMyCR
Repetition event caused by a counter roll-over	TyREPU = 1	Yes	-
Repetition event caused by a counter reset	TyREPU = 1	-	Refer to counter reset cases below
Counter roll-over	TyRSTU = 1	Yes	-
Counter software reset	TyRST=1 in HRTIM_CR2	No	Can be done simultaneously with update in HRTIM_CR2 register
Counter reset caused by a TIMz compare	TIMzCMPn in HRTIM_RSTyR	No	Must be done using TIMzCMPn in HRTIM_RSTxR
Counter reset caused by external events	EXTEVNTn in HRTIM_RSTyR	Yes	-

Table 305. TIMx update event propagation (continued)

Source	Condition	Propagation	Comment
Counter reset caused by a master compare or a master period	MSTCMPn or MSTPER in HRTIM_RSTyR	No	-
Counter reset caused by a TIMy compare	CMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by an update	UPDT in HRTIM_RSTyR	No	Propagation would result in a lock-up situation (update causing reset causing update)
Counter reset caused by HRTIM_SCIN	SYNCRSTy in HRTIM_TIMyCR	No	-
Software update	TySWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx Counter reset causing a TIMx update

Table 306 lists the counter reset sources and indicates whether they can be used to generate an update.

Operating condition: TxRSTU bit in HRTIM_TIMxCR register.

Table 306. Reset events able to generate an update

Source	Condition	Propagation	Comment
Counter roll-over		Yes	
Update event	UPDT in HRTIM_RSTxR	No	Propagation would result in a lock-up situation (update causing a reset causing an update)
External Event	EXTEVNTn in HRTIM_RSTxR	Yes	-
TIMy compare	TIMyCMPn in HRTIM_RSTxR	Yes	-
Master compare	MSTCMPn in HRTIM_RSTxR	Yes	-
Master period	MSTPER in HRTIM_RSTxR	Yes	-
Compare 2 and 4	CMPn in HRTIM_RSTxR	Yes	-
Software	TxRST=1 in HRTIM_CR2	Yes	-
HRTIM_SCIN	SYNCRSTx in HRTIM_TIMxCR	Yes	-

TIMx update causing a TIMx counter reset

Table 307 lists the update event sources and indicates whether they can be used to generate a counter reset.

Operating condition: UPDT bit set in HRTIM_RSTxR.

Table 307. Update event propagation for a timer reset

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	Yes	-
Update caused by the update enable input	UPDGAT[3:0] = 0011, 0100, 0101	Yes	-
Master update caused by a roll-over after a Burst DMA	MSTU = 1 in HRTIM_TIMxCR BRSTDMA[1:0] = 10 in HRTIM_MCR	Yes	-
Master update caused by a repetition event following a roll-over	MSTU = 1 in HRTIM_TIMxCR MREPU = 1 in HRTIM_MCR	Yes	-
Master update caused by a repetition event following a counter reset (software or due to HRTIM_SCIN)		No	-
Software triggered master timer update	MSTU = 1 in HRTIM_TIMxCR MSWU = 1 in HRTIM_CR2	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update
TIMy update caused by a TIMy counter roll-over	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by a TIMy repetition event	TyU = 1 in HRTIM_TIMxCR TyREPU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by an external event or a TIMy compare (through a TIMy reset)	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR EXTEVNTn or CMP4/2 in HRTIM_RSTyCR	Yes	-
TIMy update caused by sources other than those listed above	TyU = 1 in HRTIM_TIMxCR	No	-

Table 307. Update event propagation for a timer reset (continued)

Source	Condition	Propagation	Comment
Repetition event following a roll-over	TxREPU = 1 in HRTIM_TIMxCR	Yes	-
Repetition event following a counter reset		No	-
Timer reset	TxRSTU = 1 in HRTIM_TIMxCR	No	Propagation would result in a lock-up situation (reset causing an update causing a reset)
Software	TxSWU in HRTIM_CR2	No	-

36.3.12 Output management

Each timing unit controls a pair of outputs. The outputs have three operating states:

- RUN: this is the main operating mode, where the output can take the active or inactive level as programmed in the crossbar unit.
- IDLE: this state is the default operating state after an HRTIM reset, when the outputs are disabled by software or during a burst mode operation (where outputs are temporary disabled during a normal operating mode; refer to *Section 36.3.13* for more details). It is either permanently active or inactive.
- FAULT: this is the safety state, entered in case of a shut-down request on FAULTx inputs. It can be permanently active, inactive or Hi-Z.

The output status is indicated by TxyOEN bit in HRTIM_OENR register and TxyODS bit in HRTIM_ODSR register, as in *Table 308*.

Table 308. Output state programming, x= A..E, y = 1 or 2

TxyOEN (control/status) (set by software, cleared by hardware)	TxyODS (status)	Output operating state
1	x	RUN
0	0	IDLE
0	1	FAULT

TxyOEN bit is both a control and a status bit: it must be set by software to have the output in RUN mode. It is cleared by hardware when the output goes back in IDLE or FAULT mode. When TxyOEN bit is cleared, TxyODS bit indicates whether the output is in the IDLE or FAULT state. A third bit in the HRTIM_ODISR register allows to disable the output by software.

Figure 314. Output management overview

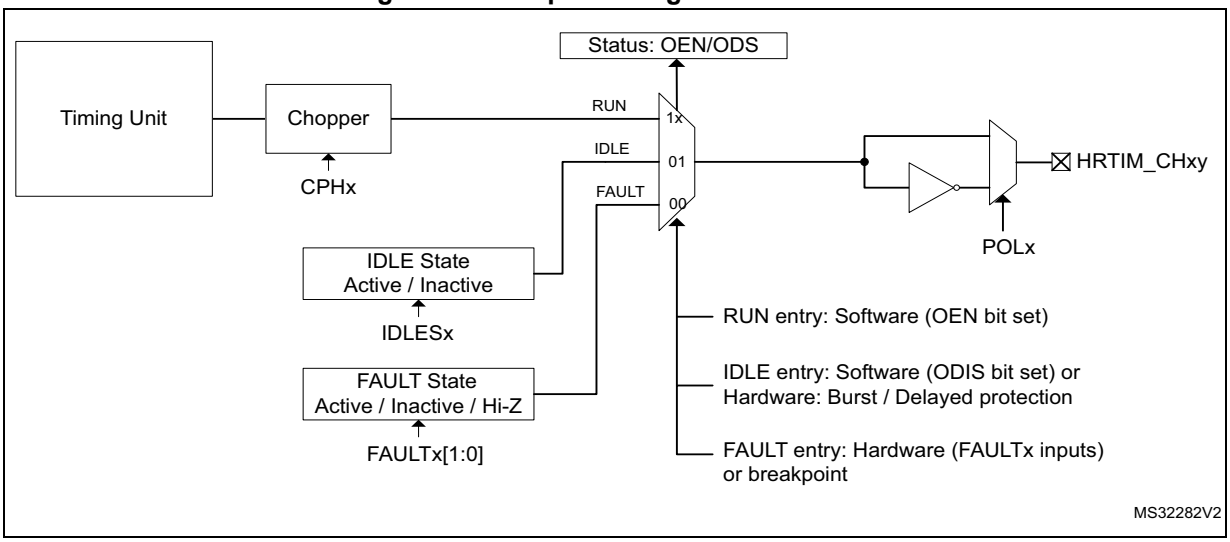
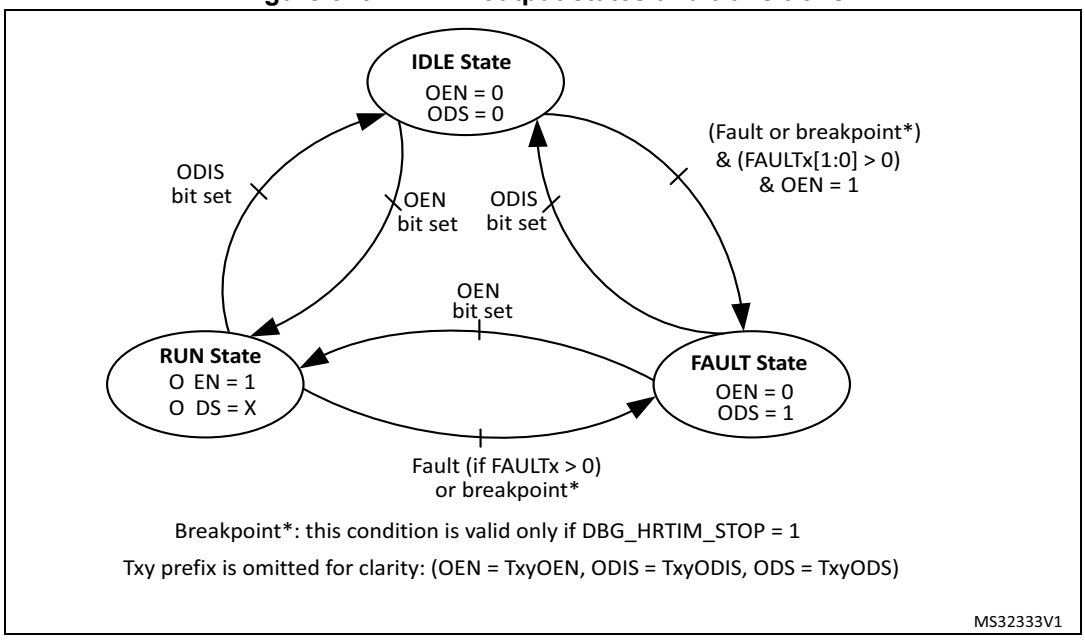


Figure 315 summarizes the bit values for the three states and how the transitions are triggered. Faults can be triggered by any external or internal fault source, as listed in *Section 36.3.15*, while the Idle state can be entered when the burst mode or delayed protections are active.

Figure 315. HRTIM output states and transitions



The FAULT and IDLE levels are defined as active or inactive. Active (or inactive) refers to the level on the timer output that causes a power switch to be closed (or opened for an inactive state).

The IDLE state has the highest priority: the transition FAULT → IDLE is possible even if the FAULT condition is still valid, triggered by ODIS bit set.

The FAULT state has priority over the RUN state: if TxyOEN bit is set simultaneously with a Fault event, the FAULT state will be entered. The condition is given on the transition IDLE →

FAULT, as in *Figure 315*: fault protection needs to be enabled (FAULTx[1:0] bits = 01, 10, 11) and the Txy OEN bit set with a fault active (or during a breakpoint if DBG_HRTIM_STOP = 1).

The output polarity is programmed using POLx bits in HRTIM_OUTxR. When POLx = 0, the polarity is positive (output active high), while it is active low in case of a negative polarity (POLx = 1). Practically, the polarity is defined depending on the power switch to be driven (PMOS vs. NMOS) or on a gate driver polarity.

The output level in the FAULT state is configured using FAULTx[1:0] bits in HRTIM_OUTxR, for each output, as follows:

- 00: output never enters the fault state and stays in RUN or IDLE state
- 01: output at active level when in FAULT
- 10: output at inactive level when in FAULT
- 11: output is tri-stated when in FAULT. The safe state must be forced externally with pull-up or pull-down resistors, for instance.

Note: FAULTx[1:0] bits must not be changed as long as the outputs are in FAULT state.

The level of the output in IDLE state is configured using IDLESx bit in HRTIM_OUTxR, as follows:

- 0: output at inactive level when in IDLE
- 1: output at active level when in IDLE

When TxyOEN bit is set to enter the RUN state, the output is immediately connected to the crossbar output. If the timer clock is stopped, the level will either be inactive (after an HRTIM reset) or correspond to the RUN level (when the timer was stopped and the output disabled).

During the HRTIM initialization, the output level can be prepositioned prior to have it in RUN mode, using the software forced output set and reset in the HRTIM_SETx1R and HRTIM_RSTx1R registers.

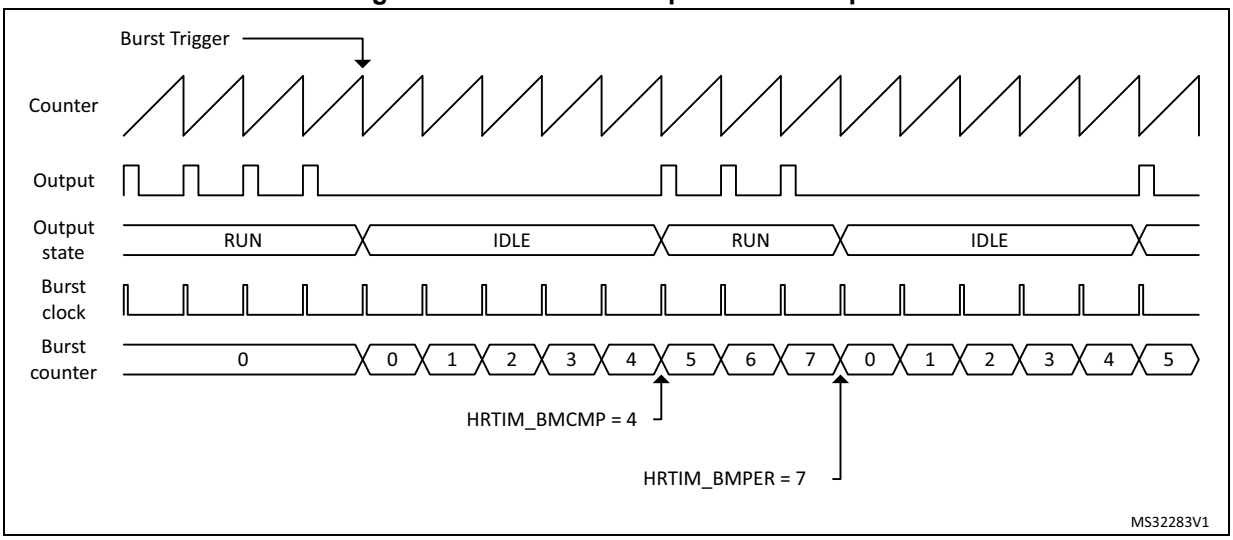
36.3.13 Burst mode controller

The burst mode controller allows to have the outputs alternatively in IDLE and RUN state, by hardware, so as to skip some switching periods with a programmable periodicity and duty cycle.

Burst mode operation is of common use in power converters when operating under light loads. It can significantly increase the efficiency of the converter by reducing the number of transitions on the outputs and the associated switching losses.

When operating in burst mode, one or a few pulses are outputs followed by an idle period equal to several counting periods, typically, where no output pulses are produced, as shown in the example on *Figure 316*.

Figure 316. Burst mode operation example



The burst mode controller consists of:

- A counter that can be clocked by various sources, either within or outside the HRTIM (typically the end of a PWM period).
- A compare register to define the number of idle periods: HRTIM_BMCMP.
- A period register to define the burst repetition rate (corresponding to the sum of the idle and run periods): HRTIM_BMPER.

The burst mode controller is able to take over the control of any of the 10 PWM outputs. The state of each output during a burst mode operation is programmed using IDLESx and IDLEMx bits in the HRTIM_OUTxR register, as in *Table 309*.

Table 309. Timer output programming for burst mode

IDLEMx	IDLESx	**Output state during burst mode**
0	X	No action: the output is not affected by the burst mode operation.
1	0	Output inactive during the burst
1	1	Output active during the burst

Note: IDLEMx bit must not be changed while the burst mode is active.

The burst mode controller only acts on the output stage. A number of events are still generated during the idle period:

- Output set/reset interrupt or DMA requests
- External event filtering based on Tx2 output signal
- Capture events triggered by output set/reset

During the burst mode, neither start not reset events are generated on the HRTIM_SCOUT output, even if TxBM bit is set.

Operating mode

It is necessary to have the counter enabled (TxCEN bit set) before using the burst mode on a given timing unit.The burst mode is enabled with BME bit in the HRTIM_BMCR register.

It can operate in continuous or single-shot mode, using BMOM bit in the HRTIM_BMCR register. The continuous mode is enabled when BMOM = 1. The Burst operation is maintained until BMSTAT bit in HRTIM_BMCR is reset to terminate it.

In single-shot mode (BMOM = 0), the idle sequence is executed once, following the burst mode trigger, and the normal timer operation is resumed immediately after.

The duration of the idle and run periods is defined with a burst mode counter and 2 registers. The HRTIM_BMCMPR register defines the number of counts during which the selected timer(s) are in an idle state (idle period). HRTIM_BMPER defines the overall burst mode period (sum of the idle and run periods). Once the initial burst mode trigger has occurred, the idle period length is HRTIM_BMCMPR+1, the overall burst period is HRTIM_BMPER+1.

Note: The burst mode period must not be less than or equal to the deadtime duration defined with DTRx[8:0] and DTFx[8:0] bitfields.

The counters of the timing units and the master timer can be stopped and reset during the burst mode operation. HRTIM_BMCR holds 6 control bits for this purpose: MTBM (master) and TABM..TEBM for Timer A..E.

When MTBM or TxBM bit is reset, the counter clock is maintained. This allows to keep a phase relationship with other timers in multiphase systems, for instance.

When MTBM or TxBM bit is set, the corresponding counter is stopped and maintained in reset state during the burst idle period. This allows to have the timer restarting a full period when exiting from idle. If SYNCSRC[1:0] = 00 or 10 (synchronization output on the master start or timer A start), a pulse is sent on the HRTIM_SCOUT output when exiting the burst mode.

Note: TxBM bit must not be set when the balanced idle mode is active (DLYPRT[1:0] = 0x11).

Burst mode clock

The burst mode controller counter can be clocked by several sources, selected with BMCLK[3:0] bits in the HRTIM_BMCR register:

- BMCLK[3:0] = 0000 to 0101: Master timer and TIMA..E reset/roll-over events. This allows to have burst mode idle and run periods aligned with the timing unit counting period (both in free-running and counter reset mode).
- BMCLK[3:0] = 0110 to 1001: The clocking is provided by the general purpose timers, as in *Table 310*. In this case, the burst mode idle and run periods are not necessarily aligned with timing unit counting period (a pulse on the output may be interrupted, resulting a waveform with modified duty cycle for instance.
- BMCLK[3:0] = 1010: The f_{HRTIM} clock prescaled by a factor defined with BMPRSC[3:0] bits in HRTIM_BMCR register. In this case, the burst mode idle and run periods are not necessarily aligned with the timing unit counting period (a pulse on the output may be interrupted, resulting in a waveform with a modified duty cycle, for instance.

Table 310. Burst mode clock sources from general purpose timer

BMCLK[3:0]	Clock source
0110	hrtim_bm_ck1: TIM16 OC
0111	hrtim_bm_ck2: TIM17 OC
1000	hrtim_bm_ck3: TIM7 TRGO
1001	hrtim_bm_ck4: Reserved

The pulsewidth on TIMxx OC output must be at least N f_{HRTIM} clock cycles long to be detected by the HRTIM burst mode controller.

Burst mode triggers

To trigger the burst operation, 32 sources are available and are selected using the HRTIM_BMTRGR register:

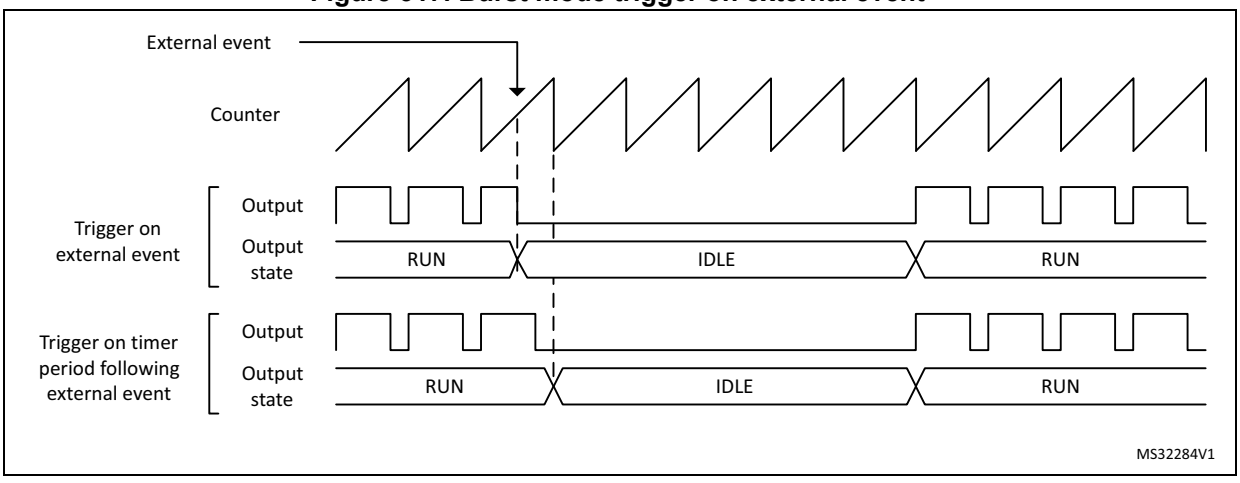
- Software trigger (set by software and reset by hardware)
- 6 Master timer events: repetition, reset/roll-over, Compare 1 to 4
- 5 x 4 events from timers A..E: repetition, reset/roll-over, Compare 1 and 2
- hrtim_evt7 (including TIMA event filtering) and hrtim_evt8 (including TIMD event filtering)
- Timer A period following hrtim_evt7 (including TIMA event filtering)
- Timer D period following hrtim_evt8 (including TIMD event filtering)
- On-chip events coming from other general purpose timer (hrtim_bm_trg output:TIM7_TRGO output)

These sources can be combined to have multiple concurrent triggers.

Burst mode is not re-triggerable. In continuous mode, new triggers are ignored until the burst mode is terminated, while in single-shot mode, the triggers are ignored until the current burst completion including run periods (HRTIM_BMPER+1 cycles). This is also valid for software trigger (the software bit is reset by hardware even if it is discarded).

Figure 317 shows how the burst mode is started in response to an external event, either immediately or on the timer period following the event.

Figure 317. Burst mode trigger on external event



For TAEEV7 and TDEEV8 combined triggers (trigger on a Timer period following an external event), the external event detection is always active, regardless of the burst mode programming and the on-going burst operation:

- When the burst mode is enabled (BME=1) or the trigger is enabled (TAEEV7 or TDEEV8 bit set in the BMTRG register) in between the external event and the timer period event, the burst is triggered.
- The single-shot burst mode is re-triggered even if the external event occurs before the burst end (as long as the corresponding period happens after the burst).

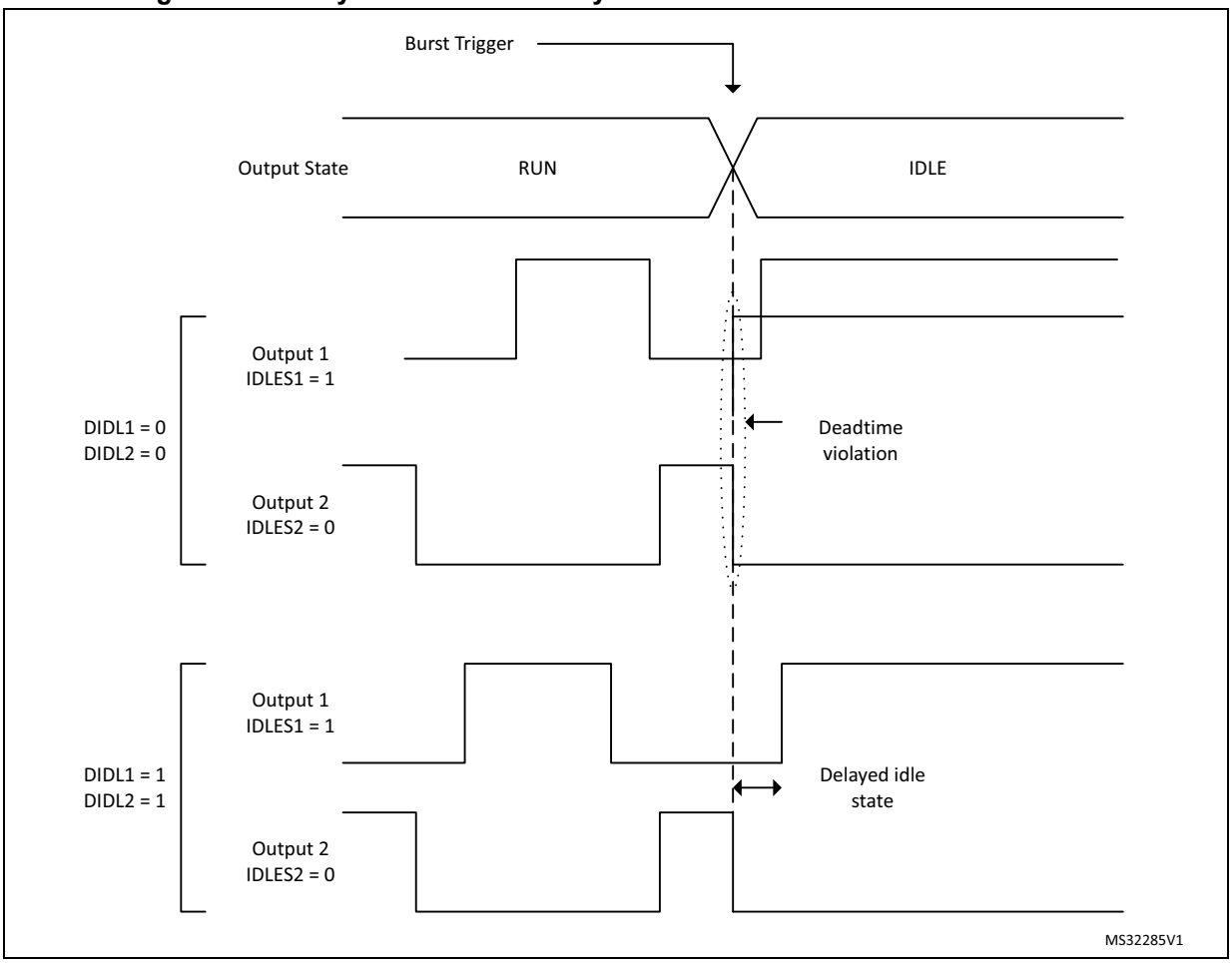
Note: TAEEV7 and TDEEV8 triggers are valid only after a period event. If the counter is reset before the period event, the pending hrtim_evt7/8 event is discarded.

Burst mode delayed entry

By default, the outputs are taking their idle level (as per IDLES1 and IDLES2 setting) immediately after the burst mode trigger.

It is also possible to delay the burst mode entry and force the output to an inactive state during a programmable period before the output takes its idle state. This is useful when driving two complementary outputs, one of them having an active idle state, to avoid a deadtime violation as shown on *Figure 318*. This prevents any risk of shoot through current in half-bridges, but causes a delayed response to the burst mode entry.

Figure 318. Delayed burst mode entry with deadtime enabled and IDLESx = 1



The delayed burst entry mode is enabled with DIDLx bit in the HRTIM_OUTxR register (one enable bit per output). It forces a deadtime insertion before the output takes its idle state. Each TIMx output has its own deadtime value:

- DTRx[8:0] on output 1 when DIDL1 = 1
- DTFx[8:0] on output 2 when DIDL2 = 1

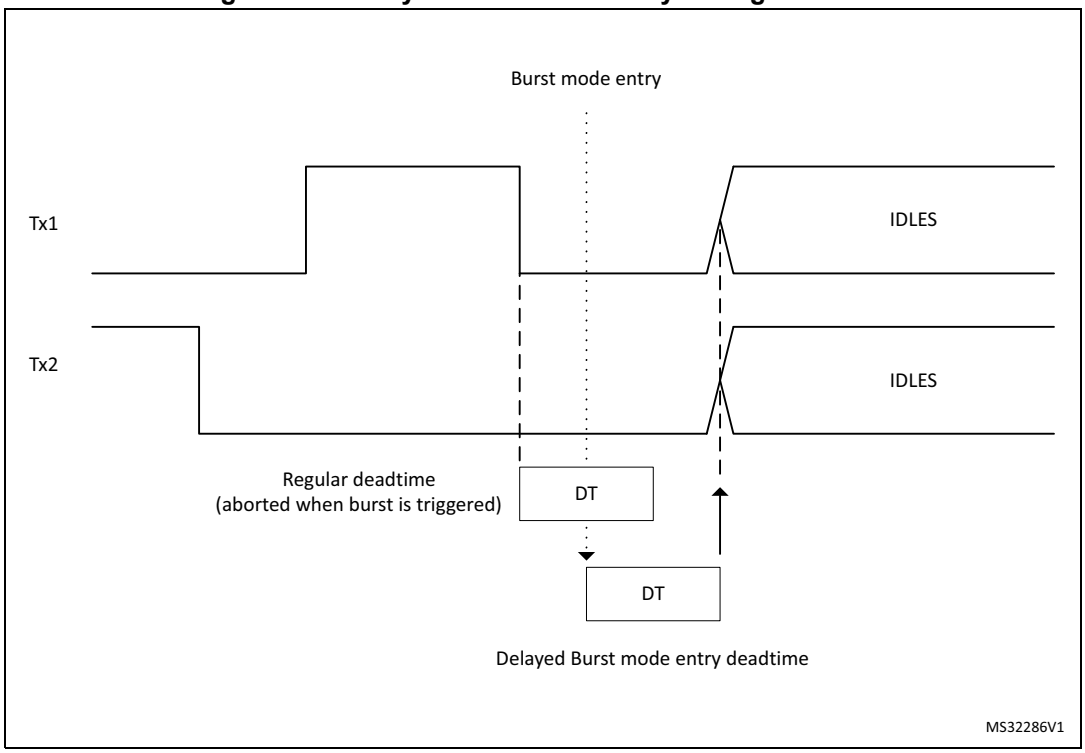
DIDLx bits can be set only if one of the outputs has an active idle level during the burst mode (IDLES = 1) and only when positive deadtimes are used (SDTR/SDTF set to 0).

Note: The delayed burst entry mode uses deadtime generator resources. Consequently, when any of the 2 DIDLx bits is set and the corresponding timing unit uses the deadtime insertion (DTEN bit set in HRTIM_OUTxR), it is not possible to use the timerx output 2 as a filter for external events (Tx2 filtering signal is not available).

When durations defined by DTRx[8:0] and DTFx[8:0] are lower than 3 f_{HRTIM} clock cycle periods, the limitations related to the narrow pulse management listed in *Section 36.3.6* must be applied.

When the burst mode entry arrives during the regular deadtime, it is aborted and a new deadtime is re-started corresponding to the inactive period, as on *Figure 319*.

Figure 319. Delayed Burst mode entry during deadtime



Burst mode exit

The burst mode exit is either forced by software (in continuous mode) or once the idle period is elapsed (in single-shot mode). In both cases, the counter is re-started immediately (if it was hold in a reset state with MTBM or TxBM bit = 1), but the effective output state transition from the idle to active mode only happens after the programmed set/reset event.

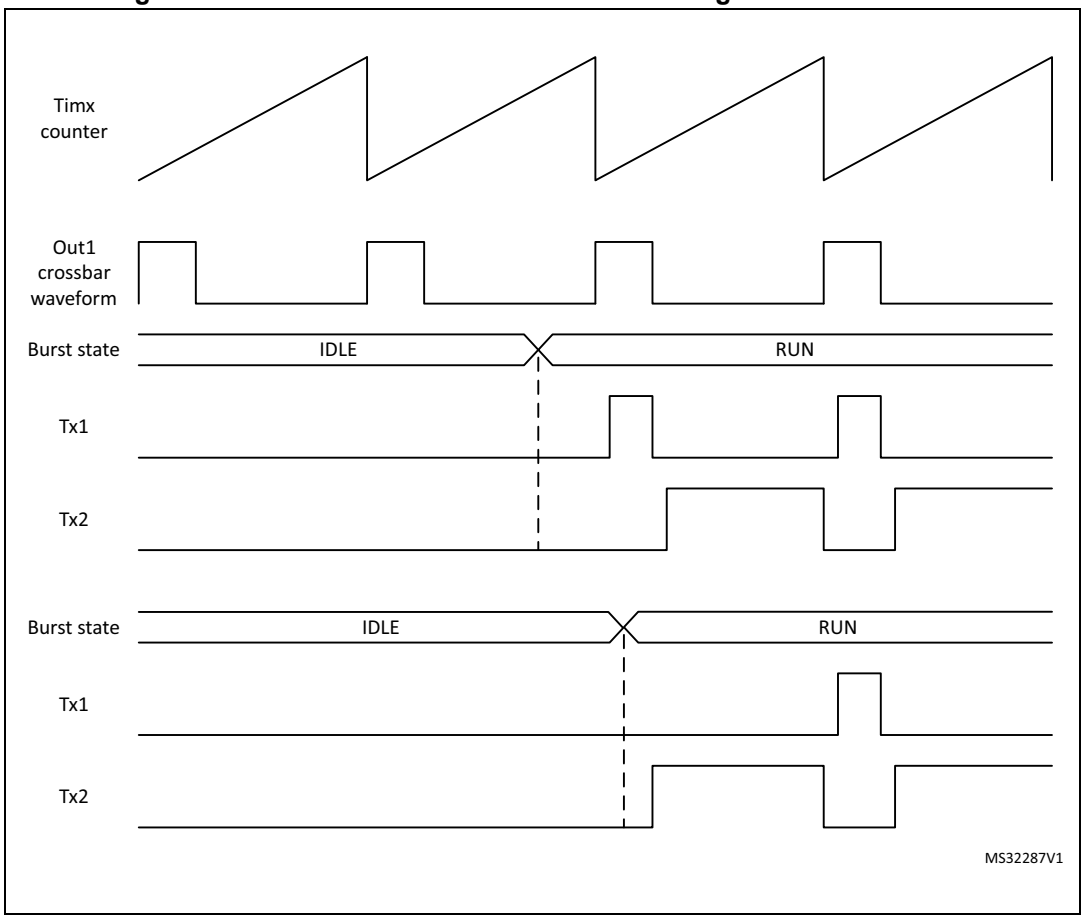
A burst period interrupt is generated in single-shot and continuous modes when BMPERIE enable bit is set in the HRTIM_IER register. This interrupt can be used to synchronize the burst mode exit with a burst period in continuous burst mode.

Figure 320 shows how a normal operation is resumed when the deadtime is enabled. Although the burst mode exit is immediate, this is only effective on the first set event on any of the complementary outputs.

Two different cases are presented:

1. The burst mode ends while the signal is inactive on the crossbar output waveform. The active state is resumed on Tx1 and Tx2 on the set event for the Tx1 output, and the Tx2 output does not take the complementary level on burst exit.
2. The burst mode ends while the crossbar output waveform is active: the activity is resumed on the set event of Tx2 output, and Tx1 does not take the active level immediately on burst exit.

Figure 320. Burst mode exit when the deadtime generator is enabled



The behavior described above is slightly different when the push-pull mode is enabled. The push-pull mode forces an output reset at the beginning of the period if the output is inactive, or symmetrically forces an active level if the output was high during the preceding period.

Consequently, an output with an active idle state can be reset at the time the burst mode is exited even if no transition is explicitly programmed. For symmetrical reasons, an output can be set at the time the burst mode is exited even if no transition is explicitly programmed, in case it was active when it entered in idle state.

Burst mode registers preloading and update

BMPREN bit (Burst mode Preload Enable) allows to have the burst mode compare and period registers preloaded (HRTIM_BMCMP and HRTIM_BMPER).

When BMPREN is set, the transfer from preload to active register happens:

- when the burst mode is enabled (BME = 1),
- at the end of the burst mode period.

A write into the HRTIM_BMPER period register disables the update temporarily, until the HRTIM_BMCMP compare register is written, to ensure the consistency of the two registers when they are modified.

If the compare register only needs to be changed, a single write is necessary. If the period only needs to be changed, it is also necessary to re-write the compare to have the new values taken into account.

When BMPREN bits is reset, the write access into BMCMPR and BMPER directly updates the active register. In this case, it is necessary to consider when the update is done during the overall burst period, for the 2 cases below:

a) Compare register update

If the new compare value is above the current burst mode counter value, the new compare is taken into account in the current period.

If the new compare value is below the current burst mode counter value, the new compare is taken into account in the next burst period in continuous mode, and ignored in single-shot mode (no compare match will occur and the idle state will last until the end of the idle period).

b) Period register update

If the new period value is above the current burst mode counter value, the change is taken into account in the current period.

Note: If the new period value is below the current burst mode counter value, the new period will not be taken into account, the burst mode counter will overflow (at 0xFFFF) and the change will be effective in the next period. In single-shot mode, the counter will roll over at 0xFFFF and the burst mode will re-start for another period up to the new programmed value.

Burst mode emulation using a compound register

The burst mode controller only controls one or a set of timers for a single converter. When the burst mode is necessary for multiple independent timers, it is possible to emulate a simple burst mode controller using the DMA and the HRTIM_CMP1CxR compound register, which holds aliases of both the repetition and the Compare 1 registers.

This is applicable to a converter which only requires a simple PWM (typically a buck converter), where the duty cycle only needs to be updated. In this case, the CMP1 register is used to reset the output (and define the duty cycle), while it is set on the period event.

In this case, a single 32-bit write access in CMP1CxR is sufficient to define the duty cycle (with the CMP1 value) and the number of periods during which this duty cycle is maintained (with the repetition value). To implement a burst mode, it is then only necessary to transfer by DMA (upon repetition event) two 32-bit data in continuous mode, organized as follows:

CMPC1xR = {REP_Run; CMP1 = Duty_Cycle}, {REP_Idle; CMP1 = 0}

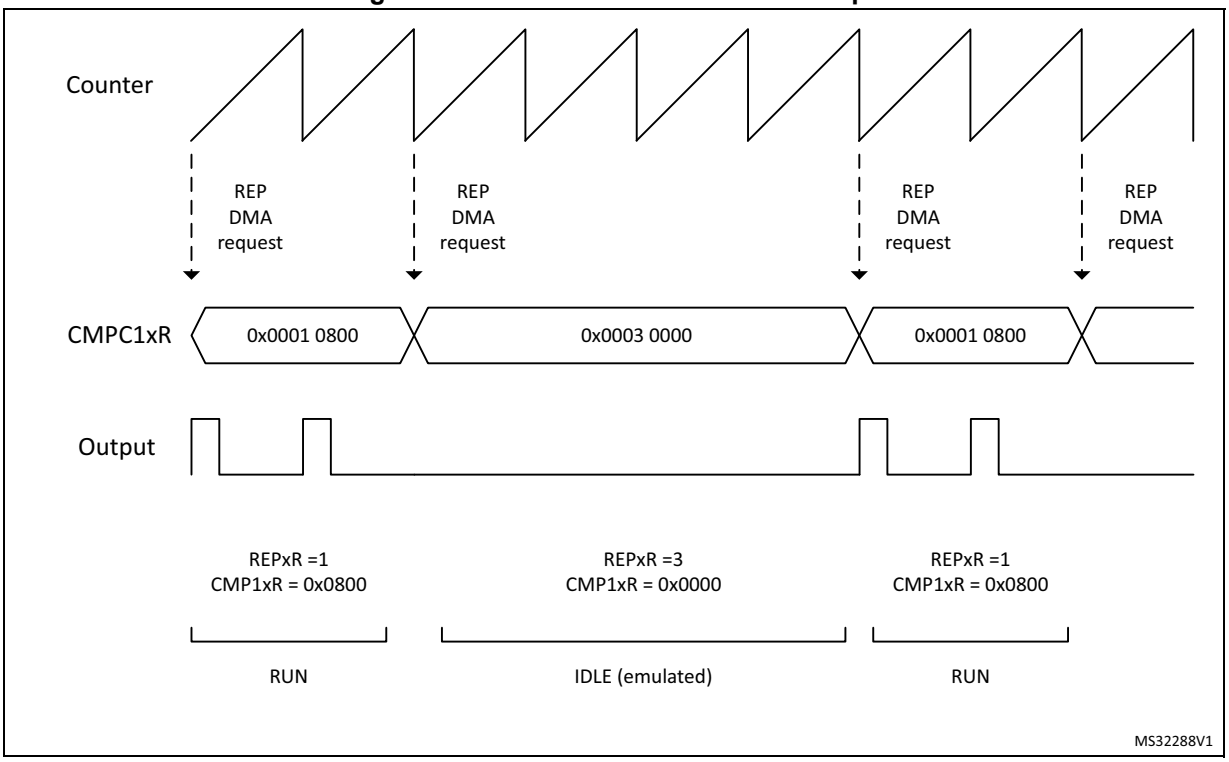
For instance, the values:

{0x0003 0000}: CMP1 = 0 for 3 periods

{0x0001 0800}: CMP1 = 0x0800 for 1 period

will provide a burst mode with 2 periods active every 6 PWM periods, as shown on *Figure 321*.

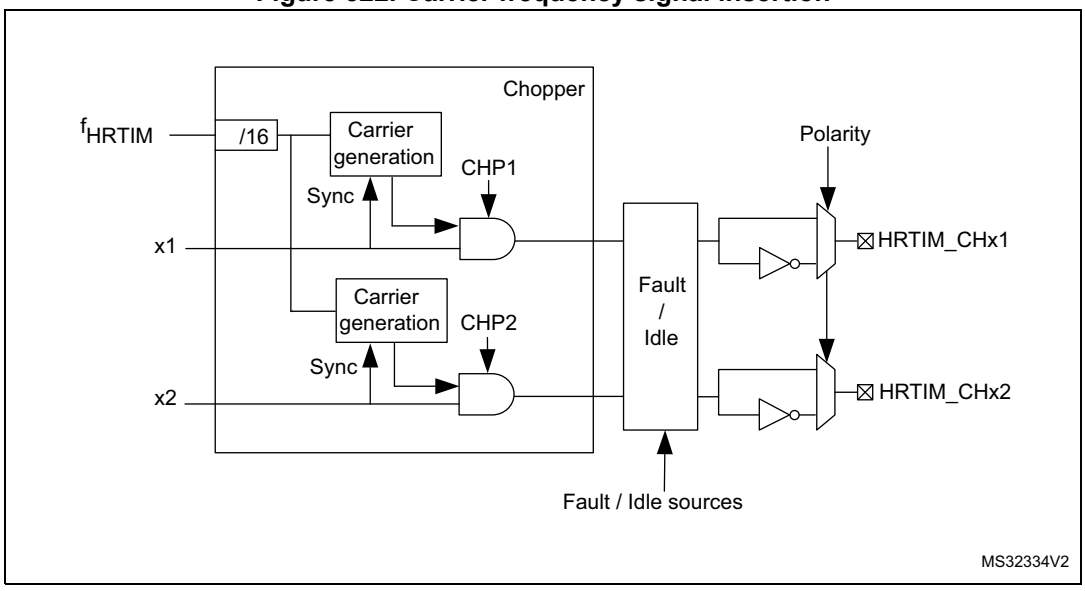
Figure 321. Burst mode emulation example



36.3.14 Chopper

A high-frequency carrier can be added on top of the timing unit output signals to drive isolation transformers. This is done in the output stage before the polarity insertion, as shown on *Figure 322*, using CHP1 and CHP2 bits in the HRTIM_OUTxR register, to enable chopper on outputs 1 and 2, respectively.

Figure 322. Carrier frequency signal insertion



The chopper parameters can be adjusted using the HRIM_CHPxR register, with the possibility to define a specific pulsewidth at the beginning of the pulse, to be followed by a carrier frequency with programmable frequency and duty cycle, as in *Figure 323*.

CARFRQ[3:0] bits define the frequency, ranging from 156 MHz to 25 MHz (for f_{HRTIM} = 400 MHz) following the formula $F_{CHPFRQ} = f_{HRTIM} / (16 \times (CARFRQ[3:0]+1))$.

The duty cycle can be adjusted by 1/8 step with CARDTY[2:0], from 0/8 up to 7/8 duty cycle. When CARDTY[2:0] = 000 (duty cycle = 0/8), the output waveform only contains the starting pulse following the rising edge of the reference waveform, without any added carrier.

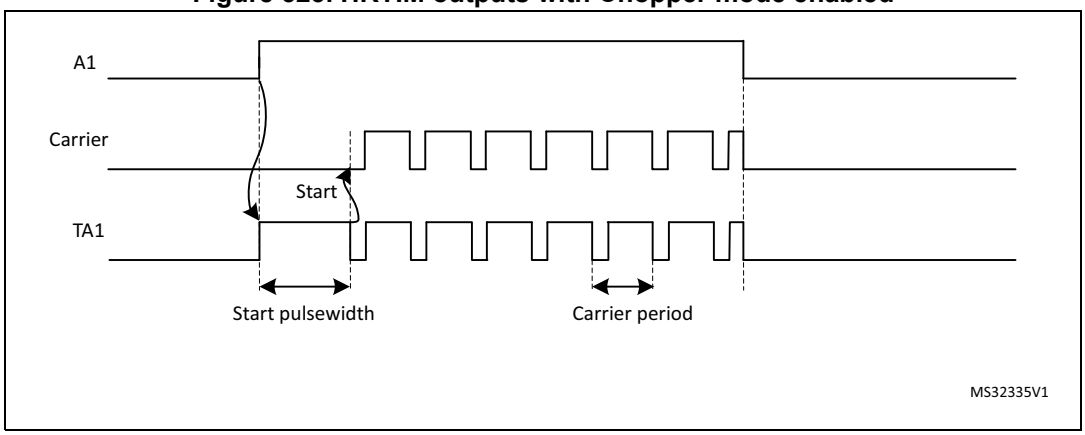
The pulsewidth of the initial pulse is defined using the STRPW[3:0] bitfield as follows: $t1STPW = (STRPW[3:0]+1) \times 16 \times t_{HRTIM}$ and ranges from 40 ns to 0.63 µs (for f_{HRTIM}=400 MHz).

The carrier frequency parameters are defined based on the f_{HRTIM} frequency, and are not dependent from the CKPSC[2:0] setting.

In chopper mode, the carrier frequency and the initial pulsewidth are combined with the reference waveform using an AND function. A synchronization is performed at the end of the initial pulse to have a repetitive signal shape.

The chopping signal is stopped at the end of the output waveform active state, without waiting for the current carrier period to be completed. It can thus contain shorter pulses than programmed.

Figure 323. HRTIM outputs with Chopper mode enabled



Note: CHP1 and CHP2 bits must be set prior to the output enable done with TxyOEN bits in the HRTIM_OENR register.

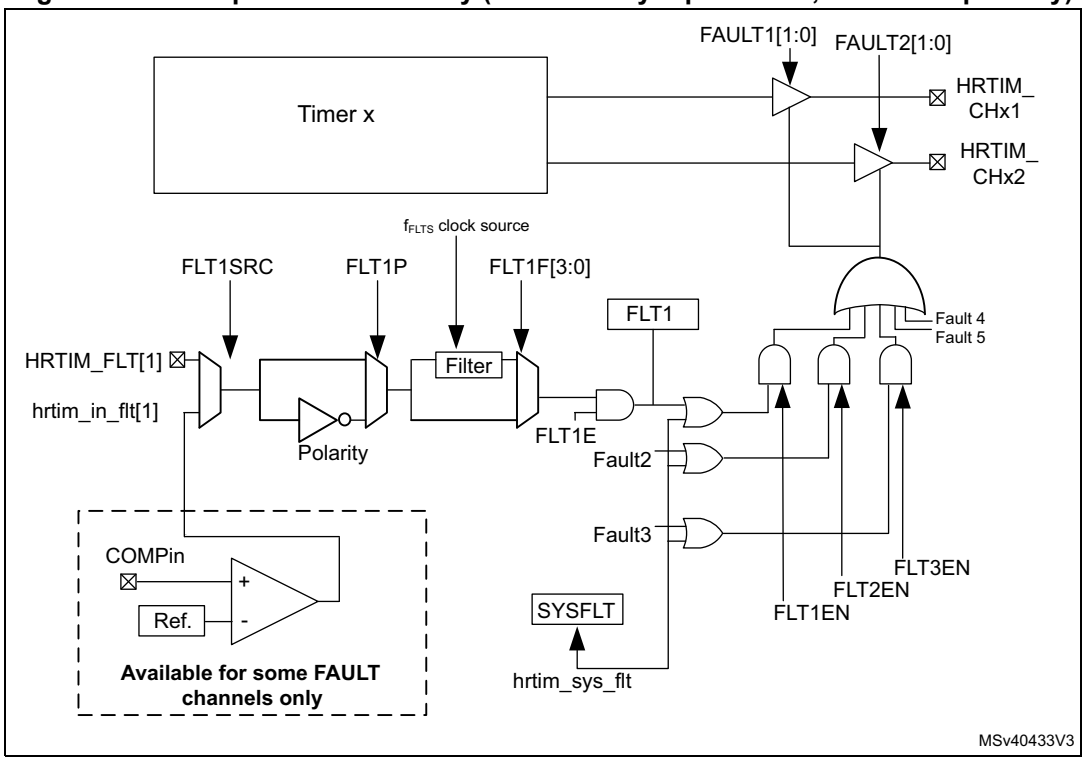
CARFRQ[2:0], CARDTY[2:0] and STRPW[3:0] bitfields cannot be modified while the chopper mode is active (at least one of the two CHPx bits is set).

36.3.15 Fault protection

The HRTIMER has a versatile fault protection circuitry to disable the outputs in case of an abnormal operation. Once a fault has been triggered, the outputs take a predefined safe state. This state is maintained until the output is re-enabled by software. In case of a permanent fault request, the output will remain in its fault state, even if the software attempts to re-enable them, until the fault source disappears.

The HRTIM has 5 FAULT input channels; all of them are available and can be combined for each of the 5 timing units, as shown on *Figure 324*.

Figure 324. Fault protection circuitry (FAULT1 fully represented, FAULT2..5 partially)



Each fault channel is fully configurable using HRTIM_FLTINR1 and HRTIM_FLTINR2 registers before being routed to the timing units. FLTxSRC bit selects the source of the Fault signal, that can be either a digital input or an internal event (built-in comparator output).

Table 311 summarizes the available sources for each of the 10 faults channels:

Table 311. Fault inputs

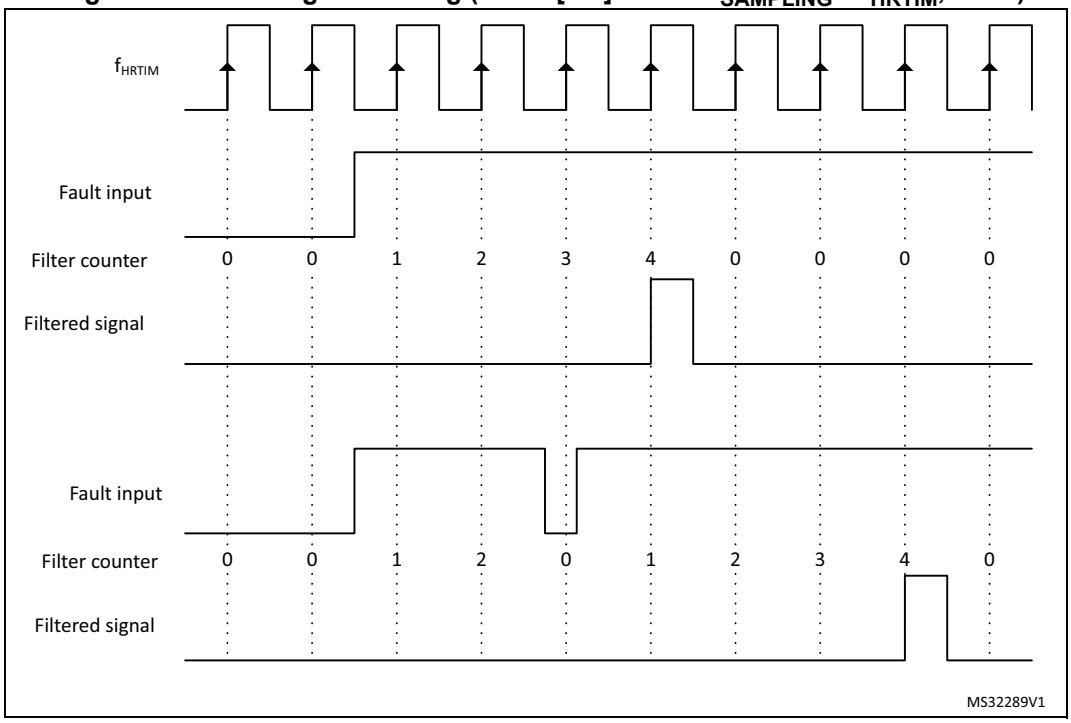
Fault channel	External Input (FLTxSRC = 0)	On-chip source (FLTxSRC = 1)
FAULT 1	PA15	COMP1
FAULT 2	PC11	COMP2
FAULT 3	PD4	NC
FAULT 4	PB3	NC
FAULT 5	PG10	NC

The polarity of the signal can be selected to define the active level, using the FLTxP polarity bit in HRTIM_FLTINRx registers. If FLTxP = 0, the signal is active at low level; if FLTxP = 1, it is active when high.

The fault information can be filtered after the polarity setting. If FLTxF[3:0] bitfield is set to 0000, the signal is not filtered and will act asynchronously, independently from the f_{HRTIM} clock. For all other FLTxF[3:0] bitfield values, the signal is digitally filtered. The digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external

event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected). *Figure 325* shows how a spurious fault signal is filtered.

Figure 325. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4)



The filtering period ranges from 2 cycles of the f_{HRTIM} clock up to 8 cycles of the f_{FLTS} clock divided by 32. f_{FLTS} is defined using FLTSD[1:0] bits in the HRTIM_FLTINR2 register. *Table 312* summarizes the sampling rate and the filter length. A jitter of 1 sampling clock period must be subtracted from the filter length to take into account the uncertainty due to the sampling and have the effective filtering.

Table 312. Sampling rate and filter length vs FLTFxF[3:0] and clock setting

	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 400 MHz	
FLTFxF[3:0]	**00**	**01**	**10**	**11**	**Min**	**Max**
0001,0010,0011	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}, N =2 5 ns	f_{HRTIM}, N =8 20 ns
0100, 0101	f_{HRTIM} /2	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /2, N = 6 30 ns	f_{HRTIM} /16, N = 8 320 ns
0110, 0111	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /4, N = 6 60 ns	f_{HRTIM} /32, N = 8 640 ns
1000, 1001	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /8, N = 6 120 ns	f_{HRTIM} /64, N = 8 1.28 µs

Table 312. Sampling rate and filter length vs FLTFxF[3:0] and clock setting (continued)

	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 400 MHz	
FLTFxF[3:0]	**00**	**01**	**10**	**11**	**Min**	**Max**
1010, 1011, 1100	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /16, N = 5 200 ns	f_{HRTIM} /128, N = 8 2.56 µs
1101, 1110, 1111	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /256	f_{HRTIM} /32, N = 5 400 ns	f_{HRTIM} /256, N = 8 5.12 µs

System fault input (hrtim_sys_flt)

This fault is provided by the MCU Class B circuitry (see the System configuration controller (SYSCFG) section for details) and corresponds to a system fault coming from:

- the Clock Security System
- the SRAM parity checker
- the Cortex®-M7-lockup signal
- the PVD detector

This input overrides the FAULT inputs and disables all outputs having FAULTy[1:0] = 01, 10, 11.

For each FAULT channel, a write-once FLTxLCK bit in the HRTIM_FLTxR register allows to lock FLTxE, FLTxP, FLTxSRC, FLTxF[3:0] bits (it renders them read-only), for functional safety purpose. If enabled, the fault conditioning set-up is frozen until the next HRTIM or system reset.

Once the fault signal is conditioned as explained above, it is routed to the timing units. For any of them, the 5 fault channels are enabled using bits FLT1EN to FLT5EN in the HRTIM_FLTxR register, and they can be selected simultaneously (the sysfault is automatically enabled as long as the output is protected by the fault mechanism). This allows to have, for instance:

- One fault channel simultaneously disabling several timing units
- Multiple fault channels being ORed to disable a single timing unit

A write-once FLTLCK bit in the HRTIM_FLTxR register allows to lock FLTxEN bits (it renders them read-only) until the next reset, for functional safety purpose. If enabled, the timing unit fault-related set-up is frozen until the next HRTIM or system reset.

For each of the timers, the output state during a fault is defined with FAULT1[1:0] and FAULT2[1:0] bits in the HRTIM_OUTxR register (see *Section 36.3.12*).

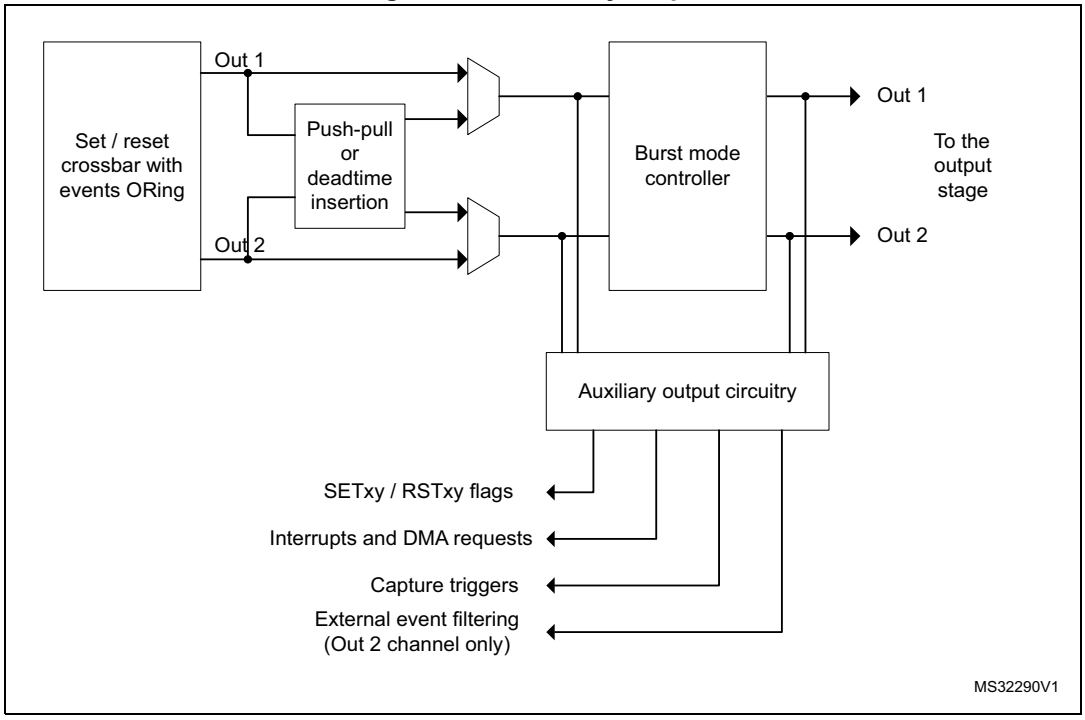
36.3.16 Auxiliary outputs

Timer A to E have auxiliary outputs in parallel with the regular outputs going to the output stage. They provide the following internal status, events and signals:

- SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset
- External event filters following a Tx2 output copy (see details in *Section 36.3.8*)

The auxiliary outputs are taken either before or after the burst mode controller, depending on the HRTIM operating mode. An overview is given on *Figure 326*.

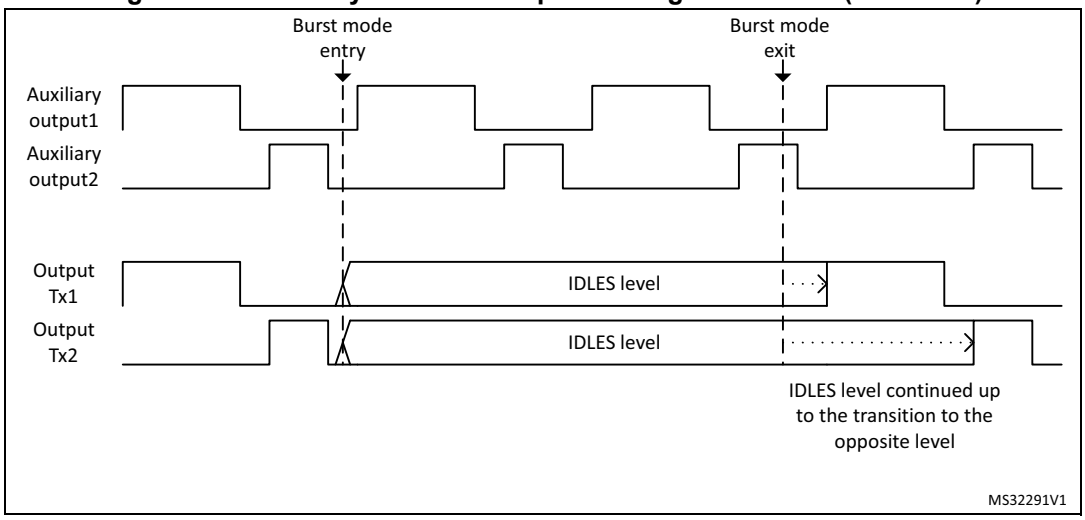
Figure 326. Auxiliary outputs



By default, the auxiliary outputs are copies of outputs Tx1 and Tx2. The exceptions are:

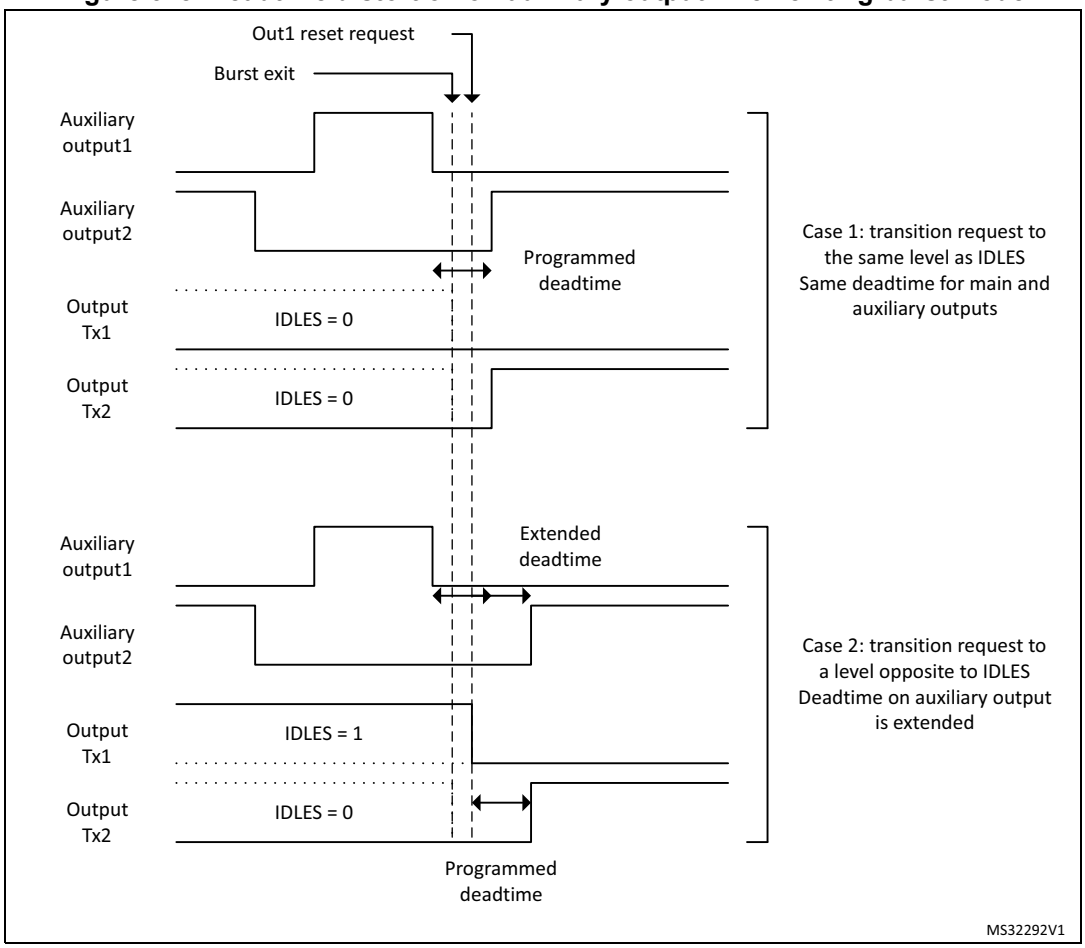
- The delayed idle and the balanced idle protections, when the deadtime is disabled (DTEN = 0). When the protection is triggered, the auxiliary outputs are maintained and follow the signal coming out of the crossbar. On the contrary, if the deadtime is enabled (DTEN = 1), both main and auxiliary outputs are forced to an inactive level.
- The burst mode (TCEN=1, IDLEMx=1); there are 2 cases:
  a) If DTEN=0 or DIDLx=0, the auxiliary outputs are not affected by the burst mode entry and continue to follow the reference signal coming out of the crossbar (see *Figure 327*).
  b) If the deadtime is enabled (DTEN=1) together with the delayed burst mode entry (DIDLx=1), the auxiliary outputs have the same behavior as the main outputs. They are forced to the IDLES level after a deadtime duration, then they keep this level during all the burst period. When the burst mode is terminated, the IDLES level is maintained until a transition occurs to the opposite level, similarly to the main output.

Figure 327. Auxiliary and main outputs during burst mode (DIDLx = 0)



The signal on the auxiliary output can be slightly distorted when exiting from the burst mode or when re-enabling the outputs after a delayed protection, if this happens during a deadtime. In this case, the deadtime applied to the auxiliary outputs is extended so that the deadtime on the main outputs is respected. *Figure 328* gives some examples.

Figure 328. Deadtime distortion on auxiliary output when exiting burst mode



36.3.17 Synchronizing the HRTIM with other timers or HRTIM instances

The HRTIM provides options for synchronizing multiple HRTIM instances, as a master unit (generating a synchronization signal) or as a slave (waiting for a trigger to be synchronized). This feature can also be used to synchronize the HRTIM with other timers, either external or on-chip. The synchronization circuitry is controlled inside the master timer.

Synchronization output

This section explains how the HRTIM must be configured to synchronize external resources and act as a master unit.

Four events can be selected as the source to be sent to the synchronization output. This is done using SYNCSRC[1:0] bits in the HRTIM_MCR register, as follows:

- 00: Master timer Start
  This event is generated when MCEN bit is set or when the timer is re-started after having reached the period value in single-shot mode. It is also generated on a reset which occurs during the counting (when CONT or RETRIG bits are set).
- 01: Master timer Compare 1 event
- 10: Timer A start
  This event is generated when TACEN bit is set or when the counter is reset and re-starts counting in response to this reset. The following counter reset events are not propagated to the synchronization output: counter roll-over in continuous mode, and discarded reset request in single-shot non-retriggerable mode. The reset is only taken into account when it occurs during the counting (CONT or RETRIG bits are set).
- 11: Timer A Compare 1 event

SYNCOUT[1:0] bits in the HRTIM_MCR register specify how the synchronization event is generated.

The synchronization pulses are generated on the HRTIM_SCOUT output pin, with SYNCOUT[1:0] = 1x. SYNCOUT[0] bit specifies the polarity of the synchronization signal. If SYNCOUT[0] = 0, the HRTIM_SCOUT pin has a low idle level and issues a positive pulse of 16 f_{HRTIM} clock cycles length for the synchronization). If SYNCOUT[0] = 1, the idle level is high and a negative pulse is generated.

Note: The synchronization pulse is followed by an idle level of 16 f_{HRTIM} clock cycles during which any new synchronization request is discarded. Consequently, the maximum synchronization frequency is $f_{HRTIM}/32$.

The idle level on the HRTIM_SCOUT pin is applied as soon as the SYNCOUT[1:0] bits are enabled (i.e. the bitfield value is different from 00).

The synchronization output initialization procedure must be done prior to the configuration of the MCU outputs and counter enable, in the following order:

1. SYNCOUT[1:0] and SYNCSRC[1:0] bitfield configuration in HRTIM_MCR
2. HRTIM_SCOUT pin configuration (see the General-purpose I/Os section)
3. Master or Timer A counter enable (MCEN or TACEN bit set)

When the synchronization input mode is enabled and starts the counter (using SYNCSTRTM/SYNCSTRTx bits) simultaneously with the synchronization output mode (SYNCSRC[1:0] = 00 or 10), the output pulse is generated only when the counter is starting or is reset while running. Any reset request clearing the counter without causing it to start will not affect the synchronization output.

Synchronization input

The HRTIM can be synchronized by external sources, as per the programming of the SYNCIN[1:0] bits in the HRTIM_MCR register:

- 00: synchronization input is disabled
- 01: reserved configuration
- 10: the on-chip TIM1 general purpose timer (TIM1 TRGO output)
- 11: a positive pulse on the HRTIM_SCIN input pin

This bitfield cannot be changed once the destination timer (master timer or timing unit) is enabled (MCEN and/or TxCEN bit set).

The HRTIM_SCIN input is rising-edge sensitive. The timer behavior is defined with the following bits present in HRTIM_MCR and HRTIM_TIMxCR registers (see *Table 313* for details):

- Synchronous start: the incoming signal starts the timer's counter (SYNCSTRTM and/or SYNCSTRTx bits set). TxCEN (MCEN) bits must be set to have the timer enabled and the counter ready to start. In continuous mode, the counter will not start until the synchronization signal is received.
- Synchronous reset: the incoming signal resets the counter (SYNCRSTM and/or SYNCRSTx bits set). This event decrements the repetition counter as any other reset event.

The synchronization events are taken into account only once the related counters are enabled (MCEN or TxCEN bit set). A synchronization request triggers a SYNC interrupt.

Note: A synchronized start event resets the counter if the current counter value is above the active period value.

The effect of the synchronization event depends on the timer operating mode, as summarized in *Table 313*.

Table 313. Effect of sync event vs timer operating modes

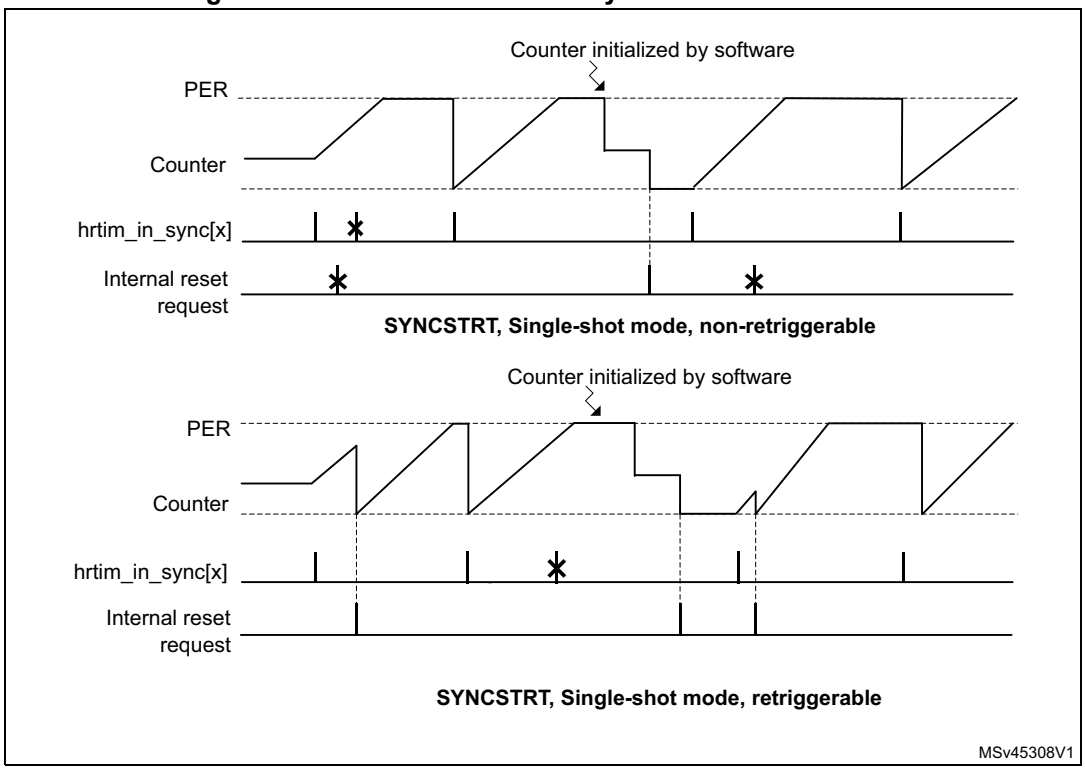
Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot non-retriggerable	0	1	Start events are taken into account when the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be re-started with the synchronization). Any reset occurring during the counting is ignored (as during regular non-retriggerable mode).
	1	X	Reset events are starting the timer counting. They are taken into account only if the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. When multiple reset requests are selected (from HRTIM_SCIN and from internal events), only the first arriving request is taken into account.

Table 313. Effect of sync event vs timer operating modes (continued)

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot retriggerable	0	1	The counter start is effective only if the counter is not started or period is elapsed. Any synchronization event occurring after counter start has no effect. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be started by the synchronization). A reset occurring during counting is taken into account (as during regular retriggerable mode).
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.
Continuous mode	0	1	The timer is enabled (MCEN or TxCEN bit set) and is waiting for the synchronization event to start the counter. Any synchronization event occurring after the counter start has no effect (the counter can solely be started by the synchronization). A reset request clears the counter but does not start it.
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.

Figure 329 presents how the synchronized start is done in single-shot mode.

Figure 329. Counter behavior in synchronized start mode



36.3.18 ADC triggers

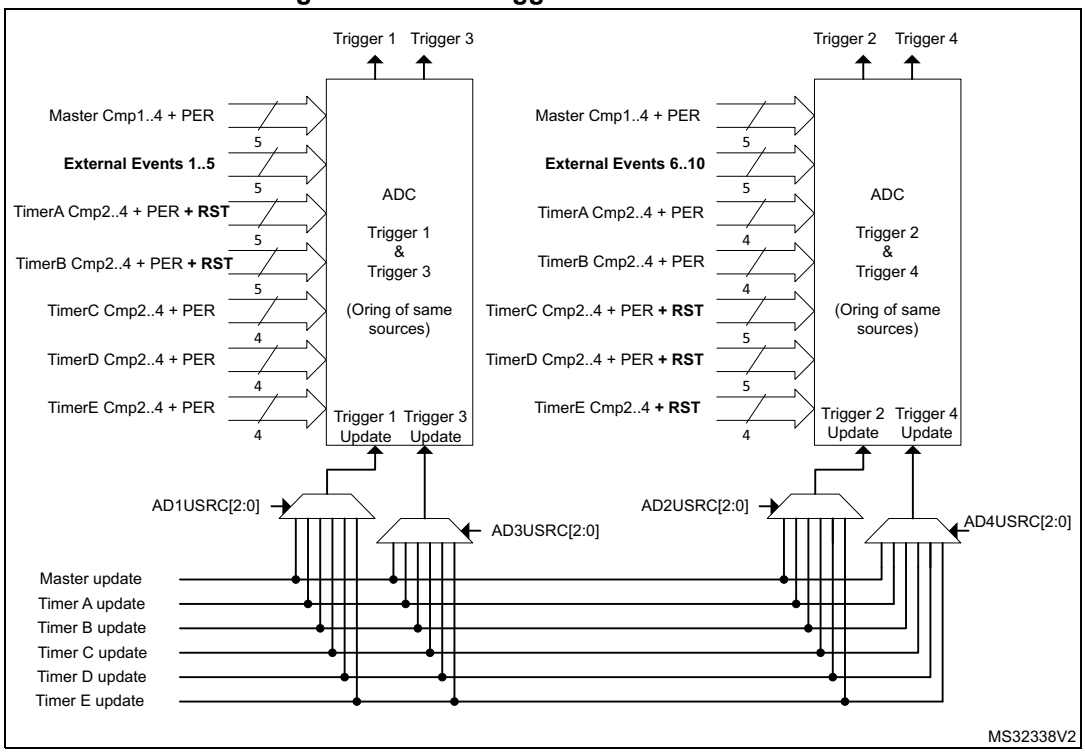
The ADCs can be triggered by the master and the 5 timing units.

4 independent triggers are available to start both the regular and the injected sequencers of the 2 ADCs. Up to 32 events can be combined (ORed) for each trigger output, in registers HRTIM_ADC1R to HRTIM_ADC4R, as shown on *Figure 330*. Triggers 1/3 and 2/4 are using the same source set.

The external events can be used as a trigger. They are taken right after the conditioning defined in HRTIM_EECRx registers, and are not depending on EEFxR1 and EEFxR2 register settings.

Multiple triggering is possible within a single switching period by selecting several sources simultaneously. A typical use case is for a non-overlapping multiphase converter, where all phases can be sampled in a row using a single ADC trigger output.

Figure 330. ADC trigger selection overview



HRTIM_ADC1R to HRTIM_ADC4R registers are preloaded and can be updated synchronously with the timer they are related to. The update source is defined with ADxUSRC[2:0] bits in the HRTIM_CR1 register.

For instance, if ADC trigger 1 outputs Timer A CMP2 events (HRTIM_ADC1R = 0x0000 0400), HRTIM_ADC1R will be typically updated simultaneously with Timer A (AD1USRC[2:0] = 001).

When the preload is disabled (PREEN bit reset) in the source timer, the HRTIM_ADCxR registers are not preloaded either: a write access will result in an immediate update of the trigger source.

36.3.19 DAC triggers

The HRTIMER allows to have the embedded DACs updated synchronously with the timer updates.

The update events from the master timer and the timer units can generate DAC update triggers on any of the 3 hrtim_dac_trgx outputs.

Note: Each timer has its own DAC-related control register.

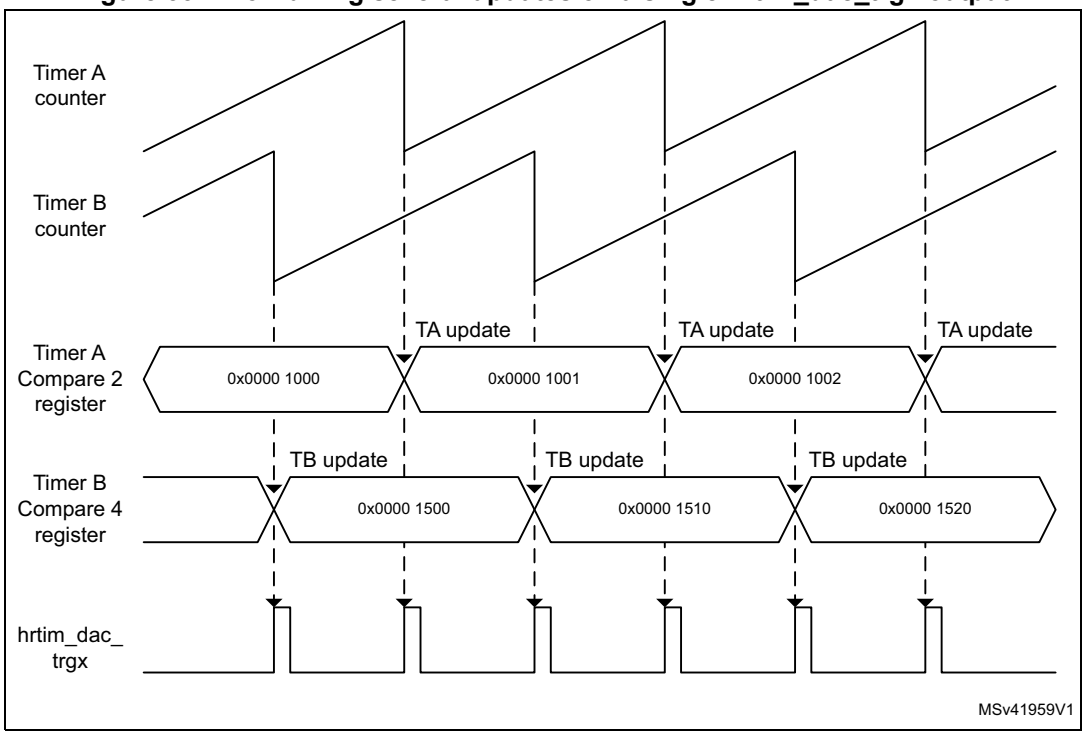
DACSYNC[1:0] bits of the HRTIM_MCR and HRTIM_TIMxCR registers are programmed as follows:

- 00: No update generated
- 01: Update generated on hrtim_dac_trg1
- 10: Update generated on hrtim_dac_trg2
- 11: Update generated on hrtim_dac_trg3

An output pulse of 1 f_{HRTIM} clock periods is generated on the hrtim_dac_trgx output.

When DACSYNC[1:0] bits are enabled in multiple timers, the hrtim_dac_trgx output will consist of an OR of all timers' update events. For instance, if DACSYNC = 1 in timer A and in timer B, the update event in timer A will be ORed with the update event in timer B to generate a DAC update trigger on the corresponding hrtim_dac_trgx output, as shown on *Figure 331*.

Figure 331. Combining several updates on a single hrtim_dac_trgx output



hrtim_dac_trgx pins are connected to the DACs as follows:

- hrtim_dac_trg1: DAC1_CH1 trigger input 9 (TSEL1[2:0] = 1001 in DAC_CR of DAC1 peripheral)
- hrtim_dac_trg2: DAC1_CH2 trigger input 10 (TSEL1[2:0] = 1010 in DAC_CR of DAC1 peripheral)
- hrtim_dac_trg3: not connected

36.3.20 HRTIM Interrupts

7 interrupts can be generated by the master timer:

- Master timer registers update
- Synchronization event received
- Master timer repetition event
- Master Compare 1 to 4 event

14 interrupts can be generated by each timing unit:

- Delayed protection triggered
- Counter reset or roll-over event
- Output 1 and output 2 reset (transition active to inactive)
- Output 1 and output 2 set (transition inactive to active)
- Capture 1 and 2 events
- Timing unit registers update
- Repetition event
- Compare 1 to 4 event

8 global interrupts are generated for the whole HRTIM:

- System fault and Fault 1 to 5 (regardless of the timing unit attribution)
- Burst mode period completed

The interrupt requests are grouped in 7 vectors as follows:

- hrtim_mst_it: Master timer interrupts (Master Update, Sync Input, Repetition, MCMP1..4) and global interrupt except faults (Burst mode period)
- hrtim_tima_it: TIMA interrupts
- hrtim_timb_it: TIMB interrupts
- hrtim_timc_it: TIMC interrupts
- hrtim_timd_it: TIMD interrupts
- hrtim_time_it: TIME interrupts
- hrtim_fault_it: Dedicated vector all fault interrupts to allow high-priority interrupt handling

Table 314 is a summary of the interrupt requests, their mapping and associated control, and status bits.

Table 314. HRTIM interrupt summary

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_mst_it	Burst mode period completed	BMPER	BMPERIE	BMPERC
	Master timer registers update	MUPD	MUPDIE	MUPDC
	Synchronization event received	SYNC	SYNCIE	SYNCC
	Master timer repetition event	MREP	MREPIE	MREPC
	Master Compare 1 to 4 event	MCMP1	MCMP1IE	MCP1C
		MCMP2	MCMP2IE	MCP2C
		MCMP3	MCMP3IE	MCP3C
		MCMP4	MCMP4IE	MCP4C
hrtim_tima_it hrtim_timb_it hrtim_timc_it hrtim_timd_it hrtim_time_it	Delayed protection triggered	DLYPRT	DLYPRTIE	DLYPRTC
	Counter reset or roll-over event	RST	RSTIE	RSTC
	Output 1 and output 2 reset (transition active to inactive)	RSTx1	RSTx1IE	RSTx1C
		RSTx2	RSTx2IE	RSTx2C
	Output 1 and output 2 set (transition inactive to active)	SETx1	SETx1IE	SETx1C
		SETx2	SETx2IE	SETx2C
	Capture 1 and 2 events	CPT1	CPT1IE	CPT1C
		CPT2	CPT2IE	CPT2C
	Timing unit registers update	UPD	UPDIE	UPDC
	Repetition event	REP	REPIE	REPC
	Compare 1 to 4 event	CMP1	CMP1IE	CMP1C
		CMP2	CMP2IE	CMP2C
		CMP3	CMP3IE	CMP3C
		CMP4	CMP4IE	CMP4C
hrtim_fault_it	System fault	SYSFLT	SYSFLTIE	SYSFLTC
	Fault 1 to 5	FLT1	FLT1IE	FLT1C
		FLT2	FLT2IE	FLT2C
		FLT3	FLT3IE	FLT3C
		FLT4	FLT4IE	FLT4C
		FLT5	FLT5IE	FLT5C

36.3.21 DMA

Most of the events able to generate an interrupt can also generate a DMA request, even both simultaneously. Each timer (master, TIMA...E) has its own DMA enable register.

The individual DMA requests are ORed into 6 channels as follows:

- 1 channel for the master timer
- 1 channel per timing unit

Note: Before disabling a DMA channel (DMA enable bit reset in TIMxDIER), it is necessary to disable first the DMA controller.

Table 315 is a summary of the events with their associated DMA enable bits.

Table 315. HRTIM DMA request summary

DMA Channel	Event	DMA capable	DMA enable bit
hrtim_dma1 (Master timer)	Burst mode period completed	No	N/A
	Master timer registers update	Yes	MUPDDE
	Synchronization event received	Yes	SYNCDE
	Master timer repetition event	Yes	MREPDE
	Master Compare 1 to 4 event	Yes	MCMP1DE
		Yes	MCMP2DE
		Yes	MCMP3DE
		Yes	MCMP4DE
hrtim_dma2 (Timer A) hrtim_dma3 (Timer B) hrtim_dma4 (Timer C) hrtim_dma5 (Timer D) hrtim_dma6 (Timer E)	Delayed protection triggered	Yes	DLYPRTDE
	Counter reset or roll-over event	Yes	RSTDE
	Output 1 and output 2 reset (transition active to inactive)	Yes	RSTx1DE
		Yes	RSTx2DE
	Output 1 and output 2 set (transition inactive to active)	Yes	SETx1DE
		Yes	SETx2DE
	Capture 1 and 2 events	Yes	CPT1DE
		Yes	CPT2DE
	Timing unit registers update	Yes	UPDDE
	Repetition event	Yes	REPDE
	Compare 1 to 4 event	Yes	CMP1DE
		Yes	CMP2DE
		Yes	CMP3DE
		Yes	CMP4DE
N/A	System fault	No	N/A
	Fault 1 to 5	No	N/A
	Burst mode period completed	No	N/A

Burst DMA transfers

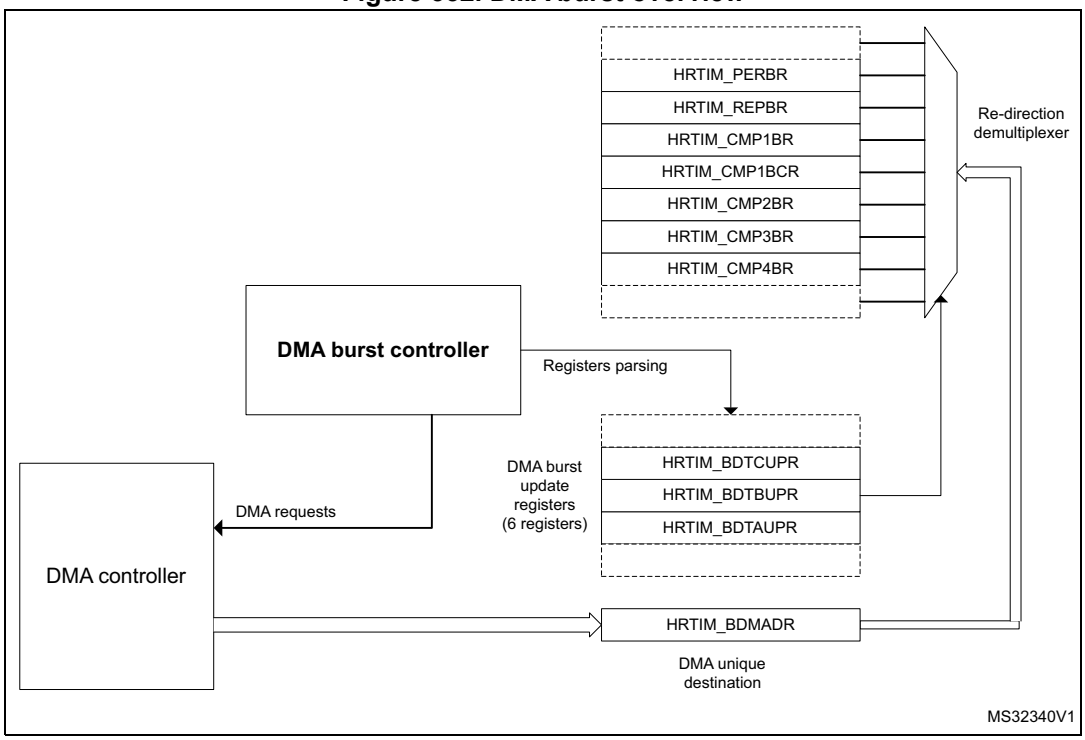
In addition to the standard DMA requests, the HRTIM features a DMA burst controller to have multiple registers updated with a single DMA request. This allows to:

- update multiple data registers with one DMA channel only,
- reprogram dynamically one or several timing units, for converters using multiple timer outputs.

The burst DMA feature is only available for one DMA channel, but any of the 6 channels can be selected for burst DMA transfers.

The principle is to program which registers are to be written by DMA. The master timer and TIMA..E have the burst DMA update register, where most of their control and data registers are associated with a selection bit: HRTIM_BDMUPR, HRTIM_BDTAUPR to HRTIM_BDTEUPR (this is applicable only for registers with write accesses). A redirection mechanism allows to forward the DMA write accesses to the HRTIM registers automatically, as shown on *Figure 332*.

Figure 332. DMA burst overview



When the DMA trigger occurs, the HRTIM generates multiple 32-bit DMA requests and parses the update register. If the control bit is set, the write access is redirected to the associated register. If the bit is reset, the register update is skipped and the register parsing is resumed until a new bit set is detected, to trigger a new request. Once the 6 update registers (HRTIM_BDMUPR, 5x HRTIM_BDTxUPR) are parsed, the burst is completed and the system is ready for another DMA trigger (see the flowchart on *Figure 333*).

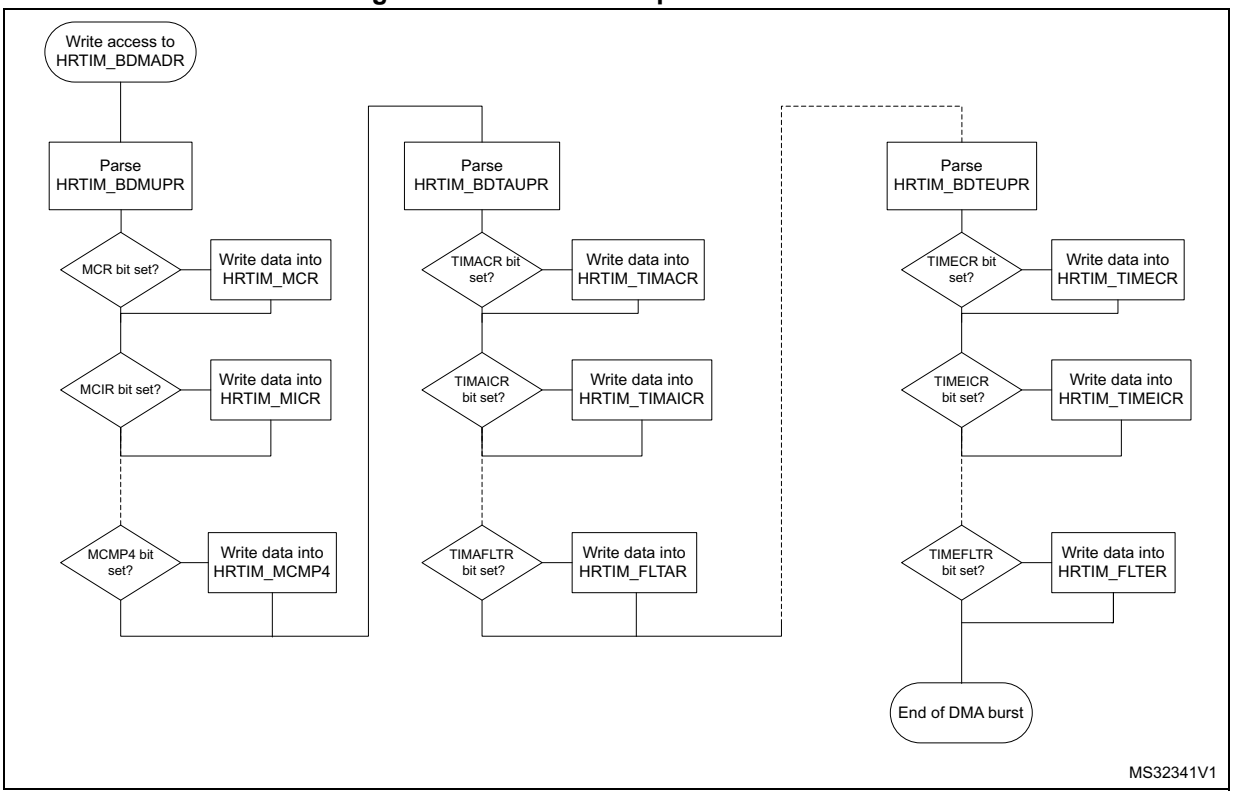
Note: Any trigger occurring while the burst is on-going is discarded, except if it occurs during the very last data transfer.

The burst DMA mode is permanently enabled (there is no enable bit). A burst DMA operation is started by the first write access into the HRTIM_BDMADR register.

It is only necessary to have the DMA controller pointing to the HRTIM_BDMADR register as the destination, in the memory, to the peripheral configuration with the peripheral increment mode disabled (the HRTIM handles internally the data re-routing to the final destination register).

To re-initialize the burst DMA mode if it was interrupted during a transaction, it is necessary to write at least to one of the 6 update registers.

Figure 333. Burst DMA operation flowchart



Several options are available once the DMA burst is completed, depending on the register update strategy.

If the PREEN bit is reset (preload disabled), the value written by the DMA is immediately transferred into the active register and the registers are updated sequentially, following the DMA transaction pace.

When the preload is enabled (PREEN bit set), there are 3 use cases:

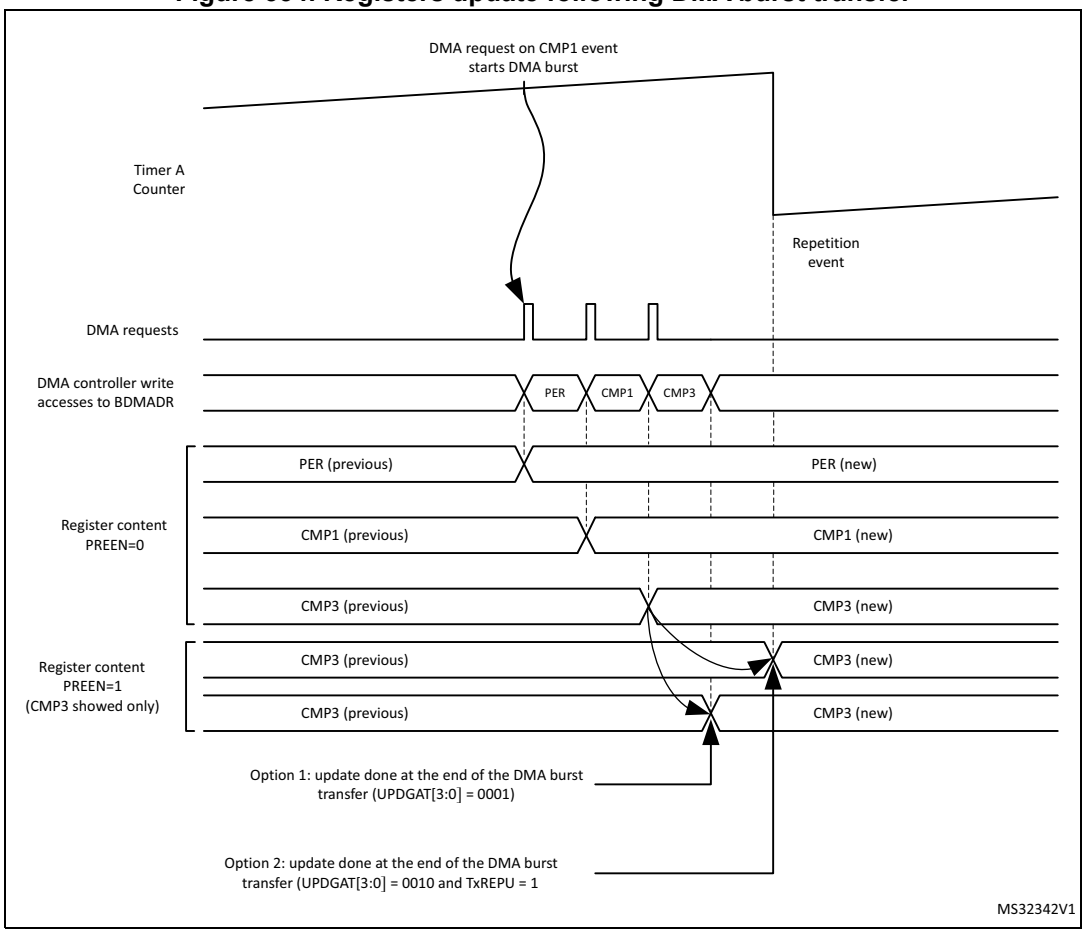
1. The update is done independently from DMA burst transfers (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 00 in HRTIM_MCR). In this case, and if it is necessary to have all transferred data taken into account simultaneously, the user must check that the DMA burst is completed before the update event takes place. On the contrary, if the update event happens while the DMA transfer is on-going, only part of the registers will be loaded and the complete register update will require 2 consecutive update events.
2. The update is done when the DMA burst transfer is completed (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 01 in HRTIM_MCR). This mode guarantees that all new register values are transferred simultaneously. This is done independently

from the counter value and can be combined with regular update events, if necessary (for instance, an update on a counter reset when TxRSTU is set).

3. The update is done on the update event following the DMA burst transfer completion (UPDGAT[3:0] = 0010 in HRTIM_TIMxCR and BRSTDMA[1:0] = 10 in HRTIM_MCR). This mode guarantees both a coherent update of all transferred data and the synchronization with regular update events, with the timer counter. In this case, if a regular update request occurs while the transfer is on-going, it will be discarded and the effective update will happen on the next coming update request.

The chronogram on *Figure 334* presents the active register content for 3 cases: PREEN=0, UPDGAT[3:0] = 0001 and UPDGAT[3:0] = 0001 (when PREEN = 1).

Figure 334. Registers update following DMA burst transfer



36.3.22 HRTIM initialization

This section describes the recommended HRTIM initialization procedure, including other related MCU peripherals.

The HRTIM clock source must be enabled in the Reset and Clock control unit (RCC).

The HRTIM control registers can be initialized as per the power converter topology and the timing units use case. All inputs have to be configured (source, polarity, edge-sensitivity).

The HRTIM outputs must be set up eventually, with the following sequence:

- the polarity must be defined using POLx bits in HRTIM_OUTxR
- the FAULT and IDLE states must be configured using FAULTx[1:0] and IDLESx bits in HRTIM_OUTxR

The HRTIM outputs are ready to be connected to the MCU I/Os. In the GPIO controller, the selected HRTIM I/Os have to be configured as per the alternate function mapping table in the product datasheet.

From this point on, the HRTIM controls the outputs, which are in the IDLE state.

The outputs are configured in RUN mode by setting TxyOEN bits in the HRTIM_OENR register. The 2 outputs are in the inactive state until the first valid set/reset event in RUN mode. Any output set/reset event (except software requests using SST, SRT) are ignored as long as TxCEN bit is reset, as well as burst mode requests (IDLEM bit value is ignored). Similarly, any counter reset request coming from the burst mode controller is ignored (if TxBM bit is set).

Note: When the deadtime insertion is enabled (DTEN bit set), it is necessary to force the output state by software, using SST and RST bits, to have the outputs in a complementary state as soon as the RUN mode is entered.

The HRTIM operation can eventually be started by setting TxCEN or MCEN bits in HRTIM_MCR.

If the HRTIM peripheral is reset with the Reset and Clock Controller, the HRTIM outputs are put in IDLE mode with a low level. It is recommended to first disconnect the HRTIMER from the outputs (using the GPIO controller) before performing a peripheral reset.

36.3.23 Debug

When a microcontroller enters the debug mode (Cortex®-M7 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_HRTIM_STOP configuration bit in DBG module:

- DBG_HRTIM_STOP = 0: no behavior change, the HRTIM continues to operate.
- DBG_HRTIM_STOP = 1: all HRTIM timers, including the master, are stopped. The outputs in RUN mode enter the FAULT state if FAULTx[1:0] = 01,10,11, or keep their current state if FAULTx[1:0] = 00. The outputs in idle state are maintained in this state. This is permanently maintained even if the MCU exits the halt mode. This allows to maintain a safe state during the execution stepping. The outputs can be enabled again by settings TxyOEN bit (requires the use of the debugger).

Timer behavior during MCU halt when DBG_HRTIM_STOP = 1

The set/reset crossbar, the dead-time and push-pull unit, the idle/balanced fault detection and all the logic driving the normal output in RUN mode are not affected by debug. The output will keep on toggling internally, so as to retrieve regular signals of the outputs when TxyOEN will be set again (during or after the MCU halt). Associated triggers and filters are also following internal waveforms when the outputs are disabled.

FAULT inputs and events (any source) are enabled during the MCU halt.

Fault status bits can be set and TxyOEN bits reset during the MCU halt if a fault occurs at that time (TxyOEN and TxyODS are not affected by DBG_HRTIM_STOP bit state).

Synchronization, counter reset, start and reset-start events are discarded in debug mode, as well as capture events. This is to keep all related registers stable as long as the MCU is halted.

The counter stops counting when a breakpoint is reached. However, the counter enable signal is not reset; consequently no start event will be emitted when exiting from debug. All counter reset and capture triggers are disabled, as well as external events (ignored as long as the MCU is halted). The outputs SET and RST flags are frozen, except in case of forced software set/reset. A level-sensitive event is masked during the debug but will be active again as soon as the debug will be exited. For edge-sensitive events, if the signal is maintained active during the MCU halt, a new edge is not generated when exiting from debug.

The update events are discarded. This prevents any update trigger on hrtim_upd_en[3:1] inputs. DMA triggers are disabled. The burst mode circuit is frozen: the triggers are ignored and the burst mode counter stopped.

36.4 Application use cases

36.4.1 Buck converter

Buck converters are of common use as step-down converters. The HRTIM can control up to 10 buck converters with 6 independent switching frequencies.

The converter usually operates at a fixed frequency and the Vin/Vout ratio depends on the duty cycle D applied to the power switch:.

$$V_{out} = D \times V_{in}$$

The topology is given on *Figure 335* with the connection to the ADC for voltage reading.

Figure 335. Buck converter topology

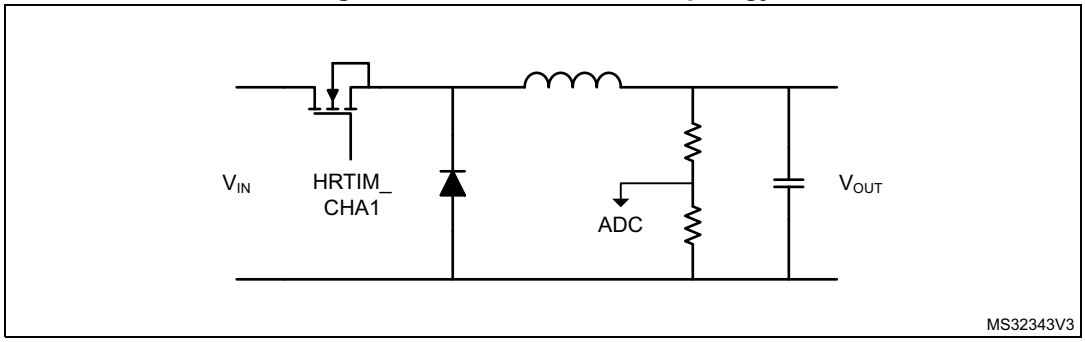
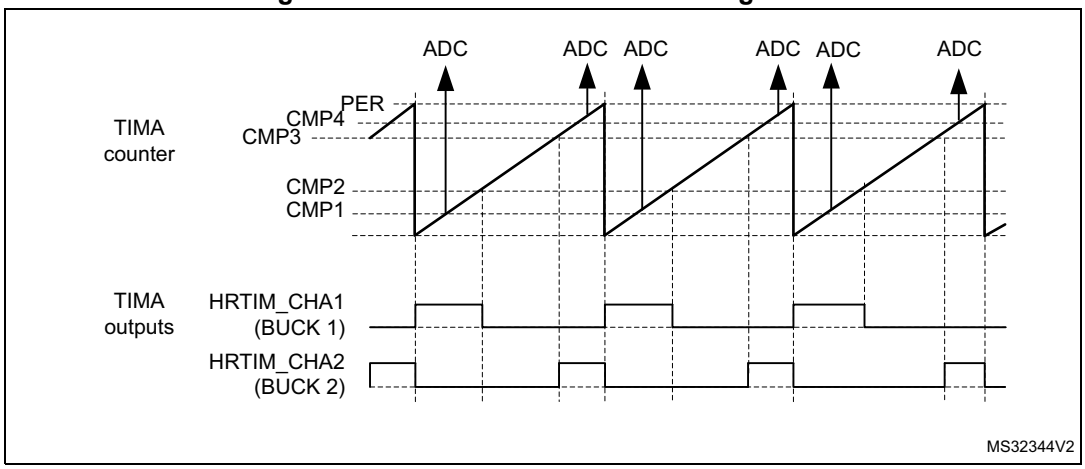


Figure 336 presents the management of two converters with identical frequency PWM signals. The outputs are defined as follows:

- HRTIM_CHA1 set on period, reset on CMP1
- HRTIM_CHA2 set on CMP3, reset on PER

The ADC is triggered twice per period, precisely in the middle of the ON time, using CMP2 and CMP4 events.

Figure 336. Dual Buck converter management

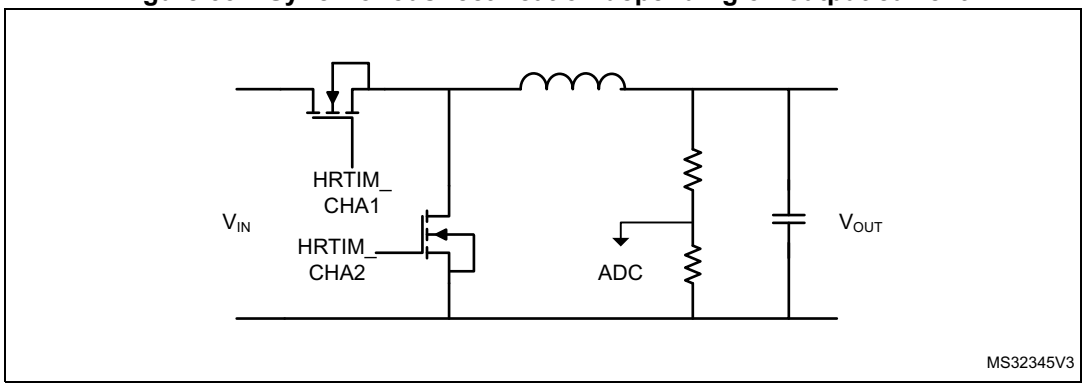


Timers A..E provide either 10 buck converters coupled by pairs (both with identical switching frequencies) or 6 completely independent converters (each of them having a different switching frequency), using the master timer as the 6th time base.

36.4.2 Buck converter with synchronous rectification

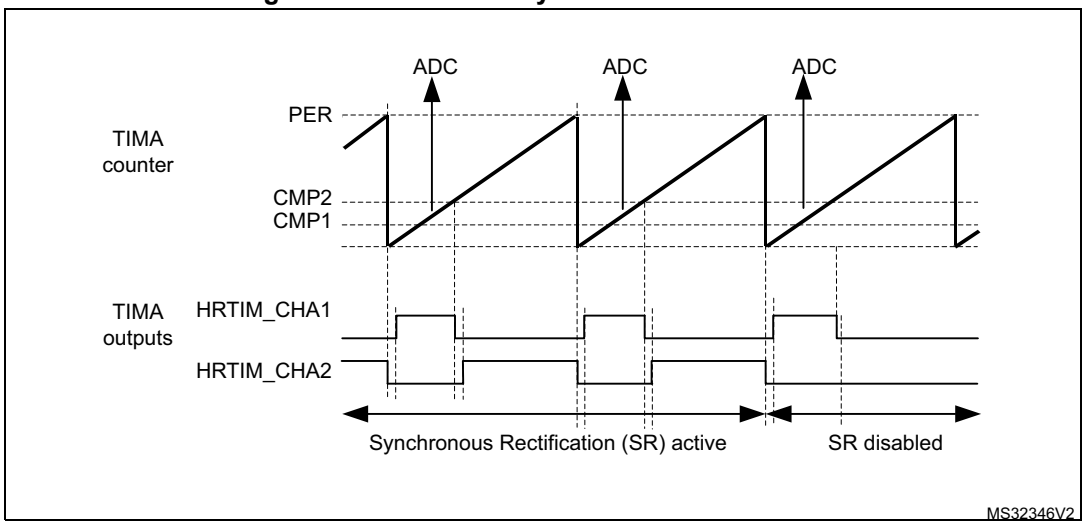
Synchronous rectification allows to minimize losses in buck converters, by means of a FET replacing the freewheeling diode. Synchronous rectification can be turned on or off on the fly depending on the output current level, as shown on *Figure 337*.

Figure 337. Synchronous rectification depending on output current



The main difference vs. a single-switch buck converter is the addition of a deadtime for an almost complementary waveform generation on HRTIM_CHA2, based on the reference waveform on HRTIM_CHA1 (see *Figure 338*).

Figure 338. Buck with synchronous rectification



36.4.3 Multiphase converters

Multiphase techniques can be applied to multiple power conversion topologies (buck, flyback). Their main benefits are:

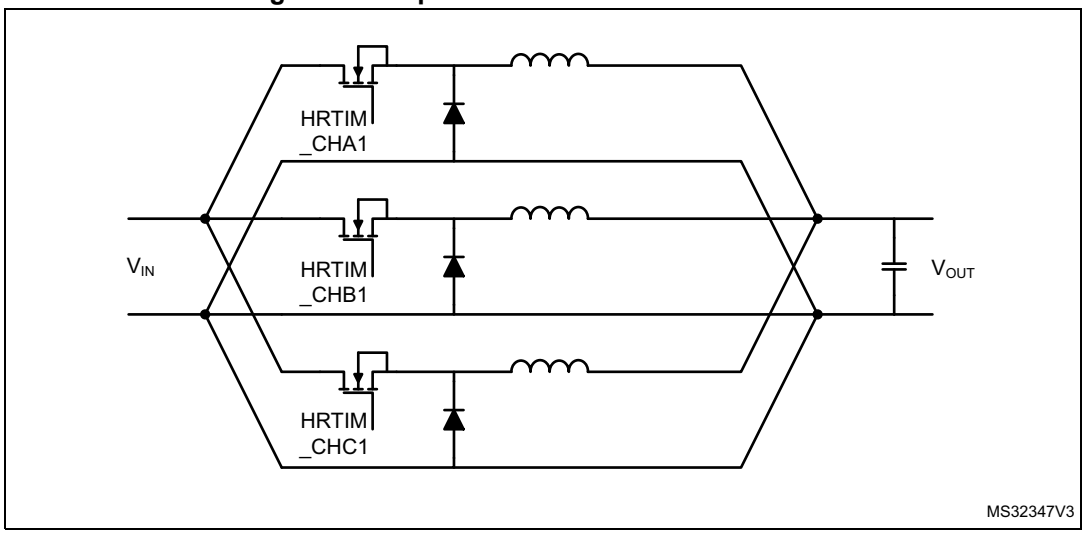
- Reduction of the current ripple on the input and output capacitors
- Reduced EMI
- Higher efficiency at light load by dynamically changing the number of phases (phase shedding)

The HRTIM is able to manage multiple converters. The number of converters that can be controlled depends on the topologies and resources used (including the ADC triggers):

- 5 buck converters with synchronous rectification (SR), using the master timer and the 5 timers
- 4 buck converters (without SR), using the master timer and 2 timers
- ...

Figure 340 presents the topology of a 3-phase interleaved buck converter.

Figure 339. 3-phase interleaved buck converter



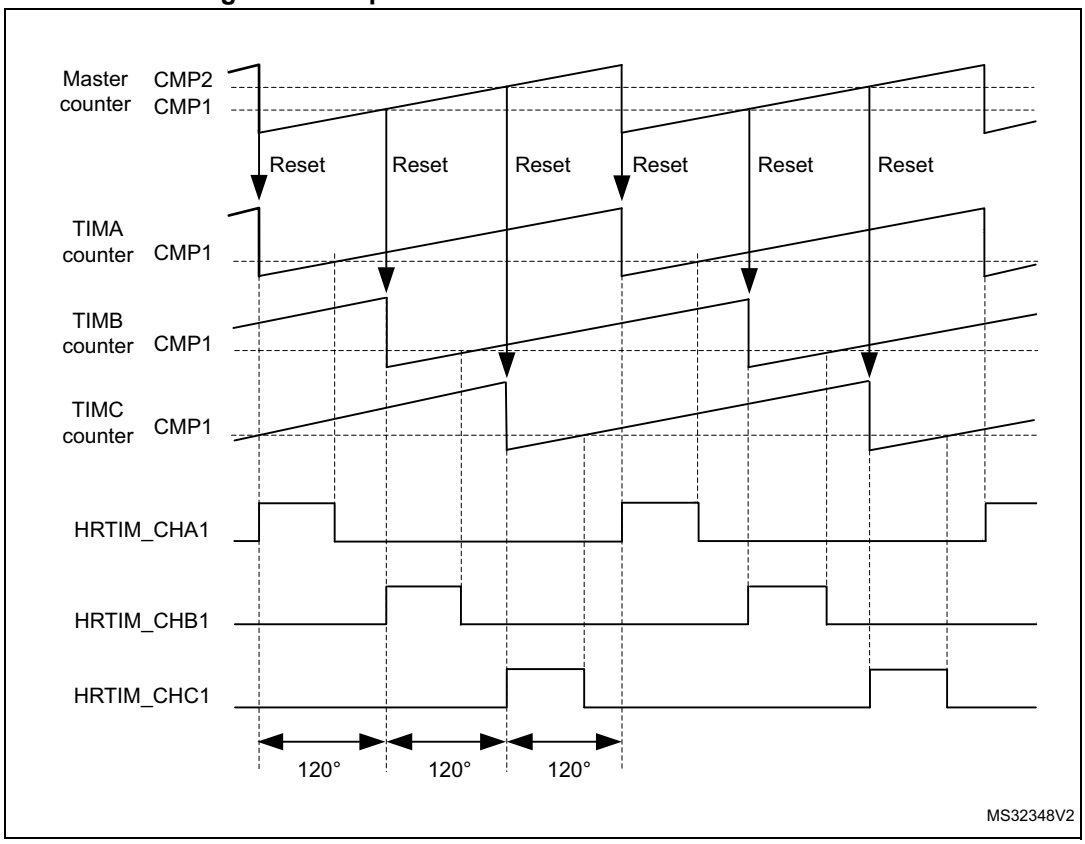
The master timer is responsible for the phase management: it defines the phase relationship between the converters by resetting the timers periodically. The phase-shift is 360° divided by the number of phases, 120° in the given example.

The duty cycle is then programmed into each of the timers. The outputs are defined as follows:

- HRTIM_CHA1 set on master timer period, reset on TACMP1
- HRTIM_CHB1 set on master timer MCMP1, reset on TBCMP1
- HRTIM_CHC1 set on master timer MCMP2, reset on TCCMP1

The ADC trigger can be generated on TxCMP2 compare event. Since all ADC trigger sources are phase-shifted because of the converter topology, it is possible to have all of them combined into a single ADC trigger to save ADC resources (for instance 1 ADC regular channel for the full multi-phase converter).

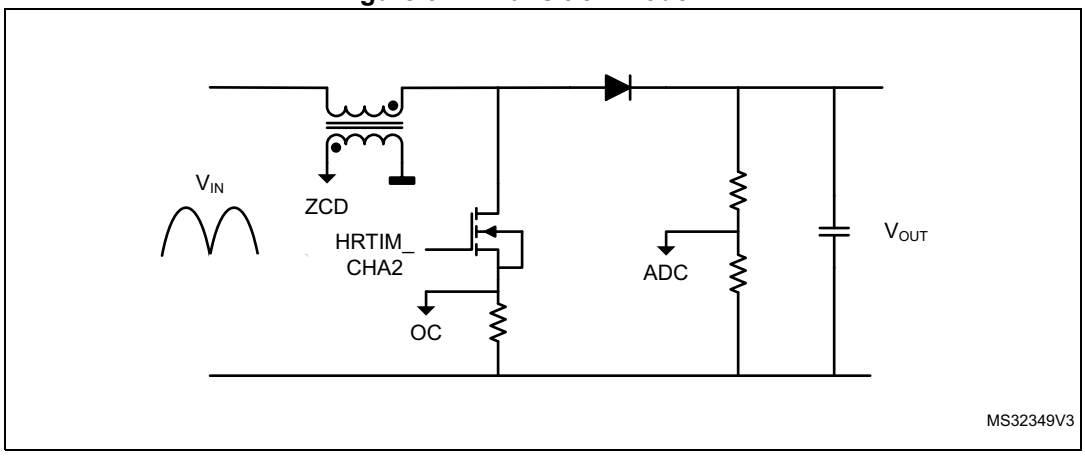
Figure 340. 3-phase interleaved buck converter control



36.4.4 Transition mode Power Factor Correction

The basic operating principle is to build up current into an inductor during a fixed Ton time. This current will then decay during the Toff time, and the period will be re-started when it becomes null. This is detected using a Zero Crossing Detection circuitry (ZCD), as shown on *Figure 341*. With a constant Ton time, the peak current value in the inductor is proportional to the rectified AC input voltage, which provides the power factor correction.

Figure 341. Transition mode PFC



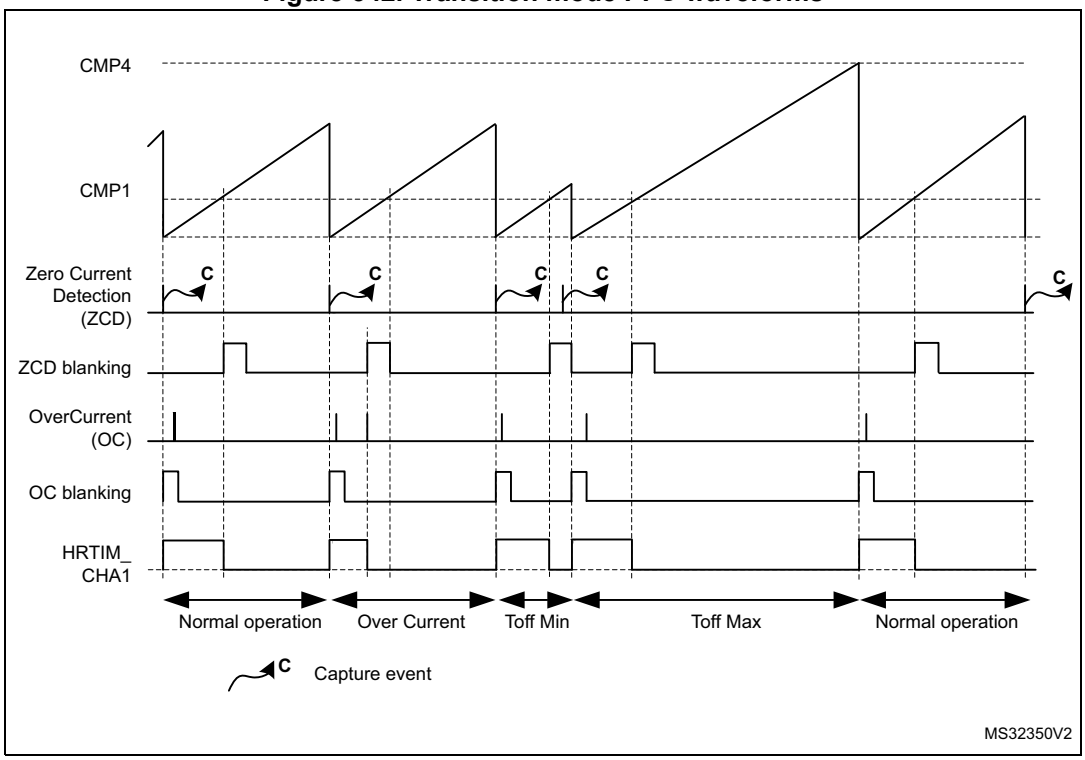
This converter is operating with a constant Ton time and a variable frequency due the Toff time variation (depending on the input voltage). It must also include some features to operate when no zero-crossing is detected, or to limit the Ton time in case of over-current (OC). The OC feedback is usually conditioned with the built-in comparator and routed onto an external event input.

Figure 342 presents the waveform during the various operating modes, with the following parameters defined:

- Ton Min: masks spurious overcurrent (freewheeling diode recovery current), represented as OC blanking
- Ton Max: practically, the converter set-point. It is defined by CMP1
- Toff Min: limits the frequency when the current limit is close to zero (demagnetization is very fast). It is defined with CMP2.
- Toff Max: prevents the system to be stuck if no ZCD occurs. It is defined with CMP4 in auto-delayed mode.

Both Toff values are auto-delayed since the value must be relative to the output falling edge.

Figure 342. Transition mode PFC waveforms



36.5 HRTIM registers

36.5.1 HRTIM Master Timer Control Register (HRTIM_MCR)

Address offset: 0x0000h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	Res.	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN
rw	rw	rw		rw	rw	rw				rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		Res.	Res.	HALF	RETRIG	CONT	CKPSC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 **BRSTDMA[1:0]**: *Burst DMA Update*

These bits define how the update occurs relatively to a burst DMA transaction.
00: Update done independently from the DMA burst transfer completion
01: Update done when the DMA burst transfer is completed
10: Update done on master timer roll-over following a DMA burst transfer completion. This mode only works in continuous mode.
11: reserved

Bit 29 **MREPU**: Master *Timer Repetition update*

This bit defines whether an update occurs when the master timer repetition period is completed (either due to roll-over or reset events). MREPU can be set only if BRSTDMA[1:0] = 00 or 01.
0: Update on repetition disabled
1: Update on repetition enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **PREEN**: *Preload enable*

This bit enables the registers preload mechanism and defines whether the write accesses to the memory mapped registers are done into HRTIM active or preload registers.
0: Preload disabled: the write access is directly done into the active register
1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** *DAC Synchronization*

A DAC synchronization event can be enabled and generated when the master timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 36.3.19: DAC triggers* for connections details).
00: No DAC trigger generated
01: Trigger generated on hrtim_dac_trg1
10: Trigger generated on hrtim_dac_trg2
11: Trigger generated on hrtim_dac_trg3

Bits 24:22 Reserved, must be kept at reset value.

Bit 21 **TECEN**: *Timer E counter enable*

This bit starts the Timer E counter.
0: Timer E counter disabled
1: Timer E counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 20 **TDCEN**: *Timer D counter enable*
This bit starts the Timer D counter.
0: Timer D counter disabled
1: Timer D counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 19 **TCCEN**: *Timer C counter enable*
This bit starts the Timer C counter.
0: Timer C counter disabled
1: Timer C counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 18 **TBCEN**: *Timer B counter enable*
This bit starts the Timer B counter.
0: Timer B counter disabled
1: Timer B counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 17 **TACEN**: *Timer A counter enable*
This bit starts the Timer A counter.
0: Timer A counter disabled
1: Timer A counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 16 **MCEN**: *Master timer counter enable*
This bit starts the Master timer counter.
0: Master counter disabled
1: Master counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bits 15:14 **SYNCSRC[1:0]**: *Synchronization source*
These bits define the source and event to be sent on the synchronization outputs SYNCOUT[2:1]
00: Master timer Start
01: Master timer Compare 1 event
10: Timer A start/reset
11: Timer A Compare 1 event

Bits 13:12 **SYNCOUT[1:0]**: *Synchronization output*
These bits define the routing and conditioning of the synchronization output event.
00: disabled
01: Reserved.
10: Positive pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
11: Negative pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTM**: *Synchronization Starts Master*
This bit enables the Master timer start when receiving a synchronization input event:
0: No effect on the Master timer
1: A synchronization input event starts the Master timer

Bit 10 **SYNCRSTM**: *Synchronization Resets Master*
This bit enables the Master timer reset when receiving a synchronization input event:
0: No effect on the Master timer
1: A synchronization input event resets the Master timer

Bits 9:8 **SYNCIN[1:0]** *Synchronization input*

These bits are defining the synchronization input source.
00: disabled. HRTIM is not synchronized and runs in standalone mode.
01: Reserved.
10: Internal event: the HRTIM is synchronized with the on-chip timer (see *Synchronization input*).
11: External event (input pin). A positive pulse on HRTIM_SCIN input triggers the HRTIM.

Note: This parameter cannot be changed once the impacted timers are enabled.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **HALF**: *Half mode*

This bit enables the half duty-cycle mode: the HRTIM_MCMP1xR active register is automatically updated with HRTIM_MPER/2 value when HRTIM_MPER register is written.
0: Half mode disabled
1: Half mode enabled

Bit 4 **RETRIG**: *Re-triggerable mode*

This bit defines the behavior of the master timer counter in single-shot mode.
0: The timer is not re-triggerable: a counter reset can be done only if the counter is stopped (period elapsed)
1: The timer is re-triggerable: a counter reset is done whatever the counter state (running or stopped)

Bit 3 **CONT**: *Continuous mode*

0: The timer operates in single-shot mode and stops when it reaches the MPER value
1: The timer operates in continuous (free-running) mode and rolls over to zero when it reaches the MPER value

Bits 2:0 **CKPSC[2:0]**: *Clock prescaler*

These bits define the master timer clock prescaler ratio.
The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{(CKPSC[2:0]-5)}$.
The prescaling ratio cannot be modified once the timer is enabled.
000: Reserved
001: Reserved
010: Reserved
011: Reserved
100: Reserved
101: $f_{COUNTER} = f_{HRTIM}$
110: $f_{COUNTER} = f_{HRTIM} / 2$
111: $f_{COUNTER} = f_{HRTIM} / 4$

36.5.2 HRTIM Master Timer Interrupt Status Register (HRTIM_MISR)

Address offset: 0x0004h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP4	MCMP3	MCMP2	MCMP1
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPD**: Master Update Interrupt Flag
This bit is set by hardware when the Master timer registers are updated.
0: No Master Update interrupt occurred
1: Master Update interrupt occurred

Bit 5 **SYNC**: Sync Input Interrupt Flag
This bit is set by hardware when a synchronization input event is received.
0: No Sync input interrupt occurred
1: Sync input interrupt occurred

Bit 4 **MREP**: Master Repetition Interrupt Flag
This bit is set by hardware when the Master timer repetition period has elapsed.
0: No Master Repetition interrupt occurred
1: Master Repetition interrupt occurred

Bit 3 **MCMP4**: Master Compare 4 Interrupt Flag
Refer to MCMP1 description

Bit 2 **MCMP3**: Master Compare 3 Interrupt Flag
Refer to MCMP1 description

Bit 1 **MCMP2**: Master Compare 2 Interrupt Flag
Refer to MCMP1 description

Bit 0 **MCMP1**: Master Compare 1 Interrupt Flag
This bit is set by hardware when the Master timer counter matches the value programmed in the master Compare 1 register.
0: No Master Compare 1 interrupt occurred
1: Master Compare 1 interrupt occurred

36.5.3 HRTIM Master Timer Interrupt Clear Register (HRTIM_MICR)

Address offset: 0x0008h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPDC**: Master update Interrupt flag clear
Writing 1 to this bit clears the MUPDC flag in HRTIM_MISR register

Bit 5 **SYNCC**: Sync Input Interrupt flag clear
Writing 1 to this bit clears the SYNC flag in HRTIM_MISR register

Bit 4 **MREPC**: Repetition Interrupt flag clear
Writing 1 to this bit clears the MREP flag in HRTIM_MISR register

Bit 3 **MCMP4C**: Master Compare 4 Interrupt flag clear
Writing 1 to this bit clears the MCMP4 flag in HRTIM_MISR register

Bit 2 **MCMP3C**: Master Compare 3 Interrupt flag clear
Writing 1 to this bit clears the MCMP3 flag in HRTIM_MISR register

Bit 1 **MCMP2C**: Master Compare 2 Interrupt flag clear
Writing 1 to this bit clears the MCMP2 flag in HRTIM_MISR register

Bit 0 **MCMP1C**: Master Compare 1 Interrupt flag clear
Writing 1 to this bit clears the MCMP1 flag in HRTIM_MISR register

36.5.4 HRTIM Master Timer DMA / Interrupt Enable Register (HRTIM_MDIER)

Address offset: 0x000Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **MUPDDE**: Master Update DMA request Enable
This bit is set and cleared by software to enable/disable the Master update DMA requests.
0: Master update DMA request disabled
1: Master update DMA request enabled

Bit 21 **SYNCDE**: Sync Input DMA request Enable
This bit is set and cleared by software to enable/disable the Sync input DMA requests.
0: Sync input DMA request disabled
1: Sync input DMA request enabled

Bit 20 **MREPDE**: Master Repetition DMA request Enable
This bit is set and cleared by software to enable/disable the Master timer repetition DMA requests.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **MCMP4DE**: Master Compare 4 DMA request Enable
Refer to MCMP1DE description

Bit 18 **MCMP3DE**: Master Compare 3 DMA request Enable
Refer to MCMP1DE description

Bit 17 **MCMP2DE**: Master Compare 2 DMA request Enable
Refer to MCMP1DE description

Bit 16 **MCMP1DE**: Master Compare 1 DMA request Enable
This bit is set and cleared by software to enable/disable the Master timer Compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bits 15:6 Reserved, must be kept at reset value.

Bit 6 **MUPDIE**: Master Update Interrupt Enable
This bit is set and cleared by software to enable/disable the Master timer registers update interrupts
0: Master update interrupts disabled
1: Master update interrupts enabled

Bit 5 **SYNCIE**: Sync Input Interrupt Enable
This bit is set and cleared by software to enable/disable the Sync input interrupts
0: Sync input interrupts disabled
1: Sync input interrupts enabled

Bit 4 **MREPIE**: Master Repetition Interrupt Enable
This bit is set and cleared by software to enable/disable the Master timer repetition interrupts
0: Master repetition interrupt disabled
1: Master repetition interrupt enabled

Bit 3 **MCMP4IE**: Master Compare 4 Interrupt Enable
Refer to MCMP1IE description

Bit 2 **MCMP3IE**: Master Compare 3 Interrupt Enable
Refer to MCMP1IE description

Bit 1 **MCMP2IE**: MAster Compare 2 Interrupt Enable
Refer to MCMP1IE description

Bit 0 **MCMP1IE**: Master Compare 1 Interrupt Enable
This bit is set and cleared by software to enable/disable the Master timer Compare 1 interrupt
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

36.5.5 HRTIM Master Timer Counter Register (HRTIM_MCNTR)

Address offset: 0x0010h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCNT[15:0]**: *Counter value*

Holds the master timer counter value. This register can only be written when the master timer is stopped (MCEN = 0 in HRTIM_MCR).

Note: The timer behavior is not guaranteed if the counter value is set above the HRTIM_MPER register value.

36.5.6 HRTIM Master Timer Period Register (HRTIM_MPER)

Address offset: 0x0014h

Reset value: 0x0000 FFDF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MPER[15:0]**: *Master Timer Period value*

This register defines the counter overflow value.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock.

The maximum value is 0x0000 FFDF.

36.5.7 HRTIM Master Timer Repetition Register (HRTIM_MREP)

Address offset: 0x0018h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MREP[7:0]**: *Master Timer Repetition period value*

This register holds the repetition period value for the master counter. It is either the preload register or the active register if preload is disabled.

36.5.8 HRTIM Master Timer Compare 1 Register (HRTIM_MCMP1R)

Address offset: 0x001Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP1[15:0]**: *Master Timer Compare 1 value*

This register holds the master timer Compare 1 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

36.5.9 HRTIM Master Timer Compare 2 Register (HRTIM_MCMP2R)

Address offset: 0x0024h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP2[15:0]**: *Master Timer Compare 2 value*

This register holds the master timer Compare 2 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

36.5.10 HRTIM Master Timer Compare 3 Register (HRTIM_MCMP3R)

Address offset: 0x0028h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP3[15:0]**: *Master Timer Compare 3 value*

This register holds the master timer Compare 3 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

36.5.11 HRTIM Master Timer Compare 4 Register (HRTIM_MCMP4R)

Address offset: 0x002Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP4[15:0]**: *Master Timer Compare 4 value*

This register holds the master timer Compare 4 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

36.5.12 HRTIM Timerx Control Register (HRTIM_TIMxCR)

Address offset: 0x0000h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDGAT[3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	Res.	TxRSTU	TxREPU	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	Res.	Res.	Res.	PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **UPDGAT[3:0]**: *Update Gating*

These bits define how the update occurs relatively to the burst DMA transaction and the external update request on update enable inputs 1 to 3 (see *Table 303: Update enable inputs and sources*)
The update events, as mentioned below, can be: MSTU, TEU, TDU, TCU, TBU, TAU, TxRSTU, TxREPU.
0000: the update occurs independently from the DMA burst transfer
0001: the update occurs when the DMA burst transfer is completed
0010: the update occurs on the update event following the DMA burst transfer completion
0011: the update occurs on a rising edge of HRTIM update enable input 1 (hrtim_upd_en1)
0100: the update occurs on a rising edge of HRTIM update enable input 2 (hrtim_upd_en2)
0101: the update occurs on a rising edge of HRTIM update enable input 3 (hrtim_upd_en3)
0110: the update occurs on the update event following a rising edge of HRTIM update enable input 1 (hrtim_upd_en1)
0111: the update occurs on the update event following a rising edge of HRTIM update enable input 2 (hrtim_upd_en2)
1000: the update occurs on the update event following a rising edge of HRTIM update enable input 3 (hrtim_upd_en3)
Other codes: reserved

Note: This bitfield must be reset before programming a new value.
For UPDGAT[3:0] values equal to 0001, 0011, 0100, 0101, it is possible to have multiple concurrent update source (for instance RSTU and DMA burst).

Bit 27 **PREEN**: *Preload enable*

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register is done into the active or the preload register.
0: Preload disabled: the write access is directly done into the active register
1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]** *DAC Synchronization*

A DAC synchronization event is generated when the timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 36.3.19: DAC triggers* for connections details).
00: No DAC trigger generated
01: Trigger generated on hrtim_dac_trg1
10: Trigger generated on hrtim_dac_trg2
11: Trigger generated on hrtim_dac_trg3

Bit 24 **MSTU**: *Master Timer update*
Register update is triggered by the master timer update.
0: Update by master timer disabled
1: Update by master timer enabled

Bit 23 In HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR:
TEU: *Timer E update*
Register update is triggered by the timer E update
0: Update by timer E disabled
1: Update by timer E enabled

In HRTIM_TIMECR:
Reserved, must be kept at reset value

Bit 22 In HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMECR:
TDU: *Timer D update*
Register update is triggered by the timer D update
0: Update by timer D disabled
1: Update by timer D enabled

In HRTIM_TIMDCR:
Reserved, must be kept at reset value

Bit 21 In HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMDCR, HRTIM_TIMECR:
TCU: *Timer C update*
Register update is triggered by the timer C update
0: Update by timer C disabled
1: Update by timer C enabled

In HRTIM_TIMCCR:
Reserved, must be kept at reset value

Bit 20 In HRTIM_TIMACR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR:
TBU: *Timer B update*
Register update is triggered by the timer B update
0: Update by timer B disabled
1: Update by timer B enabled

In HRTIM_TIMBCR:
Reserved, must be kept at reset value

Bit 19 In HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR:
TAU: *Timer A update*
Register update is triggered by the timer A update
0: Update by timer A disabled
1: Update by timer A enabled

In HRTIM_TIMACR:
Reserved, must be kept at reset value

Bit 18 **TxRSTU**: *Timerx reset update*

Register update is triggered by Timerx counter reset or roll-over to 0 after reaching the period value in continuous mode.

0: Update by timer x reset / roll-over disabled

1: Update by timer x reset / roll-over enabled

Bit 17 **TxREPU**: *Timer x Repetition update*

Register update is triggered when the counter rolls over and HRTIM_REPx = 0

0: Update on repetition disabled

1: Update on repetition enabled

Bit 16 Reserved, must be kept at reset value.

Bits 15:14 **DELCMP4[1:0]**: *CMP4 auto-delayed mode*

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Auto-delayed mode*).

00: CMP4 register is always active (standard compare mode)

01: CMP4 value is recomputed and is active following a capture 2 event

10: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 1 match (timeout function if capture 2 event is missing)

11: CMP4 value is recomputed and is active following a capture event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bits 13:12 **DELCMP2[1:0]**: *CMP2 auto-delayed mode*

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Auto-delayed mode*).

00: CMP2 register is always active (standard compare mode)

01: CMP2 value is recomputed and is active following a capture 1 event

10: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after Compare 1 match (timeout function if capture event is missing)

11: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTx**: *Synchronization Starts Timer x*

This bit defines the Timer x behavior following the synchronization event:

0: No effect on Timer x

1: A synchronization input event starts the Timer x

Bit 10 **SYNCRSTx**: *Synchronization Resets Timer x*

This bit defines the Timer x behavior following the synchronization event:

0: No effect on Timer x

1: A synchronization input event resets the Timer x

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **PSHPLL**: *Push-Pull mode enable*

This bit enables the push-pull mode.

0: Push-Pull mode disabled

1: Push-Pull mode enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 5 **HALF**: *Half mode enable*

This bit enables the half duty-cycle mode: the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/2 value when HRTIM_PERxR register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: *Re-triggerable mode*

This bit defines the counter behavior in single shot mode.

0: The timer is not re-triggerable: a counter reset is done if the counter is stopped (period elapsed in single-shot mode or counter stopped in continuous mode)

1: The timer is re-triggerable: a counter reset is done whatever the counter state.

Bit 3 **CONT**: *Continuous mode*

This bit defines the timer operating mode.

0: The timer operates in single-shot mode and stops when it reaches TIMxPER value

1: The timer operates in continuous mode and rolls over to zero when it reaches TIMxPER value

Bits 2:0 **CKPSCx[2:0]**: *HRTIM Timer x Clock prescaler*

These bits define the master timer clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{(CKPSC[2:0]-5)}$.

The prescaling ratio cannot be modified once the timer is enabled.

000: Reserved

001: Reserved

010: Reserved

011: Reserved

100: Reserved

101: $f_{COUNTER} = f_{HRTIM}$

110: $f_{COUNTER} = f_{HRTIM} / 2$

111: $f_{COUNTER} = f_{HRTIM} / 4$

36.5.13 HRTIM Timerx Interrupt Status Register (HRTIM_TIMxISR)

Address offset: 0x0004h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	Res.	REP	CMP4	CMP3	CMP2	CMP1
	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **O2CPY**: Output 2 Copy

This status bit is a raw copy of the output 2 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 2 is inactive

1: Output 2 is active

Bit 20 **O1CPY**: Output 1 Copy

This status bit is a raw copy of the output 1 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 1 is inactive

1: Output 1 is active

Bit 19 **O2STAT**: Output 2 Status

This status bit indicates the output 2 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 2 was inactive

1: Output 2 was active

Bit 18 **O1STAT**: Output 1 Status

This status bit indicates the output 1 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 1 was inactive

1: Output 1 was active

Bit 17 **IPPSTAT**: Idle Push Pull Status

This status bit indicates on which output the signal was applied, in push-pull mode balanced fault mode or delayed idle mode, when the protection was triggered (whatever the output state, active or inactive).

0: Protection occurred when the output 1 was active and output 2 forced inactive

1: Protection occurred when the output 2 was active and output 1 forced inactive

Bit 16 **CPPSTAT**: Current Push Pull Status

This status bit indicates on which output the signal is currently applied, in push-pull mode. It is only significant in this configuration.

0: Signal applied on output 1 and output 2 forced inactive

1: Signal applied on output 2 and output 1 forced inactive

Bit 15 Reserved

Bit 14 **DLYPRT**: Delayed Protection Flag

This bit indicates delayed idle or the balanced idle mode entry.

Bit 13 **RST**: Reset and/or roll-over Interrupt Flag
This bit is set by hardware when the timer x counter is reset or rolls over in continuous mode.
0: No TIMx counter reset/roll-over interrupt occurred
1: TIMX counter reset/roll-over interrupt occurred

Bit 12 **RSTx2**: Output 2 Reset Interrupt Flag
Refer to RSTx1 description

Bit 11 **SETx2**: Output 2 Set Interrupt Flag
Refer to SETx1 description

Bit 10 **RSTx1**: Output 1 Reset Interrupt Flag
This bit is set by hardware when the Tx1 output is reset (goes from active to inactive mode).
0: No Tx1 output reset interrupt occurred
1: Tx1 output reset interrupt occurred

Bit 9 **SETx1**: Output 1 Set Interrupt Flag
This bit is set by hardware when the Tx1 output is set (goes from inactive to active mode).
0: No Tx1 output set interrupt occurred
1: Tx1 output set interrupt occurred

Bit 8 **CPT2**: Capture2 Interrupt Flag
Refer to CPT1 description

Bit 7 **CPT1**: Capture1 Interrupt Flag
This bit is set by hardware when the timer x capture 1 event occurs.
0: No timer x Capture 1 reset interrupt occurred
1: Timer x output 1 reset interrupt occurred

Bit 6 **UPD**: Update Interrupt Flag
This bit is set by hardware when the timer x update event occurs.
0: No timer x update interrupt occurred
1: Timer x update interrupt occurred

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REP**: Repetition Interrupt Flag
This bit is set by hardware when the timer x repetition period has elapsed.
0: No timer x repetition interrupt occurred
1: Timer x repetition interrupt occurred

Bit 3 **CMP4**: Compare 4 Interrupt Flag
Refer to CMP1 description

Bit 2 **CMP3**: Compare 3 Interrupt Flag
Refer to CMP1 description

Bit 1 **CMP2**: Compare 2 Interrupt Flag
Refer to CMP1 description

Bit 0 **CMP1**: Compare 1 Interrupt Flag
This bit is set by hardware when the timer x counter matches the value programmed in the Compare 1 register.
0: No Compare 1 interrupt occurred
1: Compare 1 interrupt occurred

36.5.14 HRTIM Timerx Interrupt Clear Register (HRTIM_TIMxICR)

Address offset: 0x0008h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRTC	RSTC	RSTx2C	SET2xC	RSTx1C	SET1xC	CPT2C	CPT1C	UPDC	Res.	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **DLYPRTC**: Delayed Protection Flag Clear
Writing 1 to this bit clears the DLYPRT flag in HRTIM_TIMxISR register

Bit 13 **RSTC**: Reset Interrupt flag Clear
Writing 1 to this bit clears the RST flag in HRTIM_TIMxISR register

Bit 12 **RSTx2C**: Output 2 Reset flag Clear
Writing 1 to this bit clears the RSTx2 flag in HRTIM_TIMxISR register

Bit 11 **SETx2C**: Output 2 Set flag Clear
Writing 1 to this bit clears the SETx2 flag in HRTIM_TIMxISR register

Bit 10 **RSTx1C**: Output 1 Reset flag Clear
Writing 1 to this bit clears the RSTx1 flag in HRTIM_TIMxISR register

Bit 9 **SETx1C**: Output 1 Set flag Clear
Writing 1 to this bit clears the SETx1 flag in HRTIM_TIMxISR register

Bit 8 **CPT2C**: Capture2 Interrupt flag Clear
Writing 1 to this bit clears the CPT2 flag in HRTIM_TIMxISR register

Bit 7 **CPT1C**: Capture1 Interrupt flag Clear
Writing 1 to this bit clears the CPT1 flag in HRTIM_TIMxISR register

Bit 6 **UPDC**: Update Interrupt flag Clear
Writing 1 to this bit clears the UPD flag in HRTIM_TIMxISR register

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPC**: Repetition Interrupt flag Clear
Writing 1 to this bit clears the REP flag in HRTIM_TIMxISR register

Bit 3 **CMP4C**: Compare 4 Interrupt flag Clear
Writing 1 to this bit clears the CMP4 flag in HRTIM_TIMxISR register

Bit 2 **CMP3C**: Compare 3 Interrupt flag Clear
Writing 1 to this bit clears the CMP3 flag in HRTIM_TIMxISR register

Bit 1 **CMP2C**: Compare 2 Interrupt flag Clear
Writing 1 to this bit clears the CMP2 flag in HRTIM_TIMxISR register

Bit 0 **CMP1C**: Compare 1 Interrupt flag Clear
Writing 1 to this bit clears the CMP1 flag in HRTIM_TIMxISR register

36.5.15 HRTIM Timerx DMA / Interrupt Enable Register (HRTIM_TIMxDIER)

Address offset: 0x000Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SETx1DE	CPT2DE	CPT1DE	UPDDE	Res.	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	Res.	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 Reserved

Bit 30 **DLYPRTDE**: Delayed Protection DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on delayed protection.
0: Delayed protection DMA request disabled
1: Delayed protection DMA request enabled

Bit 29 **RSTDE**: Reset/roll-over DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over DMA request disabled
1: Timer x counter reset/roll-over DMA request enabled

Bit 28 **RSTx2DE**: Output 2 Reset DMA request Enable
Refer to RSTx1DE description

Bit 27 **SETx2DE**: Output 2 Set DMA request Enable
Refer to SETx1DE description

Bit 26 **RSTx1DE**: Output 1 Reset DMA request Enable
This bit is set and cleared by software to enable/disable Tx1 output reset DMA requests.
0: Tx1 output reset DMA request disabled
1: Tx1 output reset DMA request enabled

Bit 25 **SETx1DE**: Output 1 Set DMA request Enable
This bit is set and cleared by software to enable/disable Tx1 output set DMA requests.
0: Tx1 output set DMA request disabled
1: Tx1 output set DMA request enabled

Bit 24 **CPT2DE**: Capture 2 DMA request Enable
Refer to CPT1DE description

Bit 23 **CPT1DE**: Capture 1 DMA request Enable
This bit is set and cleared by software to enable/disable Capture 1 DMA requests.
0: Capture 1 DMA request disabled
1: Capture 1 DMA request enabled

Bit 22 **UPDDE**: Update DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on update event.
0: Update DMA request disabled
1: Update DMA request enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **REPDE**: Repetition DMA request Enable
This bit is set and cleared by software to enable/disable DMA requests on repetition event.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **CMP4DE**: Compare 4 DMA request Enable
Refer to CMP1DE description

Bit 18 **CMP3DE**: Compare 3 DMA request Enable
Refer to CMP1DE description

Bit 17 **CMP2DE**: Compare 2 DMA request Enable
Refer to CMP1DE description

Bit 16 **CMP1DE**: Compare 1 DMA request Enable
This bit is set and cleared by software to enable/disable the Compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bit 15 Reserved

Bit 14 **DLYPRTIE**: Delayed Protection Interrupt Enable
This bit is set and cleared by software to enable/disable interrupts on delayed protection.
0: Delayed protection interrupts disabled
1: Delayed protection interrupts enabled

Bit 13 **RSTIE**: Reset/roll-over Interrupt Enable
This bit is set and cleared by software to enable/disable interrupts on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over interrupt disabled
1: Timer x counter reset/roll-over interrupt enabled

Bit 12 **RSTx2IE**: Output 2 Reset Interrupt Enable
Refer to RSTx1IE description

Bit 11 **SETx2IE**: Output 2 Set Interrupt Enable
Refer to SETx1IE description

Bit 10 **RSTx1IE**: Output 1 Reset Interrupt Enable
This bit is set and cleared by software to enable/disable Tx1 output reset interrupts.
0: Tx1 output reset interrupts disabled
1: Tx1 output reset interrupts enabled

Bit 9 **SETx1IE**: Output 1 Set Interrupt Enable
This bit is set and cleared by software to enable/disable Tx1 output set interrupts.
0: Tx1 output set interrupts disabled
1: Tx1 output set interrupts enabled

Bit 8 **CPT2IE**: Capture Interrupt Enable
Refer to CPT1IE description

Bit 7 **CPT1IE**: Capture Interrupt Enable
This bit is set and cleared by software to enable/disable Capture 1 interrupts.
0: Capture 1 interrupts disabled
1: Capture 1 interrupts enabled

Bit 6 **UPDIE**: Update Interrupt Enable
This bit is set and cleared by software to enable/disable update event interrupts.
0: Update interrupts disabled
1: Update interrupts enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **REPIE**: Repetition Interrupt Enable
This bit is set and cleared by software to enable/disable repetition event interrupts.
0: Repetition interrupts disabled
1: Repetition interrupts enabled

Bit 3 **CMP4IE**: Compare 4 Interrupt Enable
Refer to CMP1IE description

Bit 2 **CMP3IE**: Compare 3 Interrupt Enable
Refer to CMP1IE description

Bit 1 **CMP2IE**: Compare 2 Interrupt Enable
Refer to CMP1IE description

Bit 0 **CMP1IE**: Compare 1 Interrupt Enable
This bit is set and cleared by software to enable/disable the Compare 1 interrupts.
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

36.5.16 HRTIM Timerx Counter Register (HRTIM_CNTxR)

Address offset: 0x0010h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNTx[15:0]**: *Timerx Counter value*

This register holds the Timerx counter value. It can only be written when the timer is stopped (TxCEN = 0 in HRTIM_TIMxCR).

Note: The timer behavior is not guaranteed if the counter value is above the HRTIM_PERxR register value.

36.5.17 HRTIM Timerx Period Register (HRTIM_PERxR)

Address offset: 0x14h (this offset address is relative to timer x base address)

Reset value: 0x0000 FFDF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PERx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PERx[15:0]**: *Timerx Period value*

This register holds timer x period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock.

The maximum value is 0x0000 FFDF.

36.5.18 HRTIM Timerx Repetition Register (HRTIM_REPxR)

Address offset: 0x18h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits31:8 Reserved, must be kept at reset value.

Bits 7:0 **REPx[7:0]**: *Timerx Repetition period value*

This register holds the repetition period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

36.5.19 HRTIM Timerx Compare 1 Register (HRTIM_CMP1xR)

Address offset: 0x1Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP1x[15:0]**: *Timerx Compare 1 value*

This register holds the compare 1 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

36.5.20 HRTIM Timerx Compare 1 Compound Register (HRTIM_CMP1CxR)

Address offset: 0x20h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **REPx[7:0]**: *Timerx Repetition value (aliased from HRTIM_REPx register)*
This bitfield is an alias from the REPx[7:0] bitfield in the HRTIMx_REPxR register.

Bits 15:0 **CMP1x[15:0]**: *Timerx Compare 1 value*
This bitfield is an alias from the CMP1x[15:0] bitfield in the HRTIMx_CMP1xR register.

36.5.21 HRTIM Timerx Compare 2 Register (HRTIM_CMP2xR)

Address offset: 0x24h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP2x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP2x[15:0]**: *Timerx Compare 2 value*
This register holds the Compare 2 value.
This register holds either the content of the preload register or the content of the active register if preload is disabled.
The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.
This register can behave as an auto-delayed compare register, if enabled with DELCMP2[1:0] bits in HRTIM_TIMxCR.

36.5.22 HRTIM Timerx Compare 3 Register (HRTIM_CMP3xR)

Address offset: 0x28h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP3x[15:0]**: *Timerx Compare 3 value*

This register holds the Compare 3 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

36.5.23 HRTIM Timerx Compare 4 Register (HRTIM_CMP4xR)

Address offset: 0x2Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP4x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP4x[15:0]**: *Timerx Compare 4 value*

This register holds the Compare 4 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

This register can behave as an auto-delayed compare register, if enabled with DELCMP4[1:0] bits in HRTIM_TIMxCR.

36.5.24 HRTIM Timerx Capture 1 Register (HRTIM_CPT1xR)

Address offset: 0x30h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT1x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CPT1x[15:0]**: *Timerx Capture 1 value*

This register holds the counter value when the capture 1 event occurred.

36.5.25 HRTIM Timerx Capture 2 Register (HRTIM_CPT2xR)

Address offset: 0x34h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT2x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CPT2x[15:0]**: *Timerx Capture 2 value*

This register holds the counter value when the capture 2 event occurred.

36.5.26 HRTIM Timerx Deadtime Register (HRTIM_DTxR)

Address offset: 0x38h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTFLKx	DTFSLKx	Res.	Res.	Res.	Res.	SDTFx	DTFx[8:0]								
rwo	rwo					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTRLKx	DTRSLKx	Res.	DTPRSC[1:0]			SDTRx	DTRx[8:0]								
rwo	rwo		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DTFLKx**: *Deadtime Falling Lock*

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime falling value and sign is writable

1: Deadtime falling value and sign is read-only

Note: This bit is not preloaded

Bit 30 **DTFSLKx**: *Deadtime Falling Sign Lock*

This write-once bit prevents the sign of falling deadtime to be modified, if enabled.

0: Deadtime falling sign is writable

1: Deadtime falling sign is read-only

Note: This bit is not preloaded

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **SDTFx**: *Sign Deadtime Falling value*

This register determines whether the deadtime is positive (signals not overlapping) or negative (signals overlapping).

0: Positive deadtime on falling edge

1: Negative deadtime on falling edge

Bits 24:16 **DTFx[8:0]**: *Deadtime Falling value*

This register holds the value of the deadtime following a falling edge of reference PWM signal.

t_{DTF} = DTFx[8:0] x t_{DTG}

Bit 15 **DTRLKx**: *Deadtime Rising Lock*

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled

0: Deadtime rising value and sign is writable

1: Deadtime rising value and sign is read-only

Note: This bit is not preloaded

Bit 14 **DTRSLKx**: *Deadtime Rising Sign Lock*

This write-once bit prevents the sign of deadtime to be modified, if enabled

0: Deadtime rising sign is writable

1: Deadtime rising sign is read-only

Note: This bit is not preloaded

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DTPRSC[2:0]**: *Deadtime Prescaler*

This register holds the value of the deadtime clock prescaler.

$t_{DTG} = (2^{(DTPRSC[2:0]-3)}) \times t_{HRTIM}$

000: Reserved

001: Reserved

010: Reserved

011: tDTG= t_{HRTIM}

100: tDTG= t_{HRTIM} x 2

101: tDTG= t_{HRTIM} x 4

110: tDTG= t_{HRTIM} x 8

111: tDTG= t_{HRTIM} x 16

This bitfield is read-only as soon as any of the lock bit is enabled (DTFLKs, DTFSLKx, DTRLKx, DTRSLKx).

Bit 9 **SDTRx**: *Sign Deadtime Rising value*

This register determines whether the deadtime is positive or negative (overlapping signals)

0: Positive deadtime on rising edge

1: Negative deadtime on rising edge

Bits 8:0 **DTRx[8:0]**: *Deadtime Rising value*

This register holds the value of the deadtime following a rising edge of reference PWM signal.

t_{DTR} = DTRx[8:0] x t_{DTG}

36.5.27 HRTIM Timerx Output1 Set Register (HRTIM_SETx1R)

Address offset: 0x3Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: *Registers update (transfer preload to active)*
Register update event forces the output to its active state.

Bit 30 **EXTEVNT10**: *External Event 10*
Refer to EXTEVNT1 description

Bit 29 **EXTEVNT9**: *External Event 9*
Refer to EXTEVNT1 description

Bit 28 **EXTEVNT8**: *External Event 8*
Refer to EXTEVNT1 description

Bit 27 **EXTEVNT7**: *External Event 7*
Refer to EXTEVNT1 description

Bit 26 **EXTEVNT6**: *External Event 6*
Refer to EXTEVNT1 description

Bit 25 **EXTEVNT5**: *External Event 5*
Refer to EXTEVNT1 description

Bit 24 **EXTEVNT4**: *External Event 4*
Refer to EXTEVNT1 description

Bit 23 **EXTEVNT3**: *External Event 3*
Refer to EXTEVNT1 description

Bit 22 **EXTEVNT2**: *External Event 2*
Refer to EXTEVNT1 description

Bit 21 **EXTEVNT1**: *External Event 1*
External event 1 forces the output to its active state.

Bit 20 **TIMEVNT9**: *Timer Event 9*
Refer to TIMEVNT1 description

Bit 19 **TIMEVNT8**: *Timer Event 8*
Refer to TIMEVNT1 description

Bit 18 **TIMEVNT7**: *Timer Event 7*
Refer to TIMEVNT1 description

Bit 17 **TIMEVNT6**: *Timer Event 6*
Refer to TIMEVNT1 description

Bit 16 **TIMEVNT5**: *Timer Event 5*
Refer to TIMEVNT1 description

Bit 15 **TIMEVNT4**: *Timer Event 4*
Refer to TIMEVNT1 description

Bit 14 **TIMEVNT3**: *Timer Event 3*
Refer to TIMEVNT1 description

Bit 13 **TIMEVNT2**: *Timer Event 2*
Refer to TIMEVNT1 description

Bit 12 **TIMEVNT1**:*Timer Event 1*
Timers event 1 forces the output to its active state (refer to *Table 296* for Timer Events assignments)

Bit 11 **MSTCMP4**: *Master Compare 4*
Master Timer Compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: *Master Compare 3*
Master Timer Compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: *Master Compare 2*
Master Timer Compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: *Master Compare 1*
Master Timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: *Master Period*
The master timer counter roll-over in continuous mode, or to the master timer reset in single-shot mode forces the output to its active state.

Bit 6 **CMP4**: *Timer x Compare 4*
Timer A compare 4 event forces the output to its active state.

Bit 5 **CMP3**: *Timer x Compare 3*
Timer A compare 3 event forces the output to its active state.

Bit 4 **CMP2**: *Timer x Compare 2*
Timer A compare 2 event forces the output to its active state.

Bit 3 **CMP1**: *Timer x Compare 1*
Timer A compare 1 event forces the output to its active state.

Bit 2 **PER**: *Timer x Period*
Timer A Period event forces the output to its active state.

Bit 1 **RESYNC:** *Timer A resynchronization*
Timer A reset event coming solely from software or SYNC input forces the output to its active state.
Note: Other timer reset are not affecting the output when RESYNC=1

Bit 0 **SST**: *Software Set trigger*
This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.
Note: This bit is not preloaded

36.5.28 HRTIM Timerx Output1 Reset Register (HRTIM_RSTx1R)

Address offset: 0x40h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.

These bits are defining the source which can force the Tx1 output to its inactive state.

36.5.29 HRTIM Timerx Output2 Set Register (HRTIM_SETx2R)

Address offset: 0x44h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.

These bits are defining the source which can force the Tx2 output to its active state.

36.5.30 HRTIM Timerx Output2 Reset Register (HRTIM_RSTx2R)

Address offset: 0x48h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_SETx1R bits description.

These bits are defining the source which can force the Tx2 output to its inactive state.

36.5.31 HRTIM Timerx External Event Filtering Register 1 (HRTIM_EEFxR1)

Address offset: 0x4Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE5FLTR[3:0]				EE5LTCH	Res.	EE4FLTR[3:0]				EE4LTCH	Res.	EE3FLTR[3]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE3FLTR[2:0]			EE3LTCH	Res.	EE2FLTR[3:0]				EE2LTCH	Res.	EE1FLTR[3:0]				EE1LTCH
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **EE5FLTR[3:0]**: *External Event 5 filter*
Refer to EE1FLTR[3:0] description

Bit 24 **EE5LTCH**: *External Event 5 latch*
Refer to EE1LTCH description

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE4FLTR[3:0]**: *External Event 4 filter*
Refer to EE1FLTR[3:0] description

Bit 18 **EE4LTCH**: *External Event 4 latch*
Refer to EE1LTCH description

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE3FLTR[3:0]**: *External Event 3 filter*
Refer to EE1FLTR[3:0] description

Bit 12 **EE3LTCH**: *External Event 3 latch*
Refer to EE1LTCH description

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE2FLTR[3:0]**: *External Event 2 filter*
Refer to EE1FLTR[3:0] description

Bit 6 **EE2LTCH**: *External Event 2 latch*
Refer to EE1LTCH description

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE1FLTR[3:0]**: *External Event 1 filter*

0000: No filtering
0001: Blanking from counter reset/roll-over to Compare 1
0010: Blanking from counter reset/roll-over to Compare 2
0011: Blanking from counter reset/roll-over to Compare 3
0100: Blanking from counter reset/roll-over to Compare 4
0101: Blanking from another timing unit: TIMFLTR1 source (see *Table 300* for details)
0110: Blanking from another timing unit: TIMFLTR2 source (see *Table 300* for details)
0111: Blanking from another timing unit: TIMFLTR3 source (see *Table 300* for details)
1000: Blanking from another timing unit: TIMFLTR4 source (see *Table 300* for details)
1001: Blanking from another timing unit: TIMFLTR5 source (see *Table 300* for details)
1010: Blanking from another timing unit: TIMFLTR6 source (see *Table 300* for details)
1011: Blanking from another timing unit: TIMFLTR7 source (see *Table 300* for details)
1100: Blanking from another timing unit: TIMFLTR8 source (see *Table 300* for details)
1101: Windowing from counter reset/roll-over to Compare 2
1110: Windowing from counter reset/roll-over to Compare 3
1111: Windowing from another timing unit: TIMWIN source (see *Table 301* for details)

Note: Whenever a compare register is used for filtering, the value must be strictly above 0.
This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 0 **EE1LTCH**: *External Event 1 latch*

0: Event 1 is ignored if it happens during a blank, or passed through during a window.
1: Event 1 is latched and delayed till the end of the blanking or windowing period.

Note: A timeout event is generated in window mode (EE1FLTR[3:0]=1101, 1110, 1111) if EE1LTCH = 0, except if the External event is programmed in fast mode (EExFAST = 1).
This bitfield must not be modified once the counter is enabled (TxCEN bit set)

36.5.32 HRTIM Timerx External Event Filtering Register 2 (HRTIM_EEFxR2)

Address offset: 0x50h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10FLTR[3:0]				EE10LTCH	Res.	EE9FLTR[3:0]				EE9LTCH	Res.	EE8FLTR[3]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8FLTR[2:0]			EE8LTCH	Res.	EE7FLTR[3:0]				EE7LTCH	Res.	EE6FLTR[3:0]				EE6LTCH
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **EE10FLTR[3:0]**: *External Event 10 filter*
Refer to EE1FLTR[3:0] description

Bit 24 **EE10LTCH**: *External Event 10 latch*
Refer to EE1LTCH description

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 **EE9FLTR[3:0]**: *External Event 9 filter*
Refer to EE1FLTR[3:0] description

Bit 18 **EE9LTCH**: *External Event 9 latch*
Refer to EE1LTCH description

Bit 17 Reserved, must be kept at reset value.

Bits 16:13 **EE8FLTR[3:0]**: *External Event 8 filter*
Refer to EE1FLTR[3:0] description

Bit 12 **EE8LTCH**: *External Event 8 latch*
Refer to EE1LTCH description

Bit 11 Reserved, must be kept at reset value.

Bits 10:7 **EE7FLTR[3:0]**: *External Event 7 filter*
Refer to EE1FLTR[3:0] description

Bit 6 **EE7LTCH**: *External Event 7 latch*
Refer to EE1LTCH description

Bit 5 Reserved, must be kept at reset value.

Bits 4:1 **EE6FLTR[3:0]**: *External Event 6 filter*
Refer to EE1FLTR[3:0] description

Bit 0 **EE6LTCH**: *External Event 6 latch*
Refer to EE1LTCH description

36.5.33 HRTIM Timerx Reset Register (HRTIM_RSTxR)

HRTIM TimerA Reset Register (HRTIM_RSTAR)

Address offset: 0xD4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TECPM4**: *Timer E Compare 4*
The timer A counter is reset upon timer E Compare 4 event.

Bit 29 **TECMP2**: *Timer E Compare 2*
The timer A counter is reset upon timer E Compare 2 event.

Bit 28 **TECMP1**: *Timer E Compare 1*
The timer A counter is reset upon timer E Compare 1 event.

Bit 27 **TDCMP4**: *Timer D Compare 4*
The timer A counter is reset upon timer D Compare 4 event.

Bit 26 **TDCMP2**: *Timer D Compare 2*
The timer A counter is reset upon timer D Compare 2 event.

Bit 25 **TDCMP1**: *Timer D Compare 1*
The timer A counter is reset upon timer D Compare 1 event.

Bit 24 **TCCMP4**: *Timer C Compare 4*
The timer A counter is reset upon timer C Compare 4 event.

Bit 23 **TCCMP2**: *Timer C Compare 2*
The timer A counter is reset upon timer C Compare 2 event.

Bit 22 **TCCMP1**: *Timer C Compare 1*
The timer A counter is reset upon timer C Compare 1 event.

Bit 21 **TBCMP4**: *Timer B Compare 4*
The timer A counter is reset upon timer B Compare 4 event.

Bit 20 **TBCMP2**: *Timer B Compare 2*
The timer A counter is reset upon timer B Compare 2 event.

Bit 19 **TBCMP1**: *Timer B Compare 1*
The timer A counter is reset upon timer B Compare 1 event.

Bit 18 **EXTEVNT10**: *External Event*
The timer A counter is reset upon external event 10.

Bit 17 **EXTEVNT9**: *External Event 9*
The timer A counter is reset upon external event 9.

Bit 16 **EXTEVNT8**: *External Event 8*
The timer A counter is reset upon external event 8.

Bit 15 **EXTEVNT7**: *External Event 7*
The timer A counter is reset upon external event 7.

Bit 14 **EXTEVNT6**: *External Event 6*
The timer A counter is reset upon external event 6.

Bit 13 **EXTEVNT5**: *External Event 5*
The timer A counter is reset upon external event 5.

Bit 12 **EXTEVNT4**: *External Event 4*
The timer A counter is reset upon external event 4.

Bit 11 **EXTEVNT3**: *External Event 3*
The timer A counter is reset upon external event 3.

Bit 10 **EXTEVNT2**: *External Event 2*
The timer A counter is reset upon external event 2.

Bit 9 **EXTEVNT1**: *External Event 1*
The timer A counter is reset upon external event 1.

Bit 8 **MSTCMP4**: *Master compare 4*
The timer A counter is reset upon master timer Compare 4 event.

Bit 7 **MSTCMP3**: *Master compare 3*
The timer A counter is reset upon master timer Compare 3 event.

Bit 6 **MSTCMP2**: *Master compare 2*
The timer A counter is reset upon master timer Compare 2 event.

Bit 5 **MSTCMP1**: *Master compare 1*
The timer A counter is reset upon master timer Compare 1 event.

Bit 4 **MSTPER** *Master timer Period*
The timer A counter is reset upon master timer period event.

Bit 3 **CMP4**: *Timer A compare 4 reset*
The timer A counter is reset upon Timer A Compare 4 event.

Bit 2 **CMP2**: *Timer A compare 2 reset*
The timer A counter is reset upon Timer A Compare 2 event.

Bit 1 **UPDT**: *Timer A Update reset*
The timer A counter is reset upon update event.

Bit 0 Reserved, must be kept at reset value.

HRTIM TimerB Reset Register (HRTIM_RSTBR)

Address offset: 0x154h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMC, TIMD and TIME)

HRTIM TimerC Reset Register (HRTIM_RSTCR)

Address offset: 0x1D4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMB, TIMD and TIME)

HRTIM TimerD Reset Register (HRTIM_RSTDR)

Address offset: 0x254h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMB, TIMC and TIME)

HRTIM Timerx Reset Register (HRTIM_RSTER)

Address offset: 0x2D4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIMD CMP4	TIMD CMP2	TIMD CMP1	TIMC CMP4	TIMC CMP2	TIMC CMP1	TIMB CMP4	TIMB CMP2	TIMB CMP1	TIMA CMP4	TIMA CMP2	TIMA CMP1	EXTEV NT10	EXTEV NT9	EXTEV NT8
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEV NT7	EXTEV NT6	EXTEV NT5	EXTEV NT4	EXTEV NT3	EXTEV NT2	EXTEV NT1	MSTC MP4	MSTC MP3	MSTC MP2	MSTC MP1	MSTPE R	CMP4	CMP2	UPDT	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 30:1 Refer to HRTIM_RSTAR bits description.
Bits 30:19 differ (reset signals come from TIMA, TIMB, TIMC and TIMD)

36.5.34 HRTIM Timerx Chopper Register (HRTIM_CHPxR)

Address offset: 0x58h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	STRTPW[3:0]				CARDTY[2:0)			CARFRQ[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:7 **STRPW[3:0]**: *Timerx start pulsewidth*

This register defines the initial pulsewidth following a rising edge on output signal.
This bitfield cannot be modified when one of the CHPx bits is set.
t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.

0000: 40 ns (1/25 MHz)
...
1111: 640 ns (16/25 MHz)

Bits 6:4 **CARDTY[2:0]**: *Timerx chopper duty cycle value*

This register defines the duty cycle of the carrier signal. This bitfield cannot be modified when one of the CHPx bits is set.
000: 0/8 (i.e. only 1st pulse is present)
...
111: 7/8

Bits 3:0 **CARFRQ[3:0]**: *Timerx carrier frequency value*

This register defines the carrier frequency F_{CHPFRQ} = f_{HRTIM} / (16 x (CARFRQ[3:0]+1)).
This bitfield cannot be modified when one of the CHPx bits is set.
0000: 25 MHz (f_{HRTIM}/ 16)
...
1111: 1.56 MHz (f_{HRTIM} / 256)

36.5.35 HRTIM Timerx Capture 1 Control Register (HRTIM_CPT1xCR)

Address offset: 0x5Ch (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved (for TIME only)				Reserved (for TIMD only)				Reserved (for TIMC only)				Reserved (for TIMB only)			
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved (for TIMA only)				EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
TACMP2	TACMP1	TA1RST	TA1SET												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Refer to HRTIM_CPT2xCR bit description

36.5.36 HRTIM Timerx Capture 2 Control Register (HRTIM_CPT2xCR)

Address offset: 0x60h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved (for TIME only)				Reserved (for TIMD only)				Reserved (for TIMC only)				Reserved (for TIMB only)			
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved (for TIMA only)				EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
TACMP2	TACMP1	TA1RST	TA1SET												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer E

Bit 30 **TECMP1**: Timer E Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer E

Bit 29 **TE1RST**: Timer E output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer E

Bit 28 **TE1SET**: Timer E output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer E

Bit 27 **TDCMP2**: Timer D Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer D

Bit 26 **TDCMP1**:Timer D Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer D

Bit 25 **TD1RST**: Timer D output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer D

Bit 24 **TD1SET**: Timer D output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer D

Bit 23 **TCCMP2**: Timer C Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer C

Bit 22 **TCCMP1**:Timer C Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer C

Bit 21 **TC1RST**: Timer C output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer C

Bit 20 **TC1SET**: Timer C output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer C

Bit 19 **TBCMP2**: Timer B Compare 2
Refer to TACMP1 description
Note: This bit is reserved for Timer B

Bit 18 **TBCMP1**: Timer B Compare 1
Refer to TACMP1 description
Note: This bit is reserved for Timer B

Bit 17 **TB1RST**: Timer B output 1 Reset
Refer to TA1RST description
Note: This bit is reserved for Timer B

Bit 16 **TB1SET**: Timer B output 1 Set
Refer to TA1SET description
Note: This bit is reserved for Timer B

Bit 15 **TACMP2**: Timer A Compare 2
Timer A Compare 2 triggers Capture 2.
Note: This bit is reserved for Timer A

Bit 14 **TACMP1**: Timer A Compare 1
Timer A Compare 1 triggers Capture 2.
Note: This bit is reserved for Timer A

Bit 13 **TA1RST**: Timer B output 1 Reset
Capture 2 is triggered by HRTIM_CHA1 output active to inactive transition.
Note: This bit is reserved for Timer A

Bit 12 **TA1SET**: Timer B output 1 Set
Capture 2 is triggered by HRTIM_CHA1 output inactive to active transition.
Note: This bit is reserved for Timer A

Bit 11 **EXEV10CPT**: *External Event 10 Capture*
Refer to EXEV1CPT description

Bit 10 **EXEV9CPT**: *External Event 9 Capture*
Refer to EXEV1CPT description

Bit 9 **EXEV8CPT**: *External Event 8 Capture*
Refer to EXEV1CPT description

Bit 8 **EXEV7CPT**: *External Event 7 Capture*
Refer to EXEV1CPT description

Bit 7 **EXEV6CPT**: *External Event 6 Capture*
Refer to EXEV1CPT description

Bit 6 **EXEV5CPT**: *External Event 5 Capture*
Refer to EXEV1CPT description

Bit 5 **EXEV4CPT**: *External Event 4 Capture*
Refer to EXEV1CPT description

Bit 4 **EXEV3CPT**: *External Event 3 Capture*
Refer to EXEV1CPT description

Bit 3 **EXEV2CPT**: *External Event 2 Capture*
Refer to EXEV1CPT description

Bit 2 **EXEV1CPT**: *External Event 1 Capture*
The External event 1 triggers the Capture 2.

Bit 1 **UPDCPT**: *Update Capture*
The update event triggers the Capture 2.

Bit 0 **SWCPT**: *Software Capture*
This bit forces the Capture 2 by software. This bit is set only, reset by hardware

36.5.37 HRTIM Timerx Output Register (HRTIM_OUTxR)

Address offset: 0x64h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM 2	POL2	Res.
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DLYPRT[2:0]			DLYPR TEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM 1	POL1	Res.
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DIDL2**: *Output 2 Deadtime upon burst mode Idle entry*

This bit can delay the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering in idle state during a burst mode operation.

0: The programmed Idle state is applied immediately to the Output 2

1: Deadtime (inactive level) is inserted on output 2 before entering the idle mode. The deadtime value is set by DTFx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 can be set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 22 **CHP2**: *Output 2 Chopper enable*

This bit enables the chopper on output 2

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 21:20 **FAULT2[1:0]**: *Output 2 Fault state*

These bits select the output 2 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 19 **IDLES2**: *Output 2 Idle State*

This bit selects the output 2 idle state

0: Inactive

1: Active

Note: This parameter must be set prior to have the HRTIM controlling the outputs.

Bit 18 **IDLEM2**: *Output 2 Idle mode*

This bit selects the output 2 idle mode

0: No action: the output is not affected by the burst mode operation

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and can be changed during run-time, but must not be changed while the burst mode is active.

Bit 17 **POL2**: *Output 2 polarity*

This bit selects the output 2 polarity
0: positive polarity (output active high)
1: negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bits 16:12 Reserved, must be kept at reset value.

Bits 12:10 **DLYPRT[2:0]**: *Delayed Protection*

These bits define the source and outputs on which the delayed protection schemes are applied.
In HRTIM_OUTAR, HRTIM_OUTBR, HRTIM_OUTCR:
000: Output 1 delayed Idle on external Event 6
001: Output 2 delayed Idle on external Event 6
010: Output 1 and output 2 delayed Idle on external Event 6
011: Balanced Idle on external Event 6
100: Output 1 delayed Idle on external Event 7
101: Output 2 delayed Idle on external Event 7
110: Output 1 and output 2 delayed Idle on external Event 7
111: Balanced Idle on external Event 7
In HRTIM_OUTDR, HRTIM_OUTER:
000: Output 1 delayed Idle on external Event 8
001: Output 2 delayed Idle on external Event 8
010: Output 1 and output 2 delayed Idle on external Event 8
011: Balanced Idle on external Event 8
100: Output 1 delayed Idle on external Event 9
101: Output 2 delayed Idle on external Event 9
110: Output 1 and output 2 delayed Idle on external Event 9
111: Balanced Idle on external Event 9

Note: This bitfield must not be modified once the delayed protection is enabled (DLYPRTEN bit set)

Bit 9 **DLYPRTEN**: *Delayed Protection Enable*

This bit enables the delayed protection scheme
0: No action
1: Delayed protection is enabled, as per DLYPRT[2:0] bits

Note: This parameter cannot be changed once the timer x is enabled (TxEN bit set).

Bit 8 **DTEN**: *Deadtime enable*

This bit enables the deadtime insertion on output 1 and output 2
0: Output 1 and output 2 signals are independent.
1: Deadtime is inserted between output 1 and output 2 (reference signal is output 1 signal generator)

Note: This parameter cannot be changed once the timer is operating (TxEN bit set) or if its outputs are enabled and set/reset by another timer.

Bit 7 **DIDL1**: *Output 1 Deadtime upon burst mode Idle entry*

This bit can delay the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering the idle state during a burst mode operation.
0: The programmed Idle state is applied immediately to the Output 1
1: Deadtime (inactive level) is inserted on output 1 before entering the idle mode. The deadtime value is set by DTRx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.
DIDL=1 can be set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 6 **CHP1**: *Output 1 Chopper enable*

This bit enables the chopper on output 1

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 5:4 **FAULT1[1:0]**: *Output 1 Fault state*

These bits select the output 1 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 3 **IDLES1**: *Output 1 Idle State*

This bit selects the output 1 idle state

0: Inactive

1: Active

Note: This parameter must be set prior to HRTIM controlling the outputs.

Bit 2 **IDLEM1**: *Output 1 Idle mode*

This bit selects the output 1 idle mode

0: No action: the output is not affected by the burst mode operation

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and can be changed during runtime, but must not be changed while burst mode is active.

Bit 1 **POL1**: *Output 1 polarity*

This bit selects the output 1 polarity

0: positive polarity (output active high)

1: negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bit 0 Reserved

36.5.38 HRTIM Timerx Fault Register (HRTIM_FLTxR)

Address offset: 0x68h (this offset address is relative to timer x base address)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rwo															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
											rw	rw	rw	rw	rw

Bit 31 **FLTLCK**: *Fault sources Lock*
0: FLT1EN..FLT5EN bits are read/write
1: FLT1EN..FLT5EN bits are read only
The FLTLCK bit is write-once. Once it has been set, it cannot be modified till the next system reset.

Bits 30:5 Reserved, must be kept at reset value.

Bit 4 **FLT5EN**: *Fault 5 enable*
0: Fault 5 input ignored
1: Fault 5 input is active and can disable HRTIM outputs.

Bit 3 **FLT4EN**: *Fault 4 enable*
0: Fault 4 input ignored
1: Fault 4 input is active and can disable HRTIM outputs.

Bit 2 **FLT3EN**: *Fault 3 enable*
0: Fault 3 input ignored
1: Fault 3 input is active and can disable HRTIM outputs.

Bit 1 **FLT2EN**: *Fault 2 enable*
0: Fault 2 input ignored
1: Fault 2 input is active and can disable HRTIM outputs.

Bit 0 **FLT1EN**: *Fault 1 enable*
0: Fault 1 input ignored
1: Fault 1 input is active and can disable HRTIM outputs.

36.5.39 HRTIM Control Register 1 (HRTIM_CR1)

Address offset: 0x380h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
										rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:25 **AD4USRC[2:0]**: *ADC Trigger 4* Update Source
Refer to AD1USRC[2:0] description

Bits 24:22 **AD3USRC[2:0]**: *ADC Trigger 3 Update Source*
Refer to AD1USRC[2:0] description

Bits 21:19 **AD2USRC[2:0]**: *ADC Trigger 2 Update Source*
Refer to AD1USRC[2:0] description

Bits 18:16 **AD1USRC[2:0]**: *ADC Trigger 1 Update Source*
These bits define the source which will trigger the update of the HRTIM_ADC1R register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, in HRTIM_MCR or HRTIM_TIMxCR.
000: Master Timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110, 111: Reserved

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TEUDIS**: *Timer E Update Disable*
Refer to TAUDIS description

Bit 4 **TDUDIS**: *Timer D Update Disable*
Refer to TAUDIS description

Bit 3 **TCUDIS**: *Timer C Update Disable*
Refer to TAUDIS description

Bit 2 **TBUDIS**: *Timer B Update Disable*

Refer to TAUDIS description

Bit 1 **TAUDIS**: *Timer A Update Disable*

This bit is set and cleared by software to enable/disable an update event generation temporarily on Timer A.

0: update enabled. The update occurs upon generation of the selected source.

1: update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 0 **MUDIS**: Master *Update Disable*

This bit is set and cleared by software to enable/disable an update event generation temporarily.

0: update enabled.

1: update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

36.5.40 HRTIM Control Register 2 (HRTIM_CR2)

Address offset: 0x384h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	Res.	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **TERST**: *Timer E counter software reset*
Refer to TARST description

Bit 12 **TDRST**: *Timer D counter software reset*
Refer to TARST description

Bit 11 **TCRST**: *Timer C counter software reset*
Refer to TARST description

Bit 10 **TBRST**: *Timer B counter software reset*
Refer to TARST description

Bit 9 **TARST**: *Timer A counter software reset*
Setting this bit resets the TimerA counter.
The bit is automatically reset by hardware.

Bit 8 **MRST**: Master *Counter software reset*
Setting this bit resets the Master timer counter.
The bit is automatically reset by hardware.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **TESWU**: *Timer E Software Update*
Refer to TASWU description

Bit 4 **TDSWU**: *Timer D Software Update*
Refer to TASWU description

Bit 3 **TCSWU**: *Timer C Software Update*
Refer to TASWU description

Bit 2 **TBSWU**: *Timer B Software Update*
Refer to TASWU description

Bit 1 **TASWU**: *Timer A Software update*
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register and any pending update request is cancelled.

Bit 0 **MSWU**: *Master Timer Software update*
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register in the master timer and any pending update request is cancelled.

36.5.41 HRTIM Interrupt Status Register (HRTIM_ISR)

Address offset: 0x388h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	Res.
														r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
											r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPER**: Burst mode Period Interrupt Flag

This bit is set by hardware when a single-shot burst mode operation is completed or at the end of a burst mode period in continuous mode. It is cleared by software writing it at 1.

0: No Burst mode period interrupt occurred

1: Burst mode period interrupt occurred

Bits 16:6 Reserved, must be kept at reset value.

Bit 5 **SYSFLT**: System Fault Interrupt Flag

Refer to FLT1 description

Bit 4 **FLT5**: Fault 5 Interrupt Flag

Refer to FLT1 description

Bit 3 **FLT4**: Fault 4 Interrupt Flag

Refer to FLT1 description

Bit 2 **FLT3**: Fault 3 Interrupt Flag

Refer to FLT1 description

Bit 1 **FLT2**: Fault 2 Interrupt Flag

Refer to FLT1 description

Bit 0 **FLT1**: Fault 1 Interrupt Flag

This bit is set by hardware when Fault 1 event occurs. It is cleared by software writing it at 1.

0: No Fault 1 interrupt occurred

1: Fault 1 interrupt occurred

36.5.42 HRTIM Interrupt Clear Register (HRTIM_ICR)

Address offset: 0x38Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
											w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERC**: Burst mode period flag Clear
Writing 1 to this bit clears the BMPER flag in HRTIM_ISR register.

Bits 16:6 Reserved, must be kept at reset value.

Bit 5 **SYSFLTC**: System Fault Interrupt Flag Clear
Writing 1 to this bit clears the SYSFLT flag in HRTIM_ISR register.

Bit 4 **FLT5C**: Fault 5 Interrupt Flag Clear
Writing 1 to this bit clears the FLT5 flag in HRTIM_ISR register.

Bit 3 **FLT4C**: Fault 4 Interrupt Flag Clear
Writing 1 to this bit clears the FLT4 flag in HRTIM_ISR register.

Bit 2 **FLT3C**: Fault 3 Interrupt Flag Clear
Writing 1 to this bit clears the FLT3 flag in HRTIM_ISR register.

Bit 1 **FLT2C**: Fault 2 Interrupt Flag Clear
Writing 1 to this bit clears the FLT2 flag in HRTIM_ISR register.

Bit 0 **FLT1C**: Fault 1 Interrupt Flag Clear
Writing 1 to this bit clears the FLT1 flag in HRTIM_ISR register.

36.5.43 HRTIM Interrupt Enable Register (HRTIM_IER)

Address offset: 0x390h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	Res.
														rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
											rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERIE**: Burst mode period Interrupt Enable
This bit is set and cleared by software to enable/disable the Burst mode period interrupt.
0: Burst mode period interrupt disabled
1: Burst mode period interrupt enabled

Bits 16:6 Reserved, must be kept at reset value.

Bit 5 **SYSFLTIE**: System Fault Interrupt Enable
Refer to FLT1IE description

Bit 4 **FLT5IE**: Fault 5 Interrupt Enable
Refer to FLT1IE description

Bit 3 **FLT4IE**: Fault 4 Interrupt Enable
Refer to FLT1IE description

Bit 2 **FLT3IE**: Fault 3 Interrupt Enable
Refer to FLT1IE description

Bit 1 **FLT2IE**: Fault 2 Interrupt Enable
Refer to FLT1IE description

Bit 0 **FLT1IE**: Fault 1 Interrupt Enable
This bit is set and cleared by software to enable/disable the Fault 1 interrupt.
0: Fault 1 interrupt disabled
1: Fault 1 interrupt enabled

36.5.44 HRTIM Output Enable Register (HRTIM_OENR)

Address offset: 0x394h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
						rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **TE2OEN**: Timer E Output 2 Enable
Refer to TA1OEN description

Bit 8 **TE1OEN**: Timer E Output 1 Enable
Refer to TA1OEN description

Bit 7 **TD2OEN**: Timer D Output 2 Enable
Refer to TA1OEN description

Bit 6 **TD1OEN**: Timer D Output 1 Enable
Refer to TA1OEN description

Bit 5 **TC2OEN**: Timer C Output 2 Enable
Refer to TA1OEN description

Bit 4 **TC1OEN**: Timer C Output 1 Enable
Refer to TA1OEN description

Bit 3 **TB2OEN**: Timer B Output 2 Enable
Refer to TA1OEN description

Bit 2 **TB1OEN**: Timer B Output 1 Enable
Refer to TA1OEN description

Bit 1 **TA2OEN**: Timer A Output 2 Enable
Refer to TA1OEN description

Bit 0 **TA1OEN**: Timer A Output 1 (HRTIM_CHA1) Enable
Setting this bit enables the Timer A output 1. Writing “0” has no effect.
Reading the bit returns the output enable/disable status.
This bit is cleared asynchronously by hardware as soon as the timer-related fault input(s) is (are) active.
0: output HRTIM_CHA1 disabled. The output is either in Fault or Idle state.
1: output HRTIM_CHA1 enabled

Note: The disable status corresponds to both idle and fault states. The output disable status is given by TA1ODS bit in the HRTIM_ODSR register.

36.5.45 HRTIM Output Disable Register (HRTIM_ODISR)

Address offset: 0x398h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
						w	w	w	w	w	w	w	w	w	w

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **TE2ODIS**: Timer E Output 2 disable
Refer to TA1ODIS description

Bit 8 **TE1ODIS**: Timer E Output 1 disable
Refer to TA1ODIS description

Bit 7 **TD2ODIS**: Timer D Output 2 disable
Refer to TA1ODIS description

Bit 6 **TD1ODIS**: Timer D Output 1 disable
Refer to TA1ODIS description

Bit 5 **TC2ODIS**: Timer C Output 2 disable
Refer to TA1ODIS description

Bit 4 **TC1ODIS**: Timer C Output 1 disable
Refer to TA1ODIS description

Bit 3 **TB2ODIS**: Timer B Output 2 disable
Refer to TA1ODIS description

Bit 2 **TB1ODIS**: Timer B Output 1 disable
Refer to TA1ODIS description

Bit 1 **TA2ODIS**: Timer A Output 2 disable
Refer to TA1ODIS description

Bit 0 **TA1ODIS**: Timer A Output 1 (HRTIM_CHA1) disable
Setting this bit disables the Timer A output 1. The output enters the idle state, either from the run state or from the fault state.
Writing “0” has no effect.

36.5.46 HRTIM Output Disable Status Register (HRTIM_ODSR)

Address offset: 0x39Ch

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **TE2ODS**: Timer E Output 2 disable status
Refer to TA1ODS description

Bit 8 **TE1ODS**: Timer E Output 1 disable status
Refer to TA1ODS description

Bit 7 **TD2ODS**: Timer D Output 2 disable status
Refer to TA1ODS description

Bit 6 **TD1ODS**: Timer D Output 1 disable status
Refer to TA1ODS description

Bit 5 **TC2ODS**: Timer C Output 2 disable status
Refer to TA1ODS description

Bit 4 **TC1ODS**: Timer C Output 1 disable status
Refer to TA1ODS description

Bit 3 **TB2ODS**: Timer B Output 2 disable status
Refer to TA1ODS description

Bit 2 **TB1ODS**: Timer B Output 1 disable status
Refer to TA1ODS description

Bit 1 **TA2ODS**: Timer A Output 2 disable status
Refer to TA1ODS description

Bit 0 **TA1ODS**: Timer A Output 1 disable status
Reading the bit returns the output disable status. It is not significant when the output is active (Tx1OEN or Tx2OEN = 1).
0: output HRTIM_CHA1 disabled, in Idle state.
1: output HRTIM_CHA1 disabled, in Fault state.

36.5.47 HRTIM Burst Mode Control Register (HRTIM_BMCR)

Address offset: 0x3A0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEBM	TDBM	TCBM	TBBM	TABM	MTBM
rc_w0										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BMSTAT**: *Burst Mode Status*

This bit gives the current operating state.

0: Normal operation

1: Burst operation on-going. Writing this bit to 0 causes a burst mode early termination.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **TEBM**: *Timer E Burst Mode*

Refer to TABM description

Bit 20 **TDBM**: *Timer D Burst Mode*

Refer to TABM description

Bit 19 **TCBM**: *Timer C Burst Mode*

Refer to TABM description

Bit 18 **TBBM**: *Timer B Burst Mode*

Refer to TABM description

Bit 17 **TABM**: *Timer A Burst Mode*

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: Timer A counter clock is maintained and the timer operates normally

1: Timer A counter clock is stopped and the counter is reset

Note: This bit must not be set when the balanced idle mode is active (DLYPRT[2:0] = 0x11)

Bit 16 **MTBM**: *Master Timer Burst Mode*

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: Master Timer counter clock is maintained and the timer operates normally

1: Master Timer counter clock is stopped and the counter is reset

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **BMPREN**: *Burst Mode Preload Enable*

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register (HRTIM_BMCMPR, HRTIM_BMPER) is done into the active or the preload register.

0: Preload disabled: the write access is directly done into active registers

1: Preload enabled: the write access is done into preload registers

Bits 9:6 **BMPRSC[3:0]**: *Burst Mode Prescaler*

Defines the prescaling ratio of the f_{HRTIM} clock for the burst mode controller. This bitfield cannot be changed while the burst mode is enabled.
0000: Clock not divided
0001: Division by 2
0010: Division by 4
0011: Division by 8
0100: Division by 16
0101: Division by 32
0110: Division by 64
0111: Division by 128
1000: Division by 256
1001: Division by 512
1010: Division by 1024
1011: Division by 2048
1100: Division by 4096
1101:Division by 8192
1110: Division by 16384
1111: Division by 32768

Bits 5:2 **BMCLK[3:0]**: *Burst Mode Clock source*

This bitfield defines the clock source for the burst mode counter. It cannot be changed while the burst mode is enabled (refer to *Table 310* for on-chip events 1..4 connections details).
0000: Master timer counter reset/roll-over
0001: Timer A counter reset/roll-over
0010: Timer B counter reset/roll-over
0011: Timer C counter reset/roll-over
0100: Timer D counter reset/roll-over
0101: Timer E counter reset/roll-over
0110: On-chip Event 1 (hrtim_bm_ck1), acting as a burst mode counter clock
0111: On-chip Event 2 (hrtim_bm_ck2) acting as a burst mode counter clock
1000: On-chip Event 3 (hrtim_bm_ck3) acting as a burst mode counter clock
1001: On-chip Event 4 (hrtim_bm_ck4) acting as a burst mode counter clock
1010: Prescaled f_{HRTIM} clock (as per BMPRSC[3:0] setting)
Other codes reserved

Bit 1 **BMOM**: *Burst Mode operating mode*

This bit defines if the burst mode is entered once or if it is continuously operating.
0: Single-shot mode
1: Continuous operation

Bit 0 **BME**: *Burst Mode enable*

This bit starts the burst mode controller which becomes ready to receive the start trigger. Writing this bit to 0 causes a burst mode early termination.
0: Burst mode disabled
1: Burst mode enabled

36.5.48 HRTIM Burst Mode Trigger Register (HRTIM_BMTRGR)

Address offset: 0x3A4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OCHPEV	EEV8	EEV7	TDEEV8	TAEEV7	TECMP2	TECMP1	TEREP	TERST	TDCMP2	TDCMP1	TDREP	TDRST	TCCMP2	TCCMP1	TCREP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OCHPEV**: *On-chip Event*

A rising edge on an on-chip Event (see *Section : Burst mode triggers*) triggers a burst mode entry.

Bit 30 **EEV8**: *External Event 8 (TIMD filters applied)*

The external event 8 conditioned by TIMD filters is starting the burst mode operation.

Bit 29 **EEV7**: *External Event 7 (TIMA filters applied)*

The external event 7 conditioned by TIMA filters is starting the burst mode operation.

Bit 28 **TDEEV8**: *Timer D period following External Event 8*

The timer D period following an external event 8 (conditioned by TIMD filters) is starting the burst mode operation.

Bit 27 **TAEEV7**: *Timer A period following External Event 7*

The timer A period following an external event 7 (conditioned by TIMA filters) is starting the burst mode operation.

Bit 26 **TECMP2**: *Timer E Compare 2 event*

Refer to TACMP1 description

Bit 25 **TECMP1**: *Timer E Compare 1 event*

Refer to TACMP1 description

Bit 24 **TEREP**: *Timer E repetition*

Refer to TAREP description

Bit 23 **TERST**: *Timer E counter reset or roll-over*

Refer to TARST description

Bit 22 **TDCMP2**: *Timer D Compare 2 event*

Refer to TACMP1 description

Bit 21 **TDCMP1**: *Timer D Compare 1 event*

Refer to TACMP1 description

Bit 20 **TDREP**: *Timer D repetition*

Refer to TAREP description

Bit 19 **TDRST**: *Timer D reset or roll-over*

Refer to TARST description

Bit 18 **TCCMP2**: *Timer C Compare 2 event*

Refer to TACMP1 description

Bit 17 **TCCMP1**: *Timer C Compare 1 event*
Refer to TACMP1 description

Bit 16 **TCREP**: *Timer C repetition*
Refer to TAREP description

Bit 15 **TCRST**: *Timer C reset or roll-over*
Refer to TARST description

Bit 14 **TBCMP2**: *Timer B Compare 2 event*
Refer to TACMP1 description

Bit 13 **TBCMP1**: *Timer B Compare 1 event*
Refer to TACMP1 description

Bit 12 **TBREP**: *Timer B repetition*
Refer to TAREP description

Bit 11 **TBRST**: *Timer B reset or roll-over*
Refer to TARST description

Bit 10 **TACMP2**: *Timer A Compare 2 event*
Refer to TACMP1 description

Bit 9 **TACMP1**: *Timer A Compare 1 event*
The timer A compare 1 event is starting the burst mode operation.

Bit 8 **TAREP**: *Timer A repetition*
The Timer A repetition event is starting the burst mode operation.

Bit 7 **TARST**: *Timer A reset or roll-over*
The Timer A reset or roll-over event is starting the burst mode operation.

Bit 6 **MSTCMP4**: *Master Compare 4*
Refer to MSTCMP1 description

Bit 5 **MSTCMP3**: *Master Compare 3*
Refer to MSTCMP1 description

Bit 4 **MSTCMP2**: *Master Compare 2*
Refer to MSTCMP1 description

Bit 3 **MSTCMP1**: *Master Compare 1*
The master timer Compare 1 event is starting the burst mode operation.

Bit 2 **MSTREP**: *Master repetition*
The master timer repetition event is starting the burst mode operation.

Bit 1 **MSTRST**: *Master reset or roll-over*
The master timer reset and roll-over event is starting the burst mode operation.

Bit 0 **SW**: *Software start*
This bit is set by software and automatically reset by hardware.
When set, It starts the burst mode operation immediately.
This bit is not active if the burst mode is not enabled (BME bit is reset).

36.5.49 HRTIM Burst Mode Compare Register (HRTIM_BMCMPR)

Address offset: 0x3A8h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMCMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMCMP[15:0]**: *Burst mode compare value*

Defines the number of periods during which the selected timers are in idle state.

This register holds either the content of the preload register or the content of the active register if the preload is disabled.

Note: BMCMP[15:0] cannot be set to 0x0000 when using the f_{HRTIM} clock without a prescaler as the burst mode clock source (BMCLK[3:0] = 1010 and BMPRESC[3:0] = 0000).

36.5.50 HRTIM Burst Mode Period Register (HRTIM_BMPER)

Address offset: 0x3ACh

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMPER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMPER[15:0]**: *Burst mode Period*

Defines the burst mode repetition period.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

Note: The BMPER[15:0] must not be null when the burst mode is enabled.

36.5.51 HRTIM Timer External Event Control Register 1 (HRTIM_EECR1)

Address offset: 0x3B0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1]
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE3SNS[0]	EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **EE5FAST**: *External Event 5 Fast mode*
Refer to EE1FAST description

Bits 28:27 **EE5SNS[1:0]**: *External Event 5 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 26 **EE5POL**: *External Event 5 Polarity*
Refer to EE1POL description

Bits 25:24 **EE5SRC[1:0]**: *External Event 5 Source*
Refer to EE1SRC[1:0] description

Bit 23 **EE4FAST**: *External Event 4 Fast mode*
Refer to EE1FAST description

Bits 22:21 **EE4SNS[1:0]**: *External Event 4 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 20 **EE4POL**: *External Event 4 Polarity*
Refer to EE1POL description

Bits 19:18 **EE4SRC[1:0]**: *External Event 4 Source*
Refer to EE1SRC[1:0] description

Bit 17 **EE3FAST**: *External Event 3 Fast mode*
Refer to EE1FAST description

Bits 16:15 **EE3SNS[1:0]**: *External Event 3 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 14 **EE3POL**: *External Event 3 Polarity*
Refer to EE1POL description

Bits 13:12 **EE3SRC[1:0]**: *External Event 3 Source*
Refer to EE1SRC[1:0] description

Bit 11 **EE2FAST**: *External Event 2 Fast mode*
Refer to EE1FAST description

Bits 10:9 **EE2SNS[1:0]**: *External Event 2 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 8 **EE2POL**: *External Event 2 Polarity*
Refer to EE1POL description

Bits 7:6 **EE2SRC[1:0]**: *External Event 2 Source*
Refer to EE1SRC[1:0] description

Bit 5 **EE1FAST**: *External Event 1 Fast mode*
0: External Event 1 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External Event 1 is acting asynchronously on outputs (low latency mode)
Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set)

Bits 4:3 **EE1SNS[1:0]**: *External Event 1 Sensitivity*
00: On active level defined by EE1POL bit
01: Rising edge, whatever EE1POL bit value
10: Falling edge, whatever EE1POL bit value
11: Both edges, whatever EE1POL bit value

Bit 2 **EE1POL**: *External Event 1 Polarity*
This bit is only significant if EE1SNS[1:0] = 00.
0: External event is active high
1: External event is active low
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

Bits 1:0 **EE1SRC[1:0]**: *External Event 1 Source*
00: hrtim_evt11
01: hrtim_evt12
10: hrtim_evt13
11: hrtim_evt14
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

36.5.52 HRTIM Timer External Event Control Register 2 (HRTIM_EECR2)

Address offset: 0x3B4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8SNS[0]	EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:27 **EE10SNS[1:0]**: *External Event 10 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 26 **EE10POL**: *External Event 10 Polarity*
Refer to EE1POL description

Bits 25:24 **EE10SRC[1:0]**: *External Event 10 Source*
Refer to EE1SRC[1:0] description

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EE9SNS[1:0]**: *External Event 9 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 20 **EE9POL**: *External Event 9 Polarity*
Refer to EE1POL description

Bits 19:18 **EE9SRC[1:0]**: *External Event 9 Source*
Refer to EE1SRC[1:0] description

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **EE8SNS[1:0]**: *External Event 8 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 14 **EE8POL**: *External Event 8 Polarity*
Refer to EE1POL description

Bits 13:12 **EE8SRC[1:0]**: *External Event 8 Source*
Refer to EE1SRC[1:0] description

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE7SNS[1:0]**: *External Event 7 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 8 **EE7POL**: *External Event 7 Polarity*
Refer to EE1POL description

Bits 7:6 **EE7SRC[1:0]**: *External Event 7 Source*
Refer to EE1SRC[1:0] description

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE6SNS[1:0]**: *External Event 6 Sensitivity*
Refer to EE1SNS[1:0] description

Bit 2 **EE6POL**: *External Event 6 Polarity*
Refer to EE1POL description

Bits 1:0 **EE6SRC[1:0]**: *External Event 6 Source*
Refer to EE1SRC[1:0] description

36.5.53 HRTIM Timer External Event Control Register 3 (HRTIM_EECR3)

Address offset: 0x3B8h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.
rw	rw			rw	rw	rw	rw			rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 **EEVSD[1:0]**: External Event Sampling clock division
This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the External Event signal sampling clock (f_{EEVS}) used by the digital filters.
00: $f_{EEVS}=f_{HRTIM}$
01: $f_{EEVS}=f_{HRTIM}$ / 2
10: $f_{EEVS}=f_{HRTIM}$ / 4
11: $f_{EEVS}=f_{HRTIM}$ / 8

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **EE10F[3:0]**: External Event 10 filter
Refer to EE6F[3:0] description

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **EE9F[3:0]**: External Event 9 filter
Refer to EE6F[3:0] description

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:12 **EE8F[3:0]**: External Event 8 filter
Refer to EE6F[3:0] description

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE7F[3:0]**: External Event 7 filter
Refer to EE6F[3:0] description

Bits 4:5 Reserved, must be kept at reset value.

Bits 3:0 **EE6F[3:0]**: External Event 6 filter
This bitfield defines the frequency used to sample External Event 6 input and the length of the digital filter applied to hrtim_evt6. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.
0000: Filter disabled
0001: $f_{SAMPLING}= f_{HRTIM}$, N=2
0010: $f_{SAMPLING}= f_{HRTIM}$, N=4
0011: $f_{SAMPLING}= f_{HRTIM}$, N=8
0100: $f_{SAMPLING}= f_{EEVS}/2$, N=6
0101: $f_{SAMPLING}= f_{EEVS}/2$, N=8
0110: $f_{SAMPLING}= f_{EEVS}/4$, N=6
0111: $f_{SAMPLING}= f_{EEVS}/4$, N=8
1000: $f_{SAMPLING}= f_{EEVS}/8$, N=6
1001: $f_{SAMPLING}= f_{EEVS}/8$, N=8
1010: $f_{SAMPLING}= f_{EEVS}/16$, N=5
1011: $f_{SAMPLING}= f_{EEVS}/16$, N=6
1100: $f_{SAMPLING}= f_{EEVS}/16$, N=8
1101: $f_{SAMPLING}= f_{EEVS}/32$, N=5
1110: $f_{SAMPLING}= f_{EEVS}/32$, N=6
1111: $f_{SAMPLING}= f_{EEVS}/32$, N=8

36.5.54 HRTIM ADC Trigger 1 Register (HRTIM_ADC1R)

Address offset: 0x3BCh

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD1TEPER	AD1TEC4	AD1TEC3	AD1TEC2	AD1TDPER	AD1TDC4	AD1TDC3	AD1TDC2	AD1TCPER	AD1TCC4	AD1TCC3	AD1TCC2	AD1TBRST	AD1TBPER	AD1TBC4	AD1TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD1TBC2	AD1TARST	AD1TAPER	AD1TAC4	AD1TAC3	AD1TAC2	AD1EEV5	AD1EEV4	AD1EEV3	AD1EEV2	AD1EEV1	AD1MPER	AD1MC4	AD1MC3	AD1MC2	AD1MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 These bits select the trigger source for th ADC Trigger 1 output (hrtim_adc_trg1). Refer to HRTIM_ADC3R bits description for details

36.5.55 HRTIM ADC Trigger 2 Register (HRTIM_ADC2R)

Address offset: 0x3C0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AD2TERST	AD2TEC4	AD2TEC3	AD2TEC2	AD2TDRST	AD2TDPER	AD2TDC4	AD2TDC3	AD2TDC2	AD2TCRST	AD2TCPER	AD2TCC4	AD2TCC3	AD2TCC2	AD2TBPER	AD2TBC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD2TBC3	AD2TBC2	AD2TAPER	AD2TAC4	AD2TAC3	AD2TAC2	AD2EEV10	AD2EEV9	AD2EEV8	AD2EEV7	AD2EEV6	AD2MPER	AD2MC4	AD2MC3	AD2MC2	AD2MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 These bits select the trigger source for th ADC Trigger 2 output (hrtim_adc_trg2). Refer to HRTIM_ADC4R bits description for details

36.5.56 HRTIM ADC Trigger 3 Register (HRTIM_ADC3R)

Address offset: 0x3C4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC3TEPER	ADC3TEC4	ADC3TEC3	ADC3TEC2	ADC3TDPER	ADC3TDC4	ADC3TDC3	ADC3TDC2	ADC3TCPER	ADC3TCC4	ADC3TCC3	ADC3TCC2	ADC3TBRST	ADC3TBPER	ADC3TBC4	ADC3TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3TBC2	ADC3TARST	ADC3TAPER	ADC3TAC4	ADC3TAC3	ADC3TAC2	ADC3EEV5	ADC3EEV4	ADC3EEV3	ADC3EEV2	ADC3EEV1	ADC3MPER	ADC3MC4	ADC3MC3	ADC3MC2	ADC3MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC3TEPER**: *ADC trigger 3 on Timer E Period*
Refer to ADC3TAPER description

Bit 30 **ADC3TEC4**: *ADC trigger 3 on Timer E Compare 4*
Refer to ADC3TAC2 description

Bit 29 **ADC3TEC3**: *ADC trigger 3 on Timer E Compare 3*
Refer to ADC3TAC2 description

Bit 28 **ADC3TEC2**: *ADC trigger 3 on Timer E Compare 2*
Refer to ADC3TAC2 description

Bit 27 **ADC3TDPER**: *ADC trigger 3 on Timer D Period*
Refer to ADC3TAPER description

Bit 26 **ADC3TDC4**: *ADC trigger 3 on Timer D Compare 4*
Refer to ADC3TAC2 description

Bit 25 **ADC3TDC3**: *ADC trigger 3 on Timer D Compare 3*
Refer to ADC3TAC2 description

Bit 24 **ADC3TDC2**: *ADC trigger 3 on Timer D Compare 2*
Refer to ADC3TAC2 description

Bit 23 **ADC3TCPER**: *ADC trigger 3 on Timer C Period*
Refer to ADC3TAPER description

Bit 22 **ADC3TCC4**: *ADC trigger 3 on Timer C Compare 4*
Refer to ADC3TAC2 description

Bit 21 **ADC3TCC3**: *ADC trigger 3 on Timer C Compare 3*
Refer to ADC3TAC2 description

Bit 20 **ADC3TCC2**: *ADC trigger 3 on Timer C Compare 2*
Refer to ADC3TAC2 description

Bit 19 **ADC3TBRST**: *ADC trigger 3 on Timer B Reset and counter roll-over*
Refer to ADC3TBRST description

Bit 18 **ADC3TBPER**: *ADC trigger 3 on Timer B Period*
Refer to ADC3TAPER description

Bit 17 **ADC3TBC4**: *ADC trigger 3 on Timer B Compare 4*
Refer to ADC3TAC2 description

Bit 16 **ADC3TBC3**: *ADC trigger 3 on Timer B Compare 3*
Refer to ADC3TAC2 description

Bit 15 **ADC3TBC2**: *ADC trigger 3 on Timer B Compare 2*
Refer to ADC3TAC2 description

Bit 14 **ADC3TARST**: *ADC trigger 3 on Timer A Reset and counter roll-over*
This bit enables the generation of an ADC Trigger upon Timer A reset and roll-over event, on ADC Trigger 1 output.

Bit 13 **ADC3TAPER**: *ADC trigger 3 on Timer A Period*
This bit enables the generation of an ADC Trigger upon Timer A period event, on ADC Trigger 3 output (hrtim_adc_trg3).

Bit 12 **ADC3TAC4**: *ADC trigger 3 on Timer A Compare 4*
Refer to ADC3TAC2 description

Bit 11 **ADC3TAC3**: *ADC trigger 3 on Timer A Compare 3*
Refer to ADC3TAC2 description

Bit 10 **ADC3TAC2**: *ADC trigger 3 on Timer A Compare 2*
This bit enables the generation of an ADC Trigger upon Timer A Compare 2 event, on ADC Trigger 3 output (hrtim_adc_trg3).

Bit 9 **ADC3EEV5**: *ADC trigger 3 on External Event 5*
Refer to ADC3EEV1 description

Bit 8 **ADC3EEV4**: *ADC trigger 3 on External Event 4*
Refer to ADC3EEV1 description

Bit 7 **ADC3EEV3**: *ADC trigger 3 on External Event 3*
Refer to ADC3EEV1 description

Bit 6 **ADC3EEV2**: *ADC trigger 3 on External Event 2*
Refer to ADC3EEV1 description

Bit 5 **ADC3EEV1**: *ADC trigger 3 on External Event 1*
This bit enables the generation of an ADC Trigger upon External event 1, on ADC Trigger 3 output (hrtim_adc_trg3).

Bit 4 **ADC3MPER**: *ADC trigger 3 on Master Period*
This bit enables the generation of an ADC Trigger upon Master timer period event, on ADC Trigger 3 output (hrtim_adc_trg3).

Bit 3 **ADC3MC4**: *ADC trigger 3 on Master Compare 4*
Refer to ADC3MC1 description

Bit 2 **ADC3MC3**: *ADC trigger 3 on Master Compare 3*
Refer to ADC3MC1 description

Bit 1 **ADC3MC2**: *ADC trigger 3 on Master Compare 2*
Refer to ADC3MC1 description

Bit 0 **ADC3MC1**: *ADC trigger 3 on Master Compare 1*
This bit enables the generation of an ADC Trigger upon Master Compare 1 event, on ADC Trigger 3 output (hrtim_adc_trg3).

36.5.57 HRTIM ADC Trigger 4 Register (HRTIM_ADC4R)

Address offset: 0x3C8h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC4TERST	ADC4TEC4	ADC4TEC3	ADC4TEC2	ADC4TDRST	ADC4TDPER	ADC4TDC4	ADC4TDC3	ADC4TDC2	ADC4TCRST	ADC4TCPER	ADC4TCC4	ADC4TCC3	ADC4TCC2	ADC4TBPER	ADC4TBC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC4TBC3	ADC4TBC2	ADC4TAPER	ADC4TAC4	ADC4TAC3	ADC4TAC2	ADC4EEV10	ADC4EEV9	ADC4EEV8	ADC4EEV7	ADC4EEV6	ADC4MPER	ADC4MC4	ADC4MC3	ADC4MC2	ADC4MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC4TERST**: *ADC trigger 4 on Timer E Reset and counter roll-over* (1)
Refer to ADC4TCRST description

Bit 30 **ADC4TEC4**: *ADC trigger 4 on Timer E Compare 4*
Refer to ADC4TAC2 description

Bit 29 **ADC4TEC3**: *ADC trigger 4 on Timer E Compare 3*
Refer to ADC4TAC2 description

Bit 28 **ADC4TEC2**: *ADC trigger 4 on Timer E Compare 2*
Refer to ADC4TAC2 description

Bit 27 **ADC4TDRST**: *ADC trigger 4 on Timer D Reset and counter roll-over* (1)
Refer to ADC4TCRST description

Bit 26 **ADC4TDPER**: *ADC trigger 4 on Timer D Period*
Refer to ADC4TAPER description

Bit 25 **ADC4TDC4**: *ADC trigger 4 on Timer D Compare 4*
Refer to ADC4TAC2 description

Bit 24 **ADC4TDC3**: *ADC trigger 4 on Timer D Compare 3*
Refer to ADC4TAC2 description

Bit 23 **ADC4TDC2**: *ADC trigger 2 on Timer D Compare 2*
Refer to ADC4TAC2 description

Bit 22 **ADC4TCRST**: *ADC trigger 4 on Timer C Reset and counter roll-over* (1)
This bit enables the generation of an ADC Trigger upon Timer C reset and roll-over event, on ADC Trigger 4 output (hrtim_adc_trg4).

Bit 21 **ADC4TCPER**: *ADC trigger 4 on Timer C Period*
Refer to ADC4TAPER description

Bit 20 **ADC4TCC4**: *ADC trigger 4 on Timer C Compare 4*
Refer to ADC4TAC2 description

Bit 19 **ADC4TCC3**: *ADC trigger 4 on Timer C Compare 3*
Refer to ADC4TAC2 description

Bit 18 **ADC4TCC2**: *ADC trigger 4 on Timer C Compare 2*
Refer to ADC4TAC2 description

Bit 17 **ADC4TBPER**: *ADC trigger 4 on Timer B Period*
Refer to ADC4TAPER description

Bit 16 **ADC4TBC4**: *ADC trigger 4 on Timer B Compare 4*
Refer to ADC4TAC2 description

Bit 15 **ADC4TBC3**: *ADC trigger 4 on Timer B Compare 3*
Refer to ADC4TAC2 description

Bit 14 **ADC4TBC2**: *ADC trigger 4 on Timer B Compare 2*
Refer to ADC4TAC2 description

Bit 13 **ADC4TAPER**: *ADC trigger 4 on Timer A Period*
This bit enables the generation of an ADC Trigger upon Timer A event, on ADC Trigger 4 output (hrtim_adc_trg4).

Bit 12 **ADC4TAC4**: *ADC trigger 4 on Timer A Compare 4*
Refer to ADC4TAC2 description

Bit 11 **ADC4TAC3**: *ADC trigger 4 on Timer A Compare 3*
Refer to ADC4TAC2 description

Bit 10 **ADC4TAC2**: *ADC trigger 4 on Timer A Compare 2*
This bit enables the generation of an ADC Trigger upon Timer A Compare 2, on ADC Trigger 4 output (hrtim_adc_trg4).

Bit 9 **ADC4EEV10**: *ADC trigger 4 on External Event 10* (1)
Refer to ADC4EEV6 description

Bit 8 **ADC4EEV9**: *ADC trigger 4 on External Event 9* (1)
Refer to ADC4EEV6 description

Bit 7 **ADC4EEV8**: *ADC trigger 4 on External Event 8* (1)
Refer to ADC4EEV6 description

Bit 6 **ADC4EEV7**: *ADC trigger 4 on External Event 7* (1)
Refer to ADC4EEV6 description

Bit 5 **ADC4EEV6**: *ADC trigger 4 on External Event 6* (1)
This bit enables the generation of an ADC Trigger upon external event 6, on ADC Trigger 4 output (hrtim_adc_trg4).

Bit 4 **ADC4MPER**: *ADC trigger 4 on Master Period*
This bit enables the generation of an ADC Trigger upon Master period event, on ADC Trigger 4 output (hrtim_adc_trg4).

Bit 3 **ADC4MC4**: *ADC trigger 4 on Master Compare 4*
Refer to ADC4MC1 description

Bit 2 **ADC4MC3**: *ADC trigger 4 on Master Compare 3*
Refer to ADC4MC1 description

Bit 1 **ADC4MC2**: *ADC trigger 4 on Master Compare 2*
Refer to ADC4MC1 description

Bit 0 **ADC4MC1**: *ADC trigger 4 on Master Compare 1*
This bit enables the generation of an ADC Trigger upon Master Compare 1 event, on ADC Trigger 4 output (hrtim_adc_trg4).

1. These triggers are differing from HRTIM_ADC1R/HRTIM_ADC3R to HRTIM_ADC2R/HRTIM_ADC4R.

36.5.58 HRTIM Fault Input Register 1 (HRTIM_FLTINR1)

Address offset: 0x3D0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4LCK	FLT4F[3:0]				FLT4SRC	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC	FLT3P	FLT3E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT2LCK	FLT2F[3:0]				FLT2SRC	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC	FLT1P	FLT1E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4LCK**: Fault 4 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 30:27 **FLT4F[3:0]**: Fault 4 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 26 **FLT4SRC**: Fault 4 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 25 **FLT4P**: Fault 4 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register

Bit 24 **FLT4E**: Fault 4 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

Bit 23 **FLT3LCK**: Fault 3 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 22:19 **FLT3F[3:0]**: Fault 3 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 18 **FLT3SRC**: Fault 3 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 17 **FLT3P**: Fault 3 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register

Bit 16 **FLT3E**: Fault 3 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

Bit 15 **FLT2LCK**: Fault 2 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 14:11 **FLT2F[3:0]**: Fault 2 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 10 **FLT2SRC**: Fault 2 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 9 **FLT2P**: Fault 2 polarity
Refer to FLT2P description in HRTIM_FLTINR2 register

Bit 8 **FLT2E**: Fault 2 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

Bit 7 **FLT1LCK**: Fault 1 Lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register

Bits 6:3 **FLT1F[3:0]**: Fault 1 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register

Bit 2 **FLT1SRC**: Fault 1 source
Refer to FLT5SRC description in HRTIM_FLTINR2 register

Bit 1 **FLT1P**: Fault 1 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register

Bit 0 **FLT1E**: Fault 1 enable
Refer to FLT5E description in HRTIM_FLTINR2 register

36.5.59 HRTIM Fault Input Register 2 (HRTIM_FLTINR2)

Address offset: 0x3D4h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5LCK	FLT5F[3:0]				FLT5SRC	FLT5P	FLT5E
								rwo	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **FLTSD[1:0]**: Fault Sampling clock division

This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the fault signal sampling clock (f_{FLTS}) used by the digital filters.

00: $f_{FLTS}=f_{HRTIM}$

01: $f_{FLTS}=f_{HRTIM} / 2$

10: $f_{FLTS}=f_{HRTIM} / 4$

11: $f_{FLTS}=f_{HRTIM} / 8$

Note: This bitfield must be written prior to any of the FLTxE enable bits.

Bits 23:8 Reserved, must be kept at reset value.

Bit 7 **FLT5LCK**: Fault 5 Lock

The FLT5LCK bit modifies the write attributes of the fault programming bit, so that they can be protected against spurious write accesses.

This bit is write-once. Once it has been set, it cannot be modified till the next system reset.

0: FLT5E, FLT5P, FLT5SRC, FLT5F[3:0] bits are read/write.

1: FLT5E, FLT5P, FLT5SRC, FLT5F[3:0] bits can no longer be written (read-only mode)

Bits 6:3 **FLT5F[3:0]**: Fault 5 filter

This bitfield defines the frequency used to sample FLT5 input and the length of the digital filter applied to FLT5. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, FLT5 acts asynchronously
0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2
0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4
0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8
0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8

Note: This bitfield can be written only when FLT5E enable bit is reset.
This bitfield cannot be modified when FLT5LOCK has been programmed.

Bit 2 **FLT5SRC**: Fault 5 source

This bit selects the FAULT5 input source (refer to *Table 311* for connection details).

0: Fault 1 input is HRTIM_FLT5 input pin

1: Fault 1 input is hrtim_in_flt5 signal

Note: This bitfield can be written only when FLT5E enable bit is reset

Bit 1 **FLT5P**: Fault 5 polarity

This bit selects the FAULT5 input polarity.

0: Fault 5 input is active low

1: Fault 5 input is active high

Note: This bitfield can be written only when FLT5E enable bit is reset

Bit 0 **FLT5E**: Fault 5 enable

This bit enables the global FAULT5 input circuitry.

0: Fault 5 input disabled
1: Fault 5 input enabled

36.5.60 HRTIM Burst DMA Master timer update Register (HRTIM_BDMUPR)

Address offset: 0x3D8h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MCMP4**: *MCMP4R register update enable*
Refer to MCR description

Bit 8 **MCMP3**: *MCMP3R register update enable*
Refer to MCR description

Bit 7 **MCMP2**: *MCMP2R register update enable*
Refer to MCR description

Bit 6 **MCMP1**: *MCMP1R register update enable*
Refer to MCR description

Bit 5 **MREP**: *MREP register update enable*
Refer to MCR description

Bit 4 **MPER**: *MPER register update enable*
Refer to MCR description

Bit 3 **MCNT**: *MCNTR register update enable*
Refer to MCR description

Bit 2 **MDIER**: *MDIER register update enable*
Refer to MCR description

Bit 1 **MICR**: *MICR register update enable*
Refer to MCR description

Bit 0 **MCR**: *MCR register update enable*
This bit defines if the master timer MCR register is part of the list of registers to be updated by the Burst DMA.
0: MCR register is not updated by Burst DMA accesses
1: MCR register is updated by Burst DMA accesses

36.5.61 HRTIM Burst DMA Timerx update Register (HRTIM_BDTxUPR)

Address offset: 0x3DCh-0x3ECh

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMxFLTR	TIMxOUTR	TIMxCHPR	TIMxRSTR	TIMxEEFR2
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMxEEFR1	TIMxRST2R	TIMxSET2R	TIMxRST1R	TIMxSET1R	TIMxDTxR	TIMxCMP4	TIMxCMP3	TIMxCMP2	TIMxCMP1	TIMxREP	TIMxPER	TIMxCNT	TIMxDIER	TIMxICR	TIMxCR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **TIMxFLTR**: HRTIM_FLTxR register update enable
Refer to TIMxCR description

Bit 19 **TIMxOUTR**: HRTIM_OUTxR register update enable
Refer to TIMxCR description

Bit 18 **TIMxCHPR**: HRTIM_CHPxR register update enable
Refer to TIMxCR description

Bit 17 **TIMxRSTR**: HRTIM_RSTxR register update enable
Refer to TIMxCR description

Bit 16 **TIMxEEFR2**: HRTIM_EEFxR2 register update enable
Refer to TIMxCR description

Bit 15 **TIMxEEFR1**: HRTIM_EEFxR1 register update enable
Refer to TIMxCR description

Bit 14 **TIMxRST2R**: HRTIM_RST2xR register update enable
Refer to TIMxCR description

Bit 13 **TIMxSET2R**: HRTIM_SET2xR register update enable
Refer to TIMxCR description

Bit 12 **TIMxRST1R**: HRTIM_RST1xR register update enable
Refer to TIMxCR description

Bit 11 **TIMxSET1R**: HRTIM_SET1xR register update enable
Refer to TIMxCR description

Bit 10 **TIMxDTR**: HRTIM_DTxR register update enable
Refer to TIMxCR description

Bit 9 **TIMxCMP4**: HRTIM_CMP4xR register update enable
Refer to TIMxCR description

Bit 8 **TIMxCMP3**: HRTIM_CMP3xR register update enable
Refer to TIMxCR description

Bit 7 **TIMxCMP2**: HRTIM_CMP2xR register update enable
Refer to TIMxCR description

Bit 6 **TIMxCMP1**: HRTIM_CMP1xR register update enable
Refer to TIMxCR description

Bit 5 **TIMxREP**: HRTIM_REPxR register update enable
Refer to TIMxCR description

Bit 4 **TIMxPER**: HRTIM_PERxR register update enable
Refer to TIMxCR description

Bit 3 **TIMxCNT**: HRTIM_CNTxR register update enable
Refer to TIMxCR description

Bit 2 **TIMxDIER**: HRTIM_TIMxDIER register update enable
Refer to TIMxCR description

Bit 1 **TIMxICR**: HRTIM_TIMxICR register update enable
Refer to TIMxCR description

Bit 0 **TIMxCR**: HRTIM_TIMxCR register update enable
This bit defines if the master timer MCR register is part of the list of registers to be updated by the Burst DMA.
0: HRTIM_TIMxCR register is not updated by Burst DMA accesses
1: HRTIM_TIMxCR register is updated by Burst DMA accesses

36.5.62 HRTIM Burst DMA Data Register (HRTIM_BDMADR)

Address offset: 0x3F0h

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BDMADR[31:16]															
wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDMADR[15:0]															
wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo	wo

Bits 31:0 **BDMADR[31:0]**: Burst DMA Data register
Write accesses to this register triggers:
– the copy of the data value into the registers enabled in BDTxUPR and BDMUPR register bits
– the increment of the register pointer to the next location to be filled

36.5.63 HRTIM register map

The tables below summarize the HRTIM registers mapping. The address offsets in *Table 317* and *Table 318* are referred to in the base address offsets given in *Table 316*.

Table 316. RTIM global register map

Base address offset	Register
0x000 - 0x07F	Master timer
0x080 - 0x0FF	Timer A
0x100 - 0x17F	Timer B
0x180 - 0x1FF	Timer C
0x200 - 0x27F	Timer D
0x280 - 0x2FF	Timer E
0x300 - 0x37F	Reserved
0x380 - 0x3FF	Common registers

Table 317. HRTIM Register map and reset values: Master timer

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	HRTIM_MCR	BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	Res.	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN	SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		Res.	Res.	HALF	RETRIG	CONT	CKPSC[2:0]		
	Reset value	0	0	0		0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x0004	HRTIM_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP4	MCMP3	MCMP2	MCMP1
	Reset value																										0	0	0	0	0	0	0
0x0008	HRTIM_MICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
	Reset value																										0	0	0	0	0	0	0
0x000C	HRTIM_MDIER(T)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x0010	HRTIM_MCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 317. HRTIM Register map and reset values: Master timer (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0014	**HRTIM_MPER**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPER[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x0018	**HRTIM_MREP**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x001C	**HRTIM_MCMP1R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0024	**HRTIM_MCMP2R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	**HRTIM_MCMP3R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	**HRTIM_MCMP4R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 302 on page 1383*).

Table 318. HRTIM Register map and reset values: TIMx (x= A..E)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	HRTIM_TIMxCR	UPDGAT [3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	Res.	TxRSTU	TxREPU	Res.	DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	Res.	Res.	Res.	PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0		0	0	0	0	0	0				0	0	0	0	0	0	0
0x0004	HRTIM_TIMxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT	Res.	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	Res.	REP	CMP4	CMP3	CMP2	CMP1
	Reset value											0	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x0008	HRTIM_TIMxICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLYPRTC	RSTC	RSTx2C	SET2xC	RSTx1C	SET1xC	CPT2C	CPT1C	UPDC	Res.	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	Reset value																	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x000C	HRTIM_TIMxDIER(1)	Res.	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SET1xDE	CPT2DE	CPT1DE	UPDDE	Res.	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE	Res.	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	Res.	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	Reset value		0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x0010	HRTIM_CNTxR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTx[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	HRTIM_PERxR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERx[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x0018	HRTIM_REPxR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x001C	HRTIM_CMP1xR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP1x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	HRTIM_CMP1CxR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]								CMP1x[15:0]															
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	HRTIM_CMP2xR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP2x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	HRTIM_CMP3xR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP3x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	HRTIM_CMP4xR(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP4x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	HRTIM_CPT1xR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT1x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 318. HRTIM Register map and reset values: TIMx (x= A..E) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0034	HRTIM_CPT2xR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT2x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	HRTIM_DTxR(1)	DTFLKx	DTFSLKx	Res.	Res.	Res.	Res.	SDTFx	DTFx[8:0]									DTRLKx	DTRSLKx	Res.	DTPRSC[2:0]			SDTRx	DTRx[8:0]								
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	HRTIM_SETx1R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	HRTIM_RSTx1R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	HRTIM_SETx2R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	HRTIM_RSTx2R(1)	UPDATE	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	HRTIM_EEFxR1	Res.	Res.	Res.	EE5FLTR[3:0]				EE5LTCH	Res.	EE4FLTR[3:0]				EE4LTCH	Res.	EE3FLTR[3:0]				EE3LTCH	Res.	EE2FLTR[3:0]				EE2LTCH	Res.	EE1FLTR[3:0]				EE1LTCH
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x0050	HRTIM_EEFxR2	Res.	Res.	Res.	EE10FLTR[3:0]				EE10LTCH	Res.	EE9FLTR[3:0]				EE9LTCH	Res.	EE8FLTR[3:0]				EE8LTCH	Res.	EE7FLTR[3:0]				EE7LTCH	Res.	EE6FLTR[3:0]				EE6LTCH
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x0054	HRTIM_RSTAR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 318. HRTIM Register map and reset values: TIMx (x= A..E) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0054	HRTIM_RSTBR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0054	HRTIM_RSTCR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0054	HRTIM_RSTDR(1)	Res.	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0054	HRTIM_RSTER(1)	Res.	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNT10	EXTEVNT9	EXTEVNT8	EXTEVNT7	EXTEVNT6	EXTEVNT5	EXTEVNT4	EXTEVNT3	EXTEVNT2	EXTEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0058	HRTIM_CHPxR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRTPW [3:0]				CARDTY [2:0]			CARFRQ [3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x005C	HRTIM_CPT1ACR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	Res.	Res.	Res.	Res.	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x005C	HRTIM_CPT1BCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	Res.	Res.	Res.	Res.	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	HRTIM_CPT1CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	Res.	Res.	Res.	Res.	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	HRTIM_CPT1DCR	TECMP2	TECMP1	TE1RST	TE1SET	Res.	Res.	Res.	Res.	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 318. HRTIM Register map and reset values: TIMx (x= A..E) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x005C	HRTIM_ CPT1ECR	Res.	Res.	Res.	Res.	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2ACR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	Res.	Res.	Res.	Res.	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2BCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	Res.	Res.	Res.	Res.	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	Res.	Res.	Res.	Res.	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2DCR	TECMP2	TECMP1	TE1RST	TE1SET	Res.	Res.	Res.	Res.	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ CPT2ECR	Res.	Res.	Res.	Res.	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEV10CPT	EXEV9CPT	EXEV8CPT	EXEV7CPT	EXEV6CPT	EXEV5CPT	EXEV4CPT	EXEV3CPT	EXEV2CPT	EXEV1CPT	UPDCPT	SWCPT
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	HRTIM_OUTxR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM2	POL2	Res.	Res.	Res.	Res.	Res.	DLYPRT[2:0]		DLYPRTEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM1	POL1	Res.
	Reset value									0	0	0	0	0	0	0						0		0	0	0	0	0		0	0	0	
0x0068	HRTIM_FLTxR	FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
	Reset value	0																											0	0	0	0	0

1. This register can be preloaded (see *Table 302 on page 1383*).

Table 319. HRTIM Register map and reset values: Common functions

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**HRTIM_CR1**	Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0											0	0	0	0	0	0
0x0004	**HRTIM_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	Res.	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x008	**HRTIM_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
	Reset value															0												0	0	0	0	0	0
0x000C	**HRTIM_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
	Reset value															0												0	0	0	0	0	0
0x0010	**HRTIM_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
	Reset value															0												0	0	0	0	0	0
0x0014	**HRTIM_OENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x0018	**HRTIM_DISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x001C	**HRTIM_ODSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x0020	**HRTIM_BMCR**	BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEBM	TDBM	TCBM	TBBM	TABM	MTBM	Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
	Reset value	0										0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0024	**HRTIM_BMTRG**	OCHPEV	Res.	Res.	Res.	Res.	TECMP2	TECMP1	TEREP	TERST	TDCMP2	TDCMP1	TDREP	TDRST	TCCMP2	TCCMP1	TCREP	TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
	Reset value	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	**HRTIM_BMCMPR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMCMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 319. HRTIM Register map and reset values: Common functions (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x002C	**HRTIM_BMPER**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	**HRTIM_EECR1**	Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1:0]		EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	**HRTIM_EECR2**	Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1:0]		EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x0038	**HRTIM_EECR3**	Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE6SRC[1:0]	
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x003C	**HRTIM_ADC1R**(1)	AD1TEPER	AD1TEC4	AD1TEC3	AD1TEC2	AD1TDPER	AD1TDC4	AD1TDC3	AD1TDC2	AD1TCPER	AD1TCC4	AD1TCC3	AD1TCC2	AD1TBRST	AD1TBPER	AD1TBC4	AD1TBC3	AD1TBC2	AD1TARST	AD1TAPER	AD1TAC4	AD1TAC3	AD1TAC2	AD1EEV5	AD1EEV4	AD1EEV3	AD1EEV2	AD1EEV1	AD1MPER	AD1MC4	AD1MC3	AD1MC2	AD1MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	**HRTIM_ADC2R**(1)	AD2TERST	AD2TEC4	AD2TEC3	AD2TEC2	AD2TDRST	AD2TDPER	AD2TDC4	AD2TDC3	AD2TDC2	AD2TCRST	AD2TCPER	AD2TCC4	AD2TCC3	AD2TCC2	AD2TBPER	AD2TBC4	AD2TBC3	AD2TBC2	AD2TAPER	AD2TAC4	AD2TAC3	AD2TAC2	AD2EEV10	AD2EEV9	AD2EEV8	AD2EEV7	AD2EEV6	AD2MPER	AD2MC4	AD2MC3	AD2MC2	AD2MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	**HRTIM_ADC3R**(1)	ADC3TEPER	AD1TEC4	AD1TEC3	AD1TEC2	AD1TDPER	AD1TDC4	AD1TDC3	AD1TDC2	AD1TCPER	AD1TCC4	AD1TCC3	AD1TCC2	AD1TBRST	AD1TBPER	AD1TBC4	AD1TBC3	AD1TBC2	AD1TARST	AD1TAPER	AD1TAC4	AD1TAC3	AD1TAC2	AD1EEV5	AD1EEV4	AD1EEV3	AD1EEV2	AD1EEV1	AD1MPER	AD1MC4	AD1MC3	AD1MC2	AD1MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	**HRTIM_ADC4R**(1)	AD2TERST	AD2TEC4	AD2TEC3	AD2TEC2	AD2TDRST	AD2TDPER	AD2TDC4	AD2TDC3	AD2TDC2	AD2TCRST	AD2TCPER	AD2TCC4	AD2TCC3	AD2TCC2	AD2TBPER	AD2TBC4	AD2TBC3	AD2TBC2	AD2TAPER	AD2TAC4	AD2TAC3	AD2TAC2	AD2EEV10	AD2EEV9	AD2EEV8	AD2EEV7	AD2EEV6	AD2MPER	AD2MC4	AD2MC3	AD2MC2	AD2MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0050	**HRTIM_FLTINxR1**	FLT4LCK	FLT4F[3:0]				FLT4SRC	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC	FLT3P	FLT3E	FLT2LCK	FLT2F[3:0]				FLT2SRC	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC	FLT1P	FLT1E
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 319. HRTIM Register map and reset values: Common functions (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0054	HRTIM_FLTINxR2	Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT5LCK	FLT5F[3:0]				FLT5SRC	FLT5P	FLT5E
	Reset value							0	0																	0	0	0	0	0	0	0	0
0x0058	HRTIM_ BDMUPDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x005C	HRTIM_BDTAUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMAFLTR	TIMAOUTR	TIMACHPR	TIMARSTR	TIMAEEFR2	TIMAEEFR1	TIMARST2R	TIMASET2R	TIMARST1R	TIMASET1R	TIMADTxR	TIMACMP4	TIMACMP3	TIMACMP2	TIMACMP1	TIMAREP	TIMAPER	TIMACNT	TIMADIER	TIMAICR	TIMACR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	HRTIM_ BDTBUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMBFLTR	TIMBOUTR	TIMBCHPR	TIMBRSTR	TIMBEEFR2	TIMBEEFR1	TIMBRST2R	TIMBSET2R	TIMBRST1R	TIMBSET1R	TIMBDTxR	TIMBCMP4	TIMBCMP3	TIMBCMP2	TIMBCMP1	TIMBREP	TIMBPER	TIMBCNT	TIMBDIER	TIMBICR	TIMBCR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	HRTIM_ BDTCUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMCFLTR	TIMCOUTR	TIMCCHPR	TIMCRSTR	TIMCEEFR2	TIMCEEFR1	TIMCRST2R	TIMCSET2R	TIMCRST1R	TIMCSET1R	TIMCDTxR	TIMCCMP4	TIMCCMP3	TIMCCMP2	TIMCCMP1	TIMCREP	TIMCPER	TIMCCNT	TIMCDIER	TIMCICR	TIMCCR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068	HRTIM_ BDTDUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMDFLTR	TIMDOUTR	TIMDCHPR	TIMDRSTR	TIMDEEFR2	TIMDEEFR1	TIMDRST2R	TIMDSET2R	TIMDRST1R	TIMDSET1R	TIMDDTxR	TIMDCMP4	TIMDCMP3	TIMDCMP2	TIMDCMP1	TIMDREP	TIMDPER	TIMDCNT	TIMDDIER	TIMDICR	TIMDCR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x006C	HRTIM_ BDTEUPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEFLTR	TIMEOUTR	TIMECHPR	TIMERSTR	TIMEEEFR2	TIMEEEFR1	TIMERST2R	TIMESET2R	TIMERST1R	TIMESET1R	TIMEDTxR	TIMECMP4	TIMECMP3	TIMECMP2	TIMECMP1	TIMEREP	TIMEPER	TIMECNT	TIMEDIER	TIMEICR	TIMECR
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0070	HRTIM_BDMADR	BDMADR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 302 on page 1383*).

37 Advanced-control timers (TIM1/TIM8)

37.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

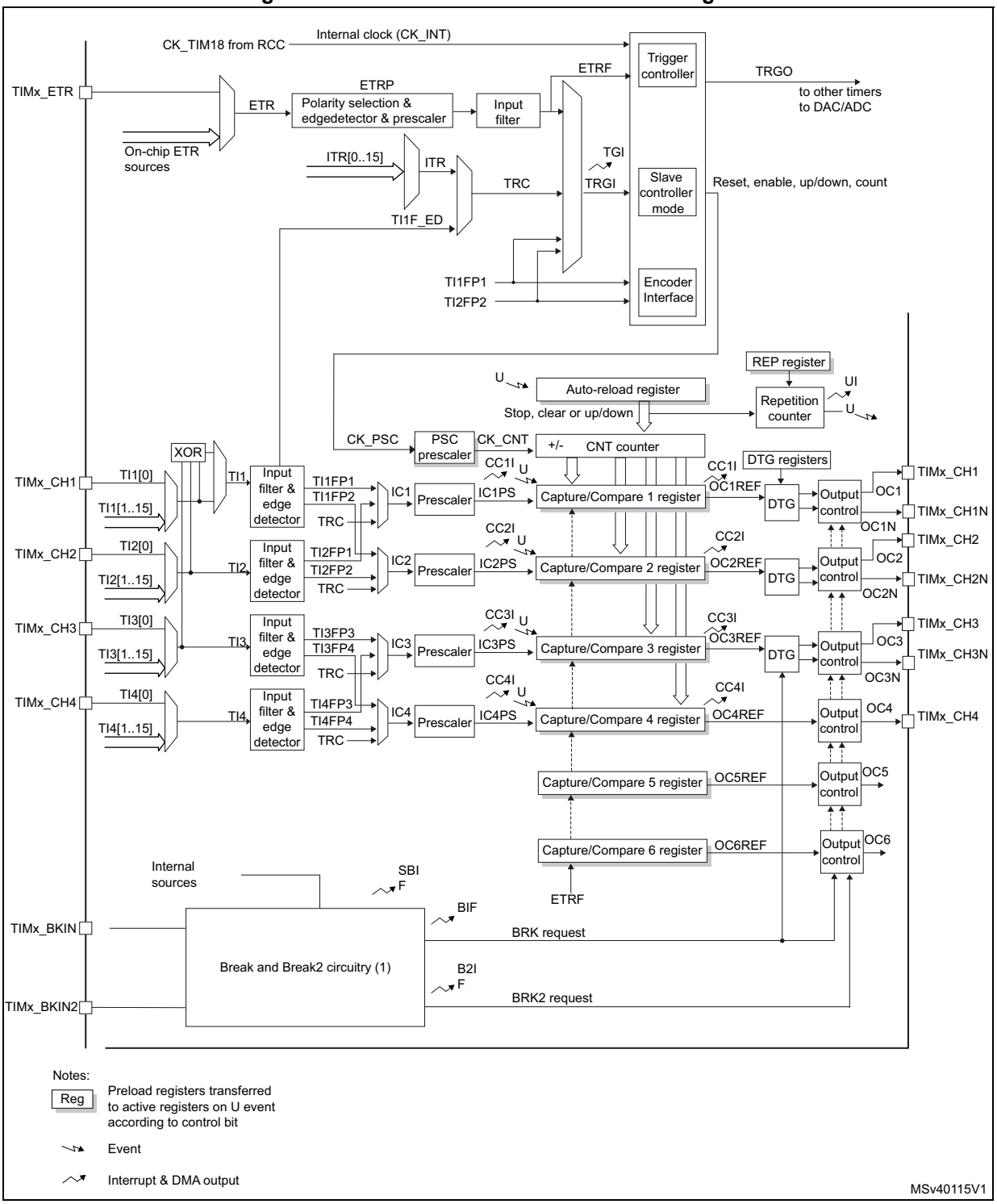
The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 37.3.26: Timer synchronization*.

37.2 TIM1/TIM8 main features

TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 343. Advanced-control timer block diagram



1. See *Figure 386: Break and Break2 circuitry overview* for details

37.3 TIM1/TIM8 functional description

37.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 344 and *Figure 345* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 344. Counter timing diagram with prescaler division change from 1 to 2

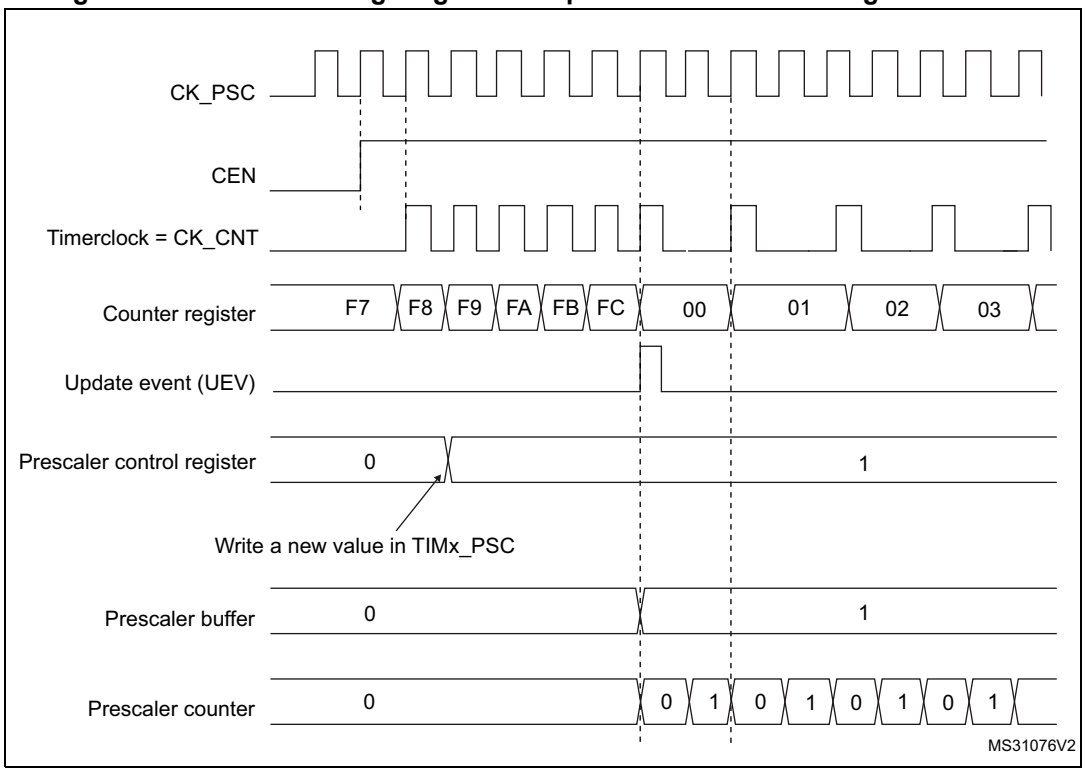
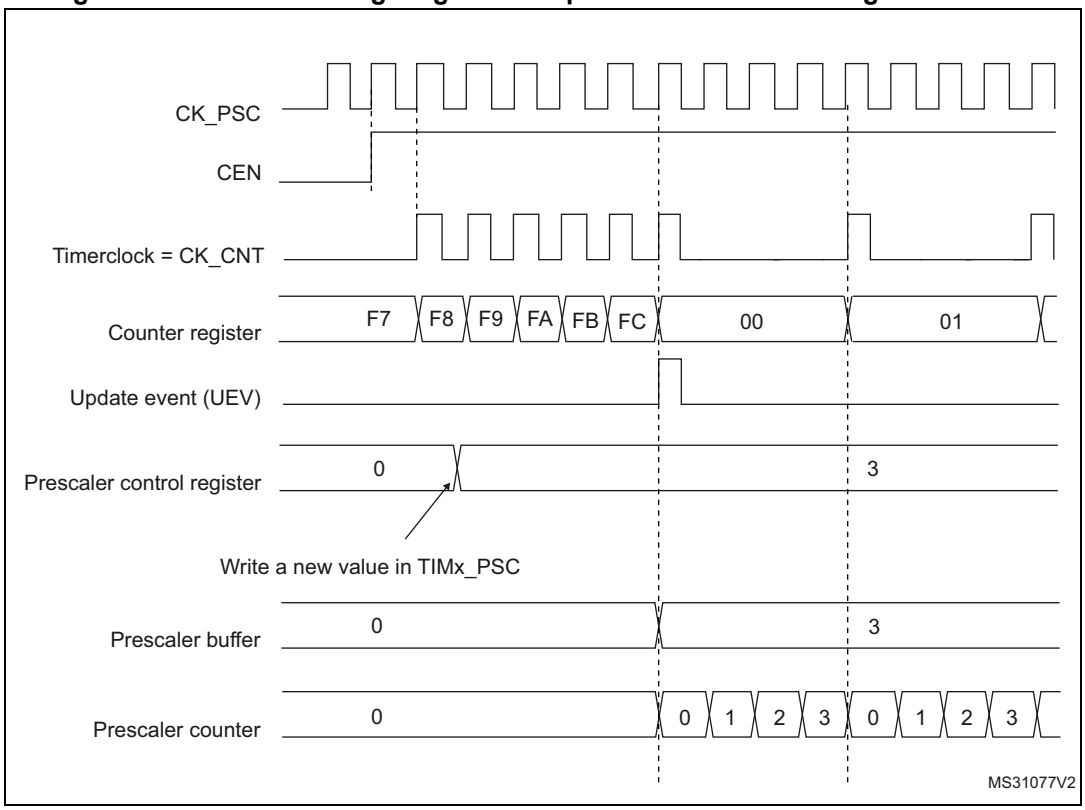


Figure 345. Counter timing diagram with prescaler division change from 1 to 4



37.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 346. Counter timing diagram, internal clock divided by 1

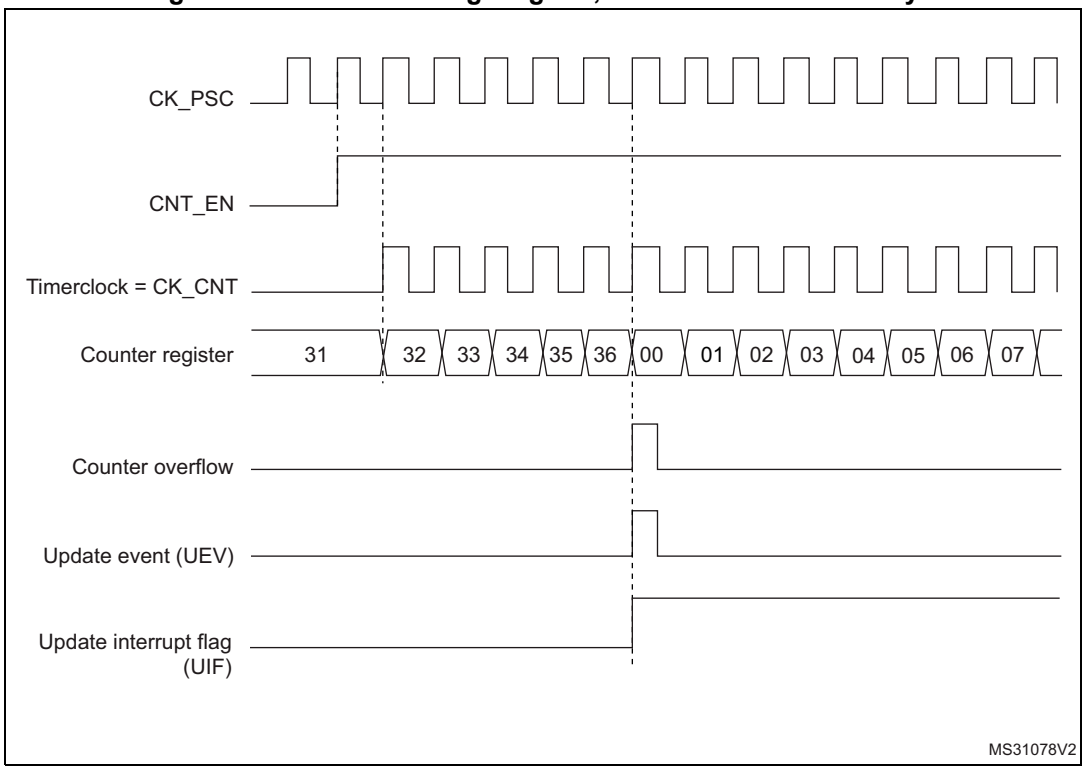


Figure 347. Counter timing diagram, internal clock divided by 2

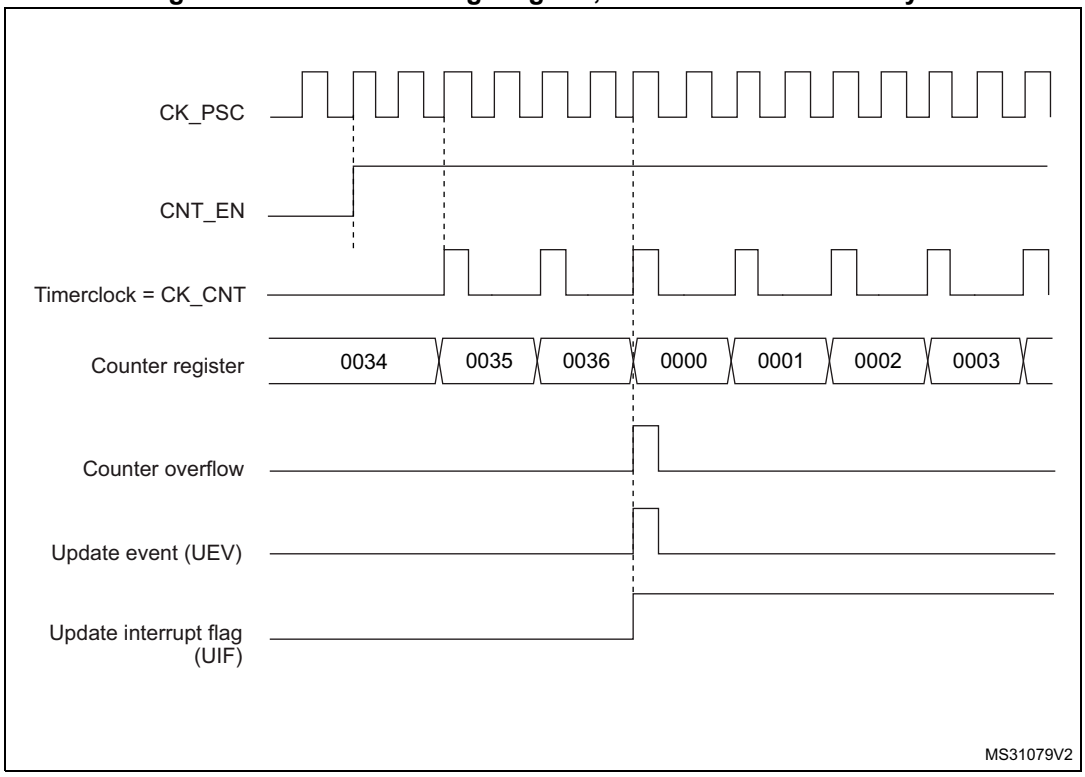


Figure 348. Counter timing diagram, internal clock divided by 4

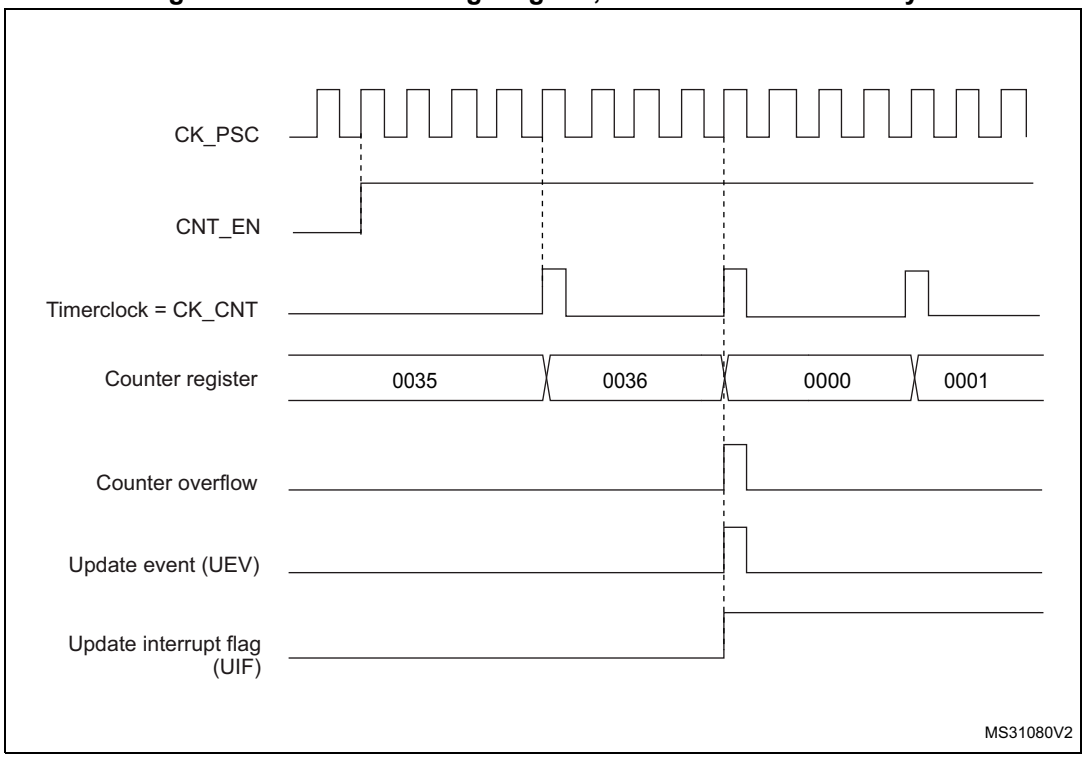


Figure 349. Counter timing diagram, internal clock divided by N

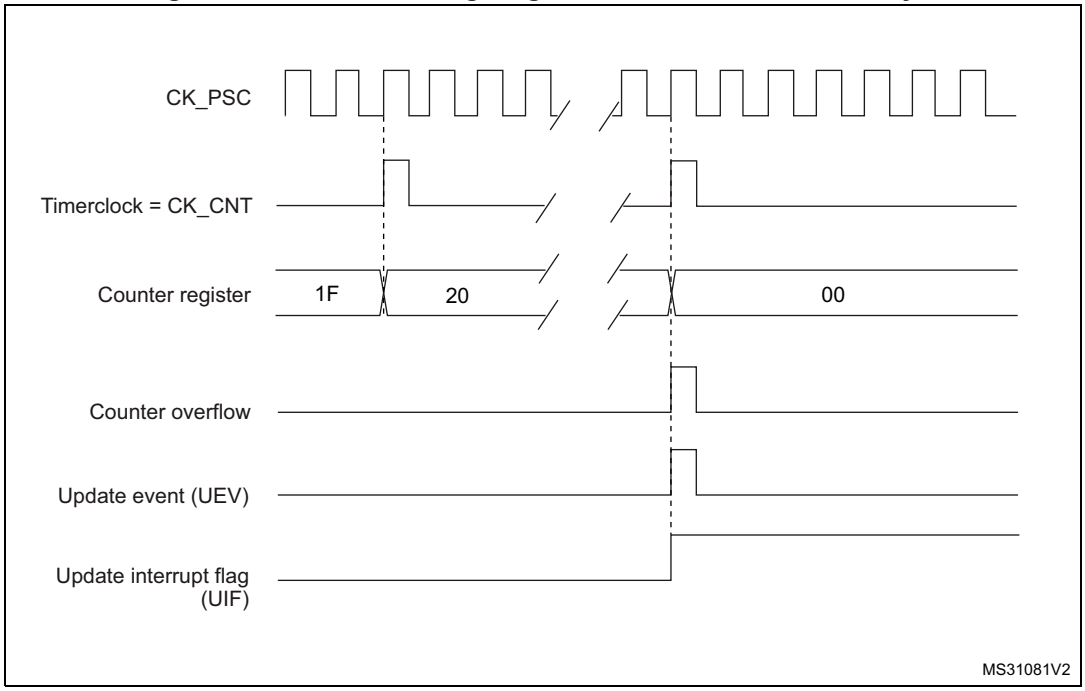


Figure 350. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

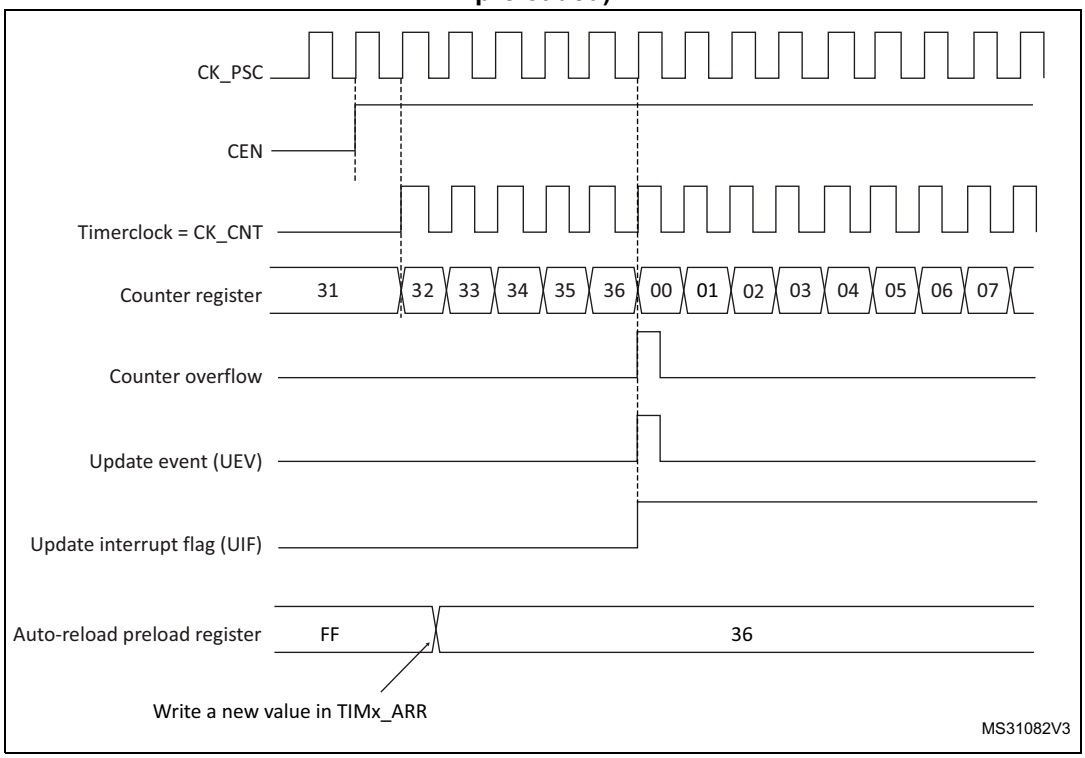
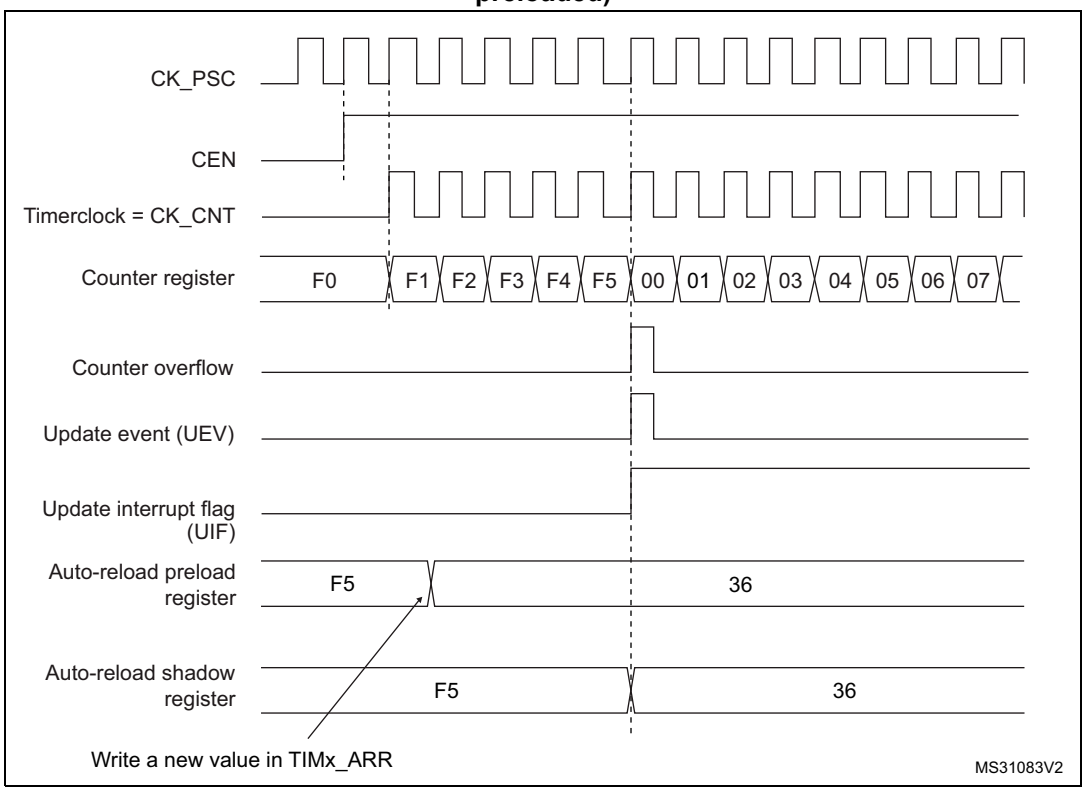


Figure 351. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 352. Counter timing diagram, internal clock divided by 1

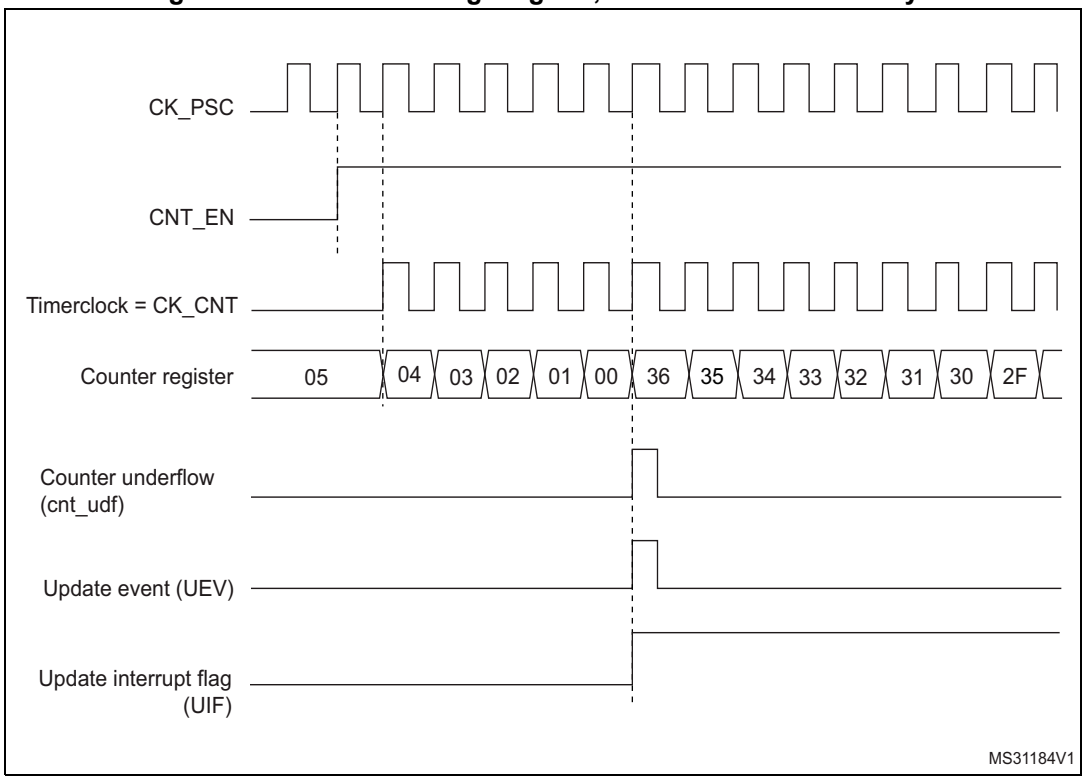


Figure 353. Counter timing diagram, internal clock divided by 2

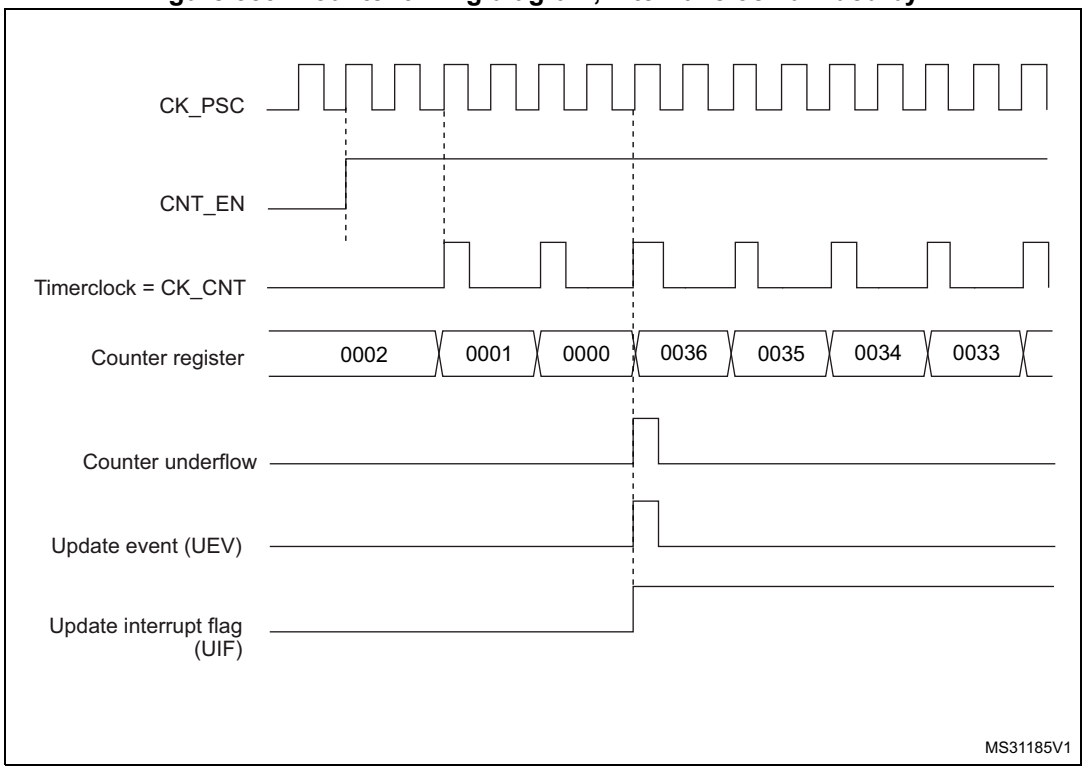


Figure 354. Counter timing diagram, internal clock divided by 4

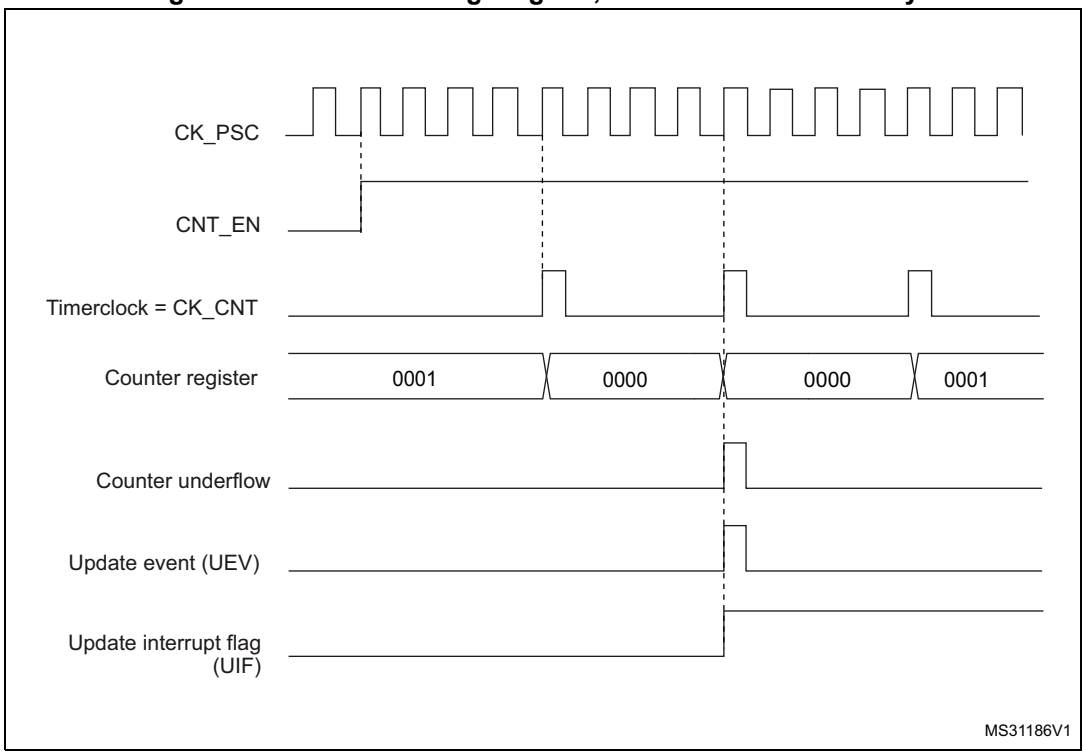


Figure 355. Counter timing diagram, internal clock divided by N

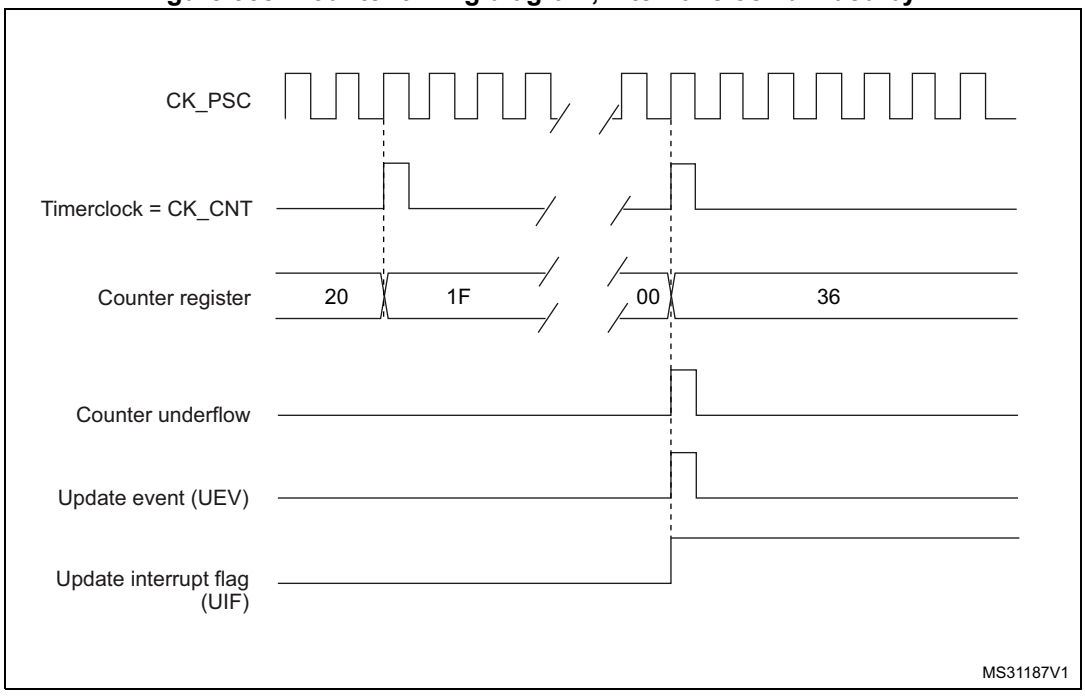
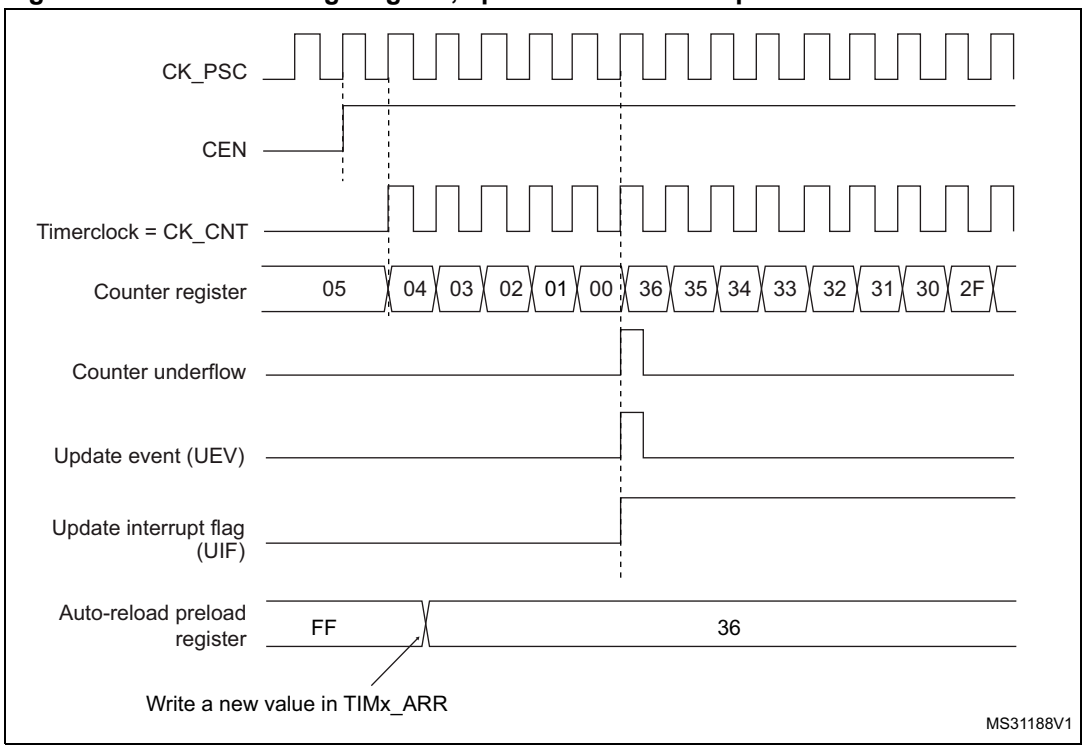


Figure 356. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or

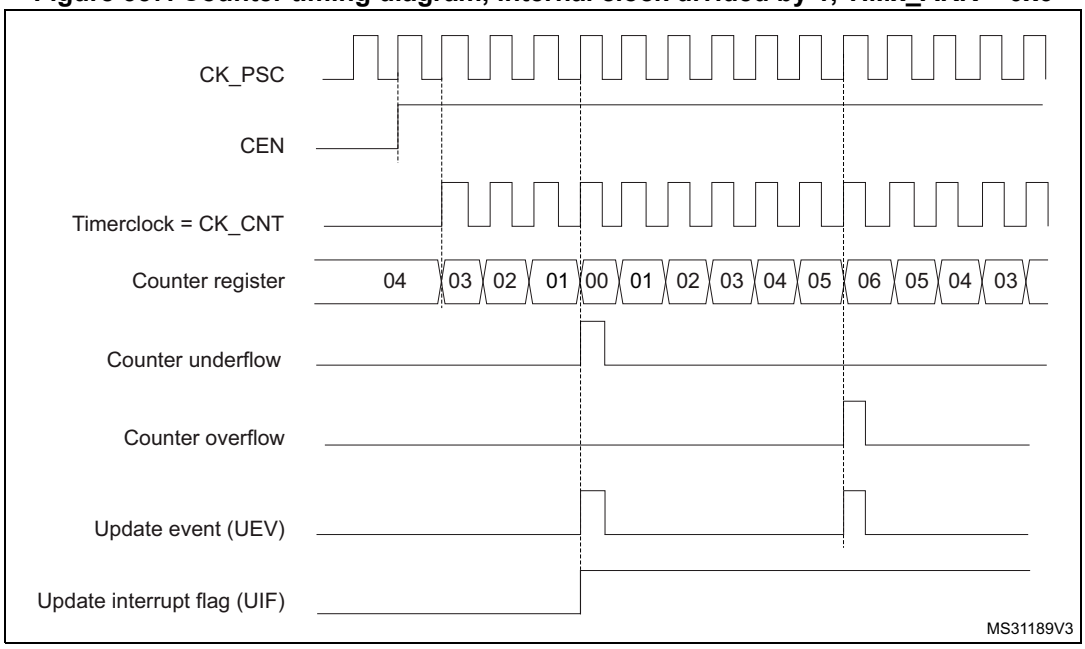
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 357. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 37.4: TIM1/TIM8 registers*).

Figure 358. Counter timing diagram, internal clock divided by 2

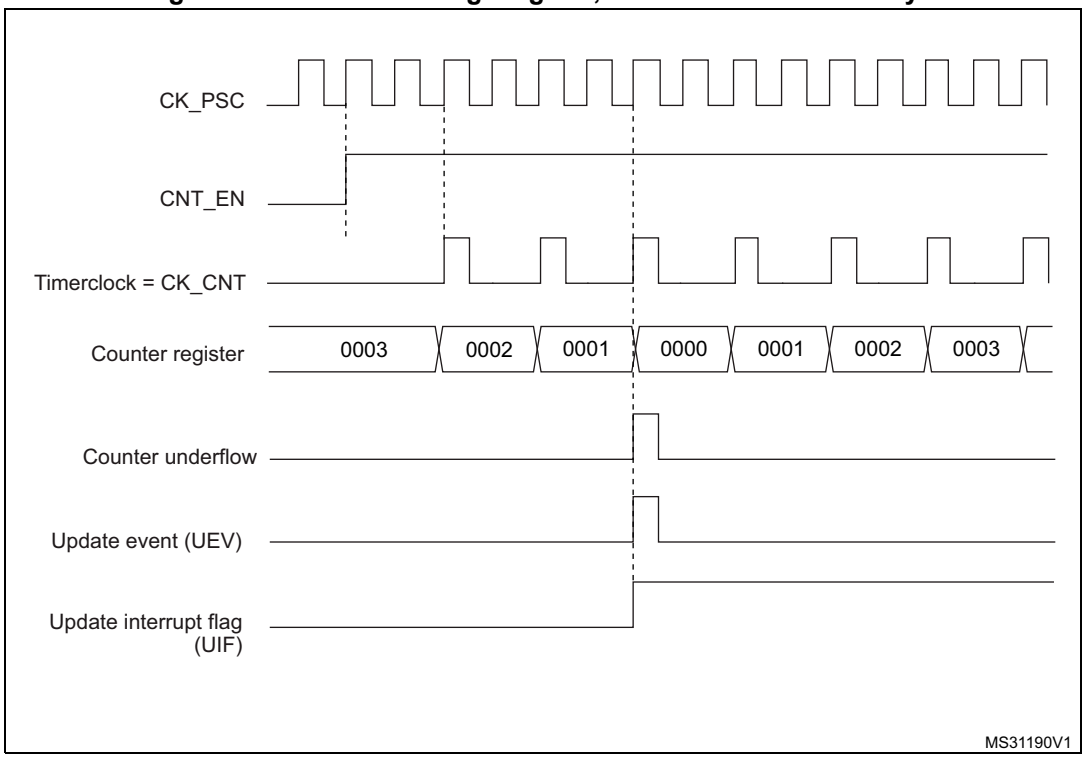


Figure 359. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

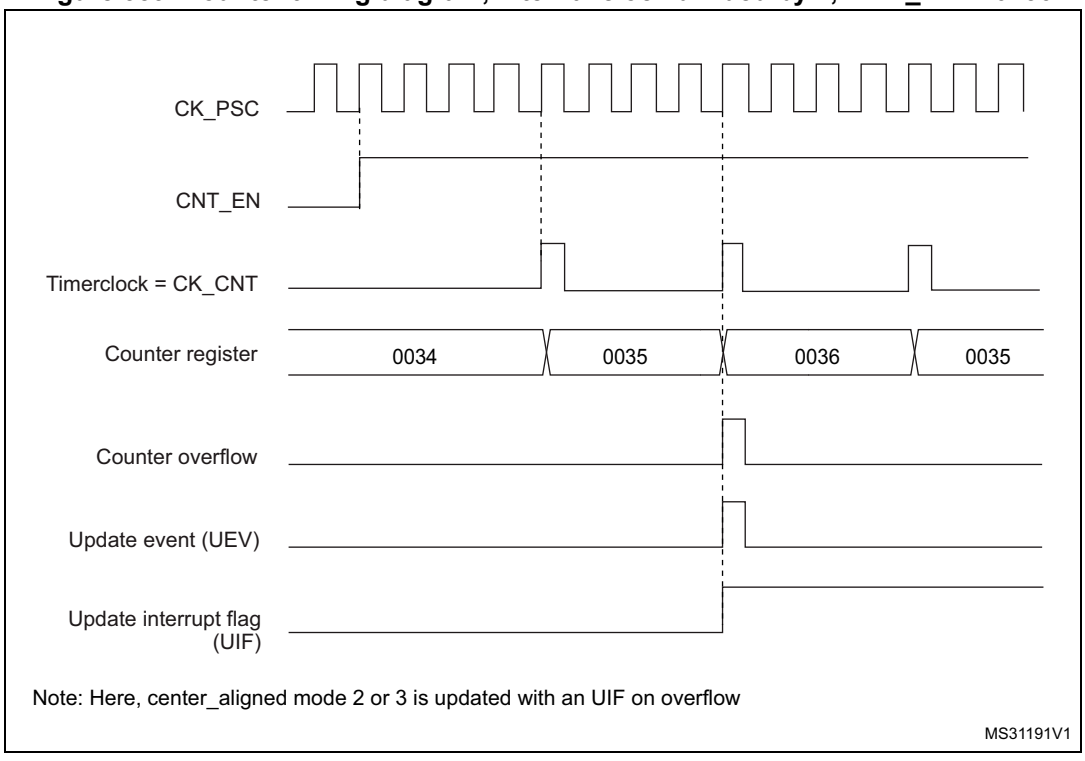


Figure 360. Counter timing diagram, internal clock divided by N

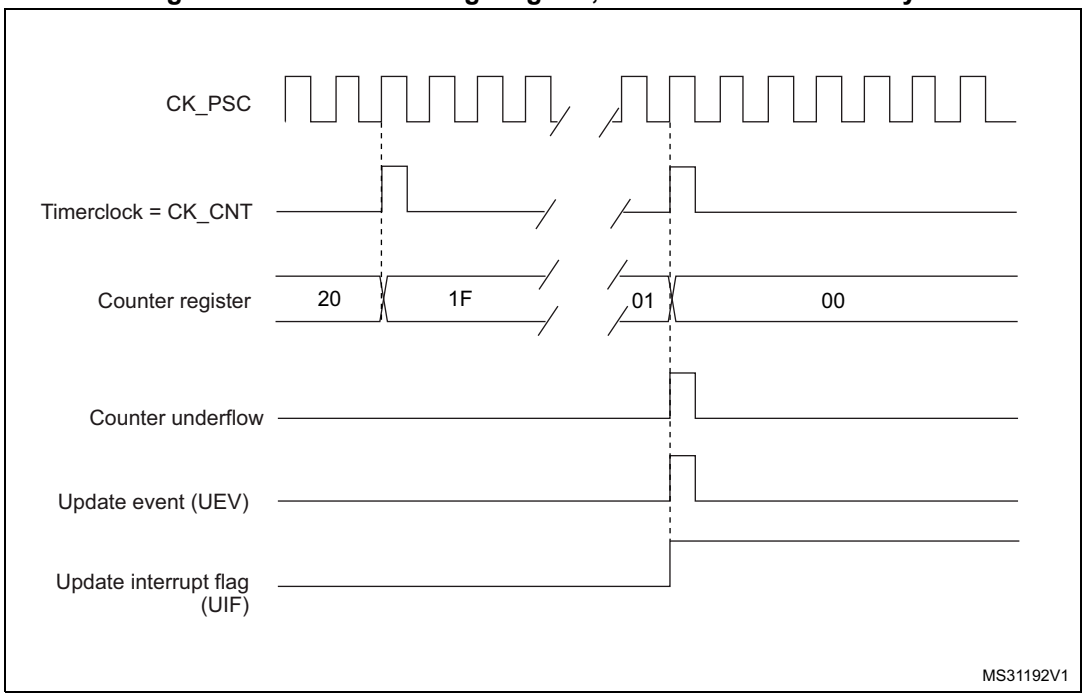


Figure 361. Counter timing diagram, update event with ARPE=1 (counter underflow)

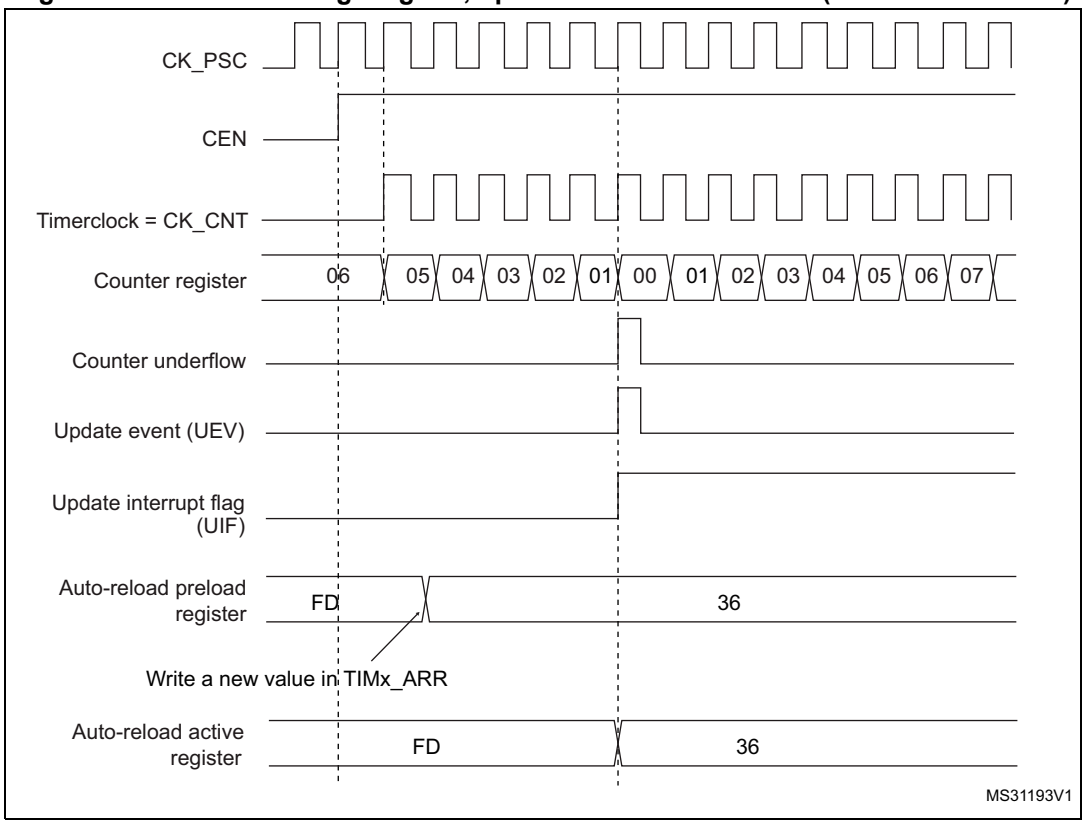
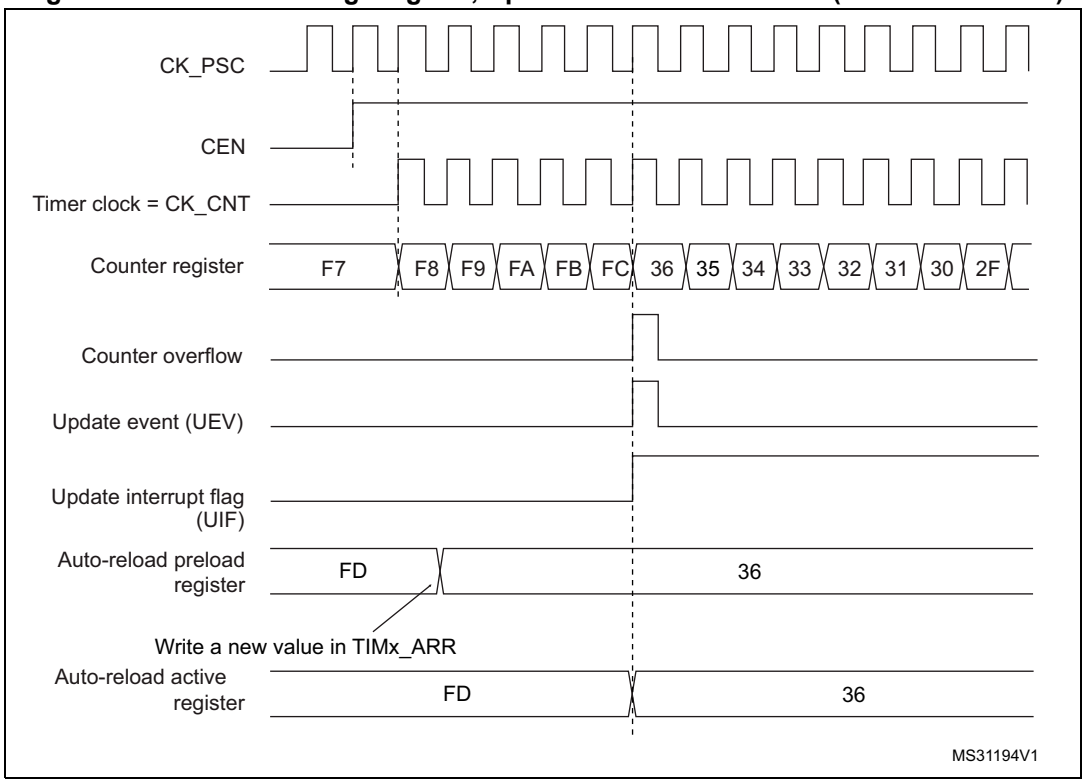


Figure 362. Counter timing diagram, Update event with ARPE=1 (counter overflow)



37.3.3 Repetition counter

Section 37.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

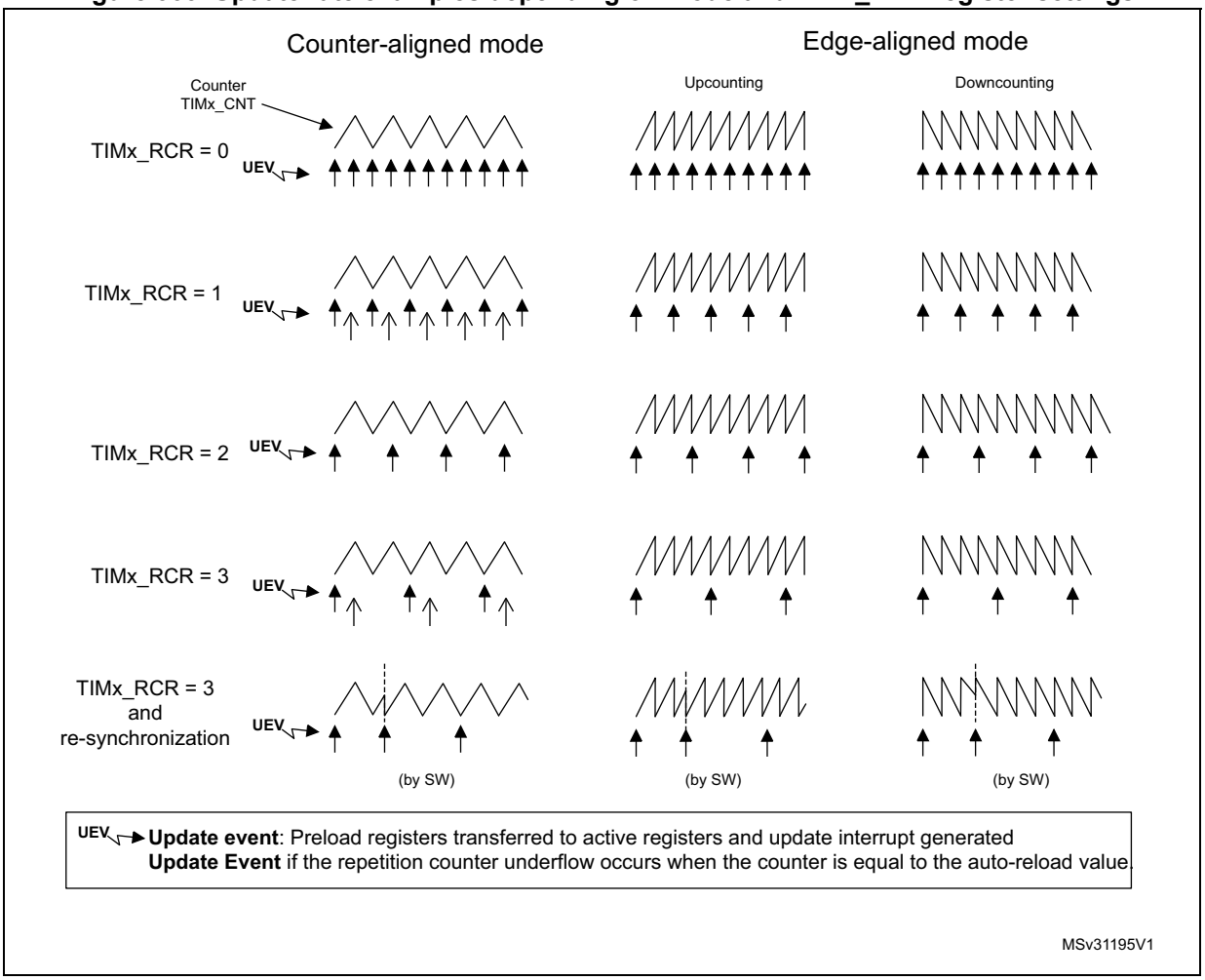
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 363*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the overflow. If the RCR was written after launching the counter, the UEV occurs on the underflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 363. Update rate examples depending on mode and TIMx_RCR register settings



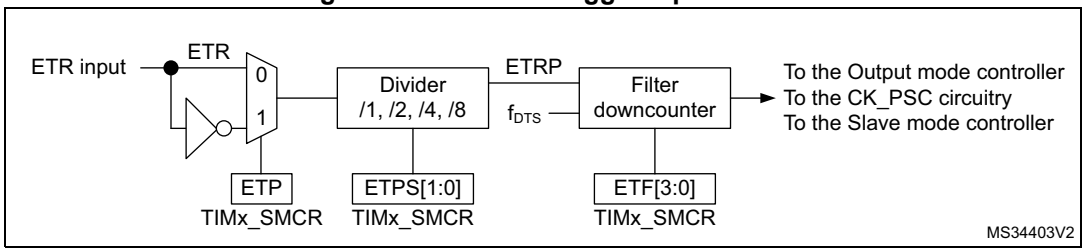
37.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 37.3.5*)
- trigger for the slave mode (see *Section 37.3.26*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 37.3.7*)

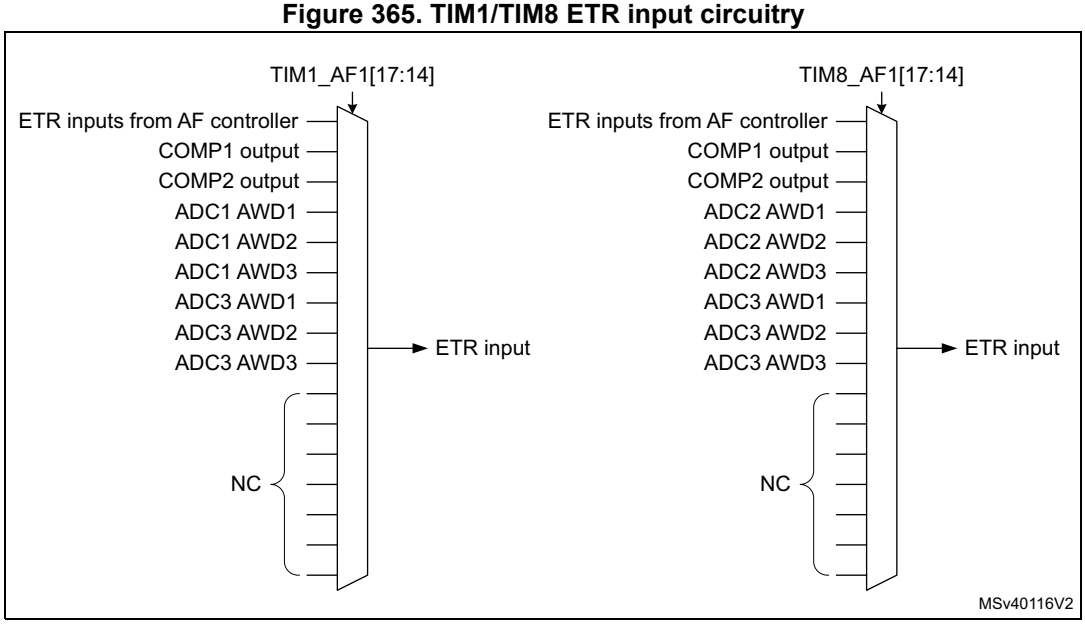
Figure 364 below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 364. External trigger input block



The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with the ETRSEL[3:0] bitfield.

Figure 365. TIM1/TIM8 ETR input circuitry



37.3.5 Clock selection

The counter clock can be provided by the following clock sources:

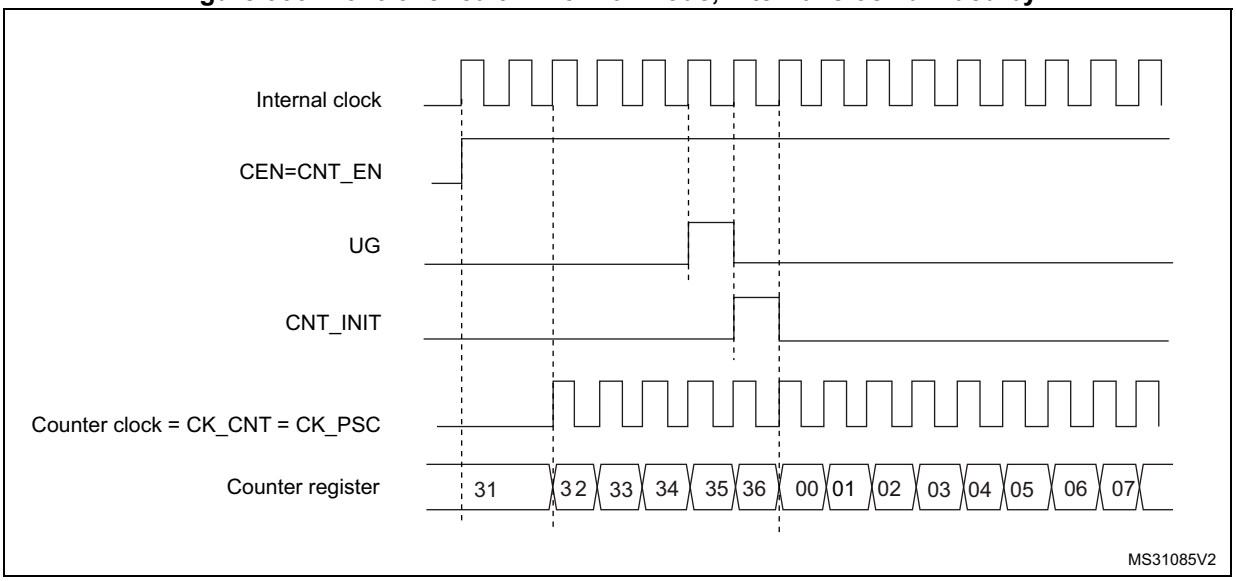
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 366 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

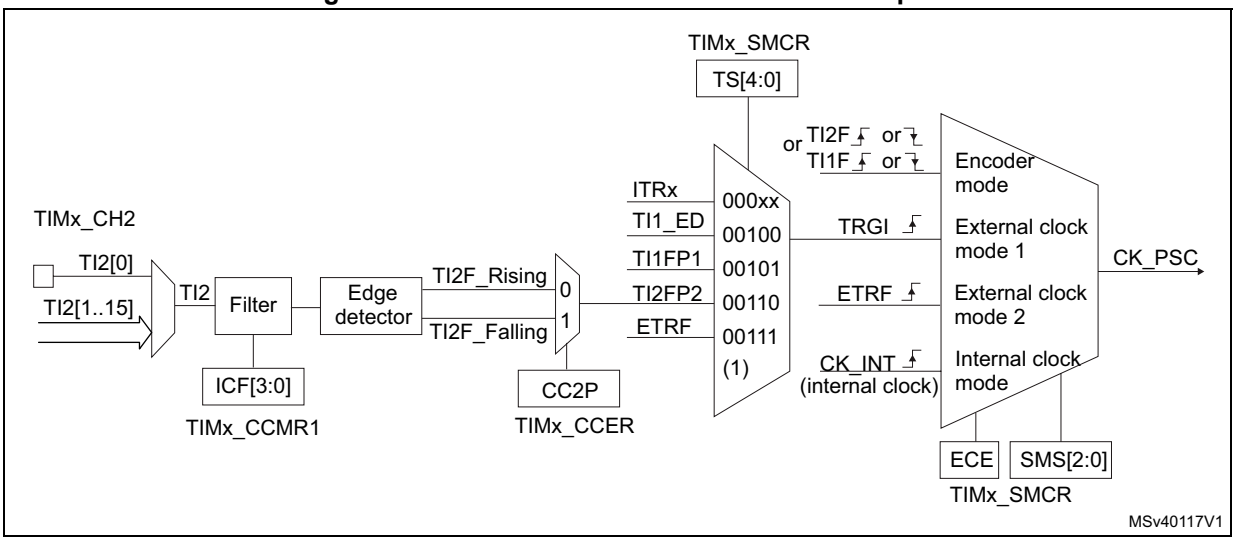
Figure 366. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 367. TI2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

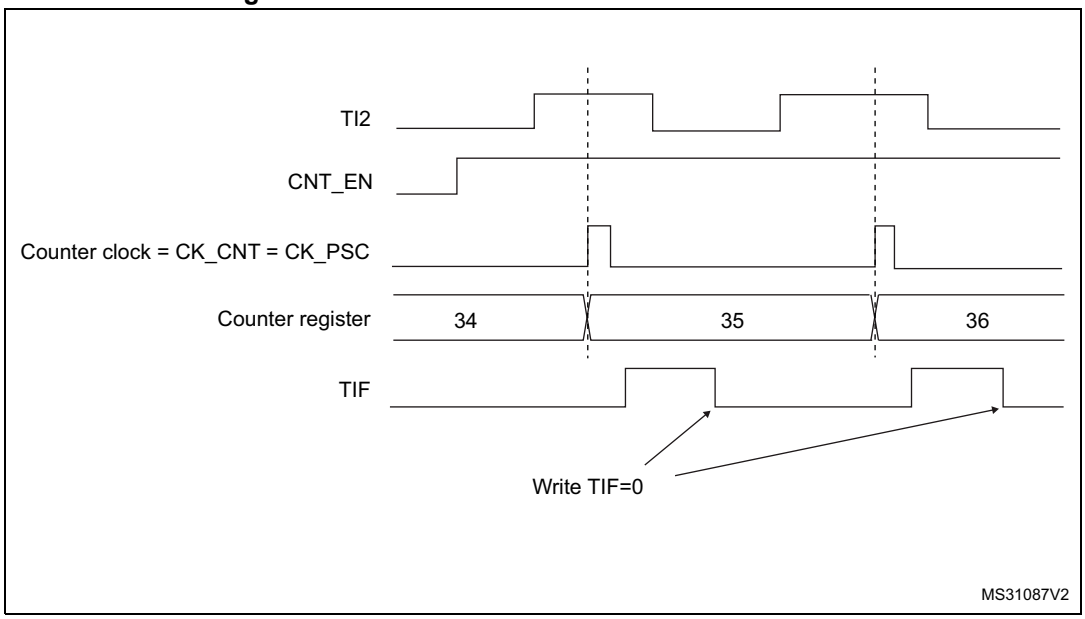
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 368. Control circuit in external clock mode 1



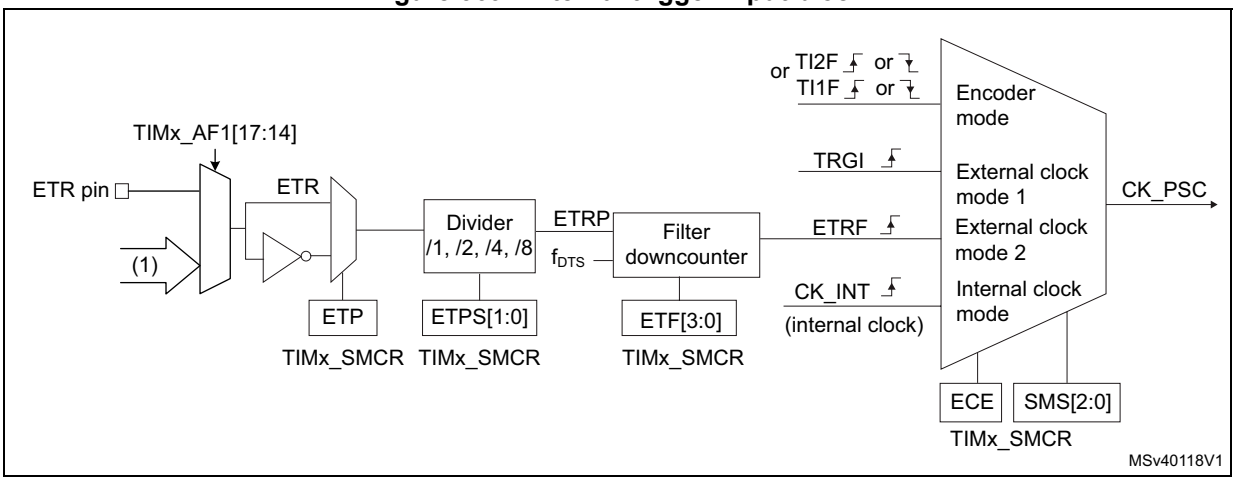
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 369* gives an overview of the external trigger input block.

Figure 369. External trigger input block



1. Refer to *Figure 365: TIM1/TIM8 ETR input circuitry*.

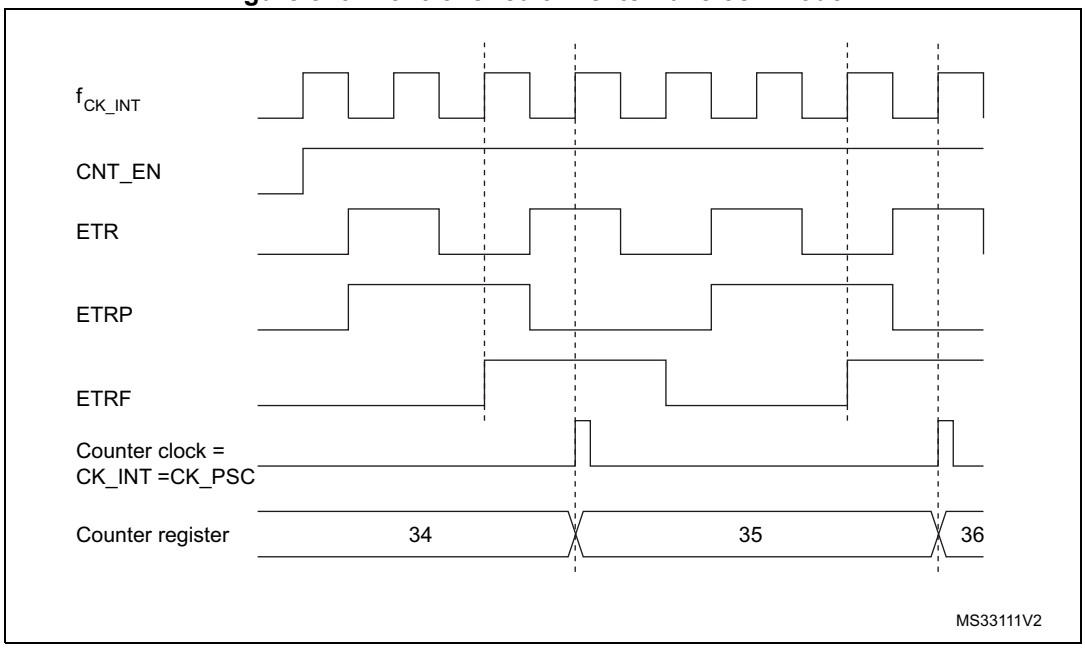
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 370. Control circuit in external clock mode 2



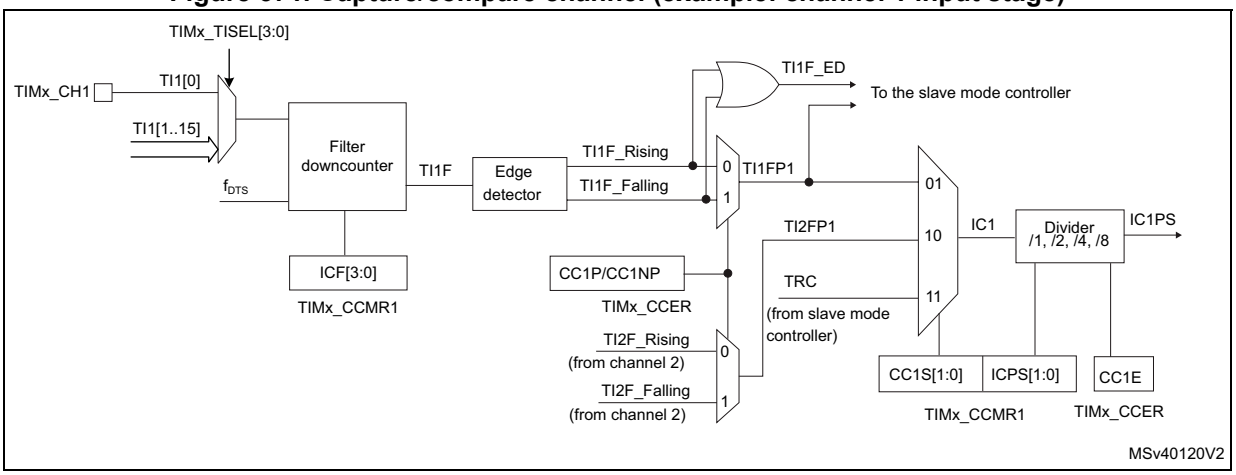
37.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 371 to *Figure 374* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 371. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 372. Capture/compare channel 1 main circuit

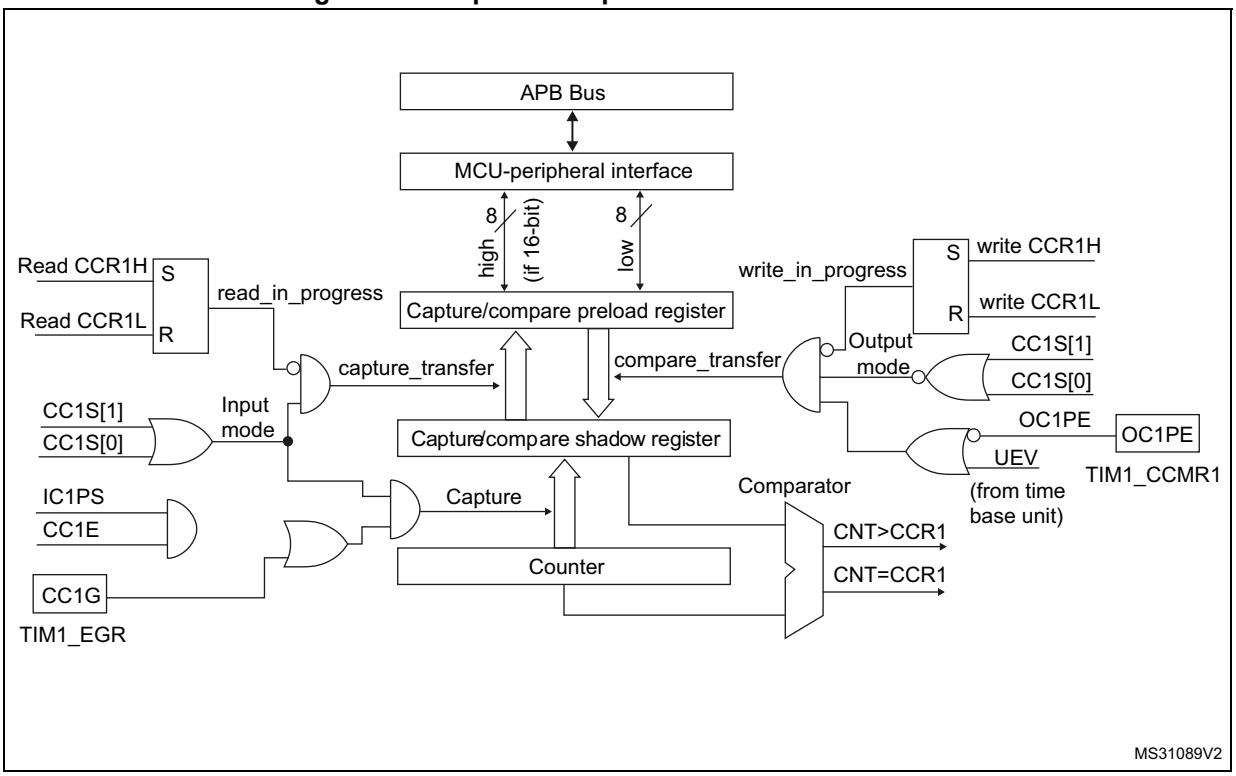
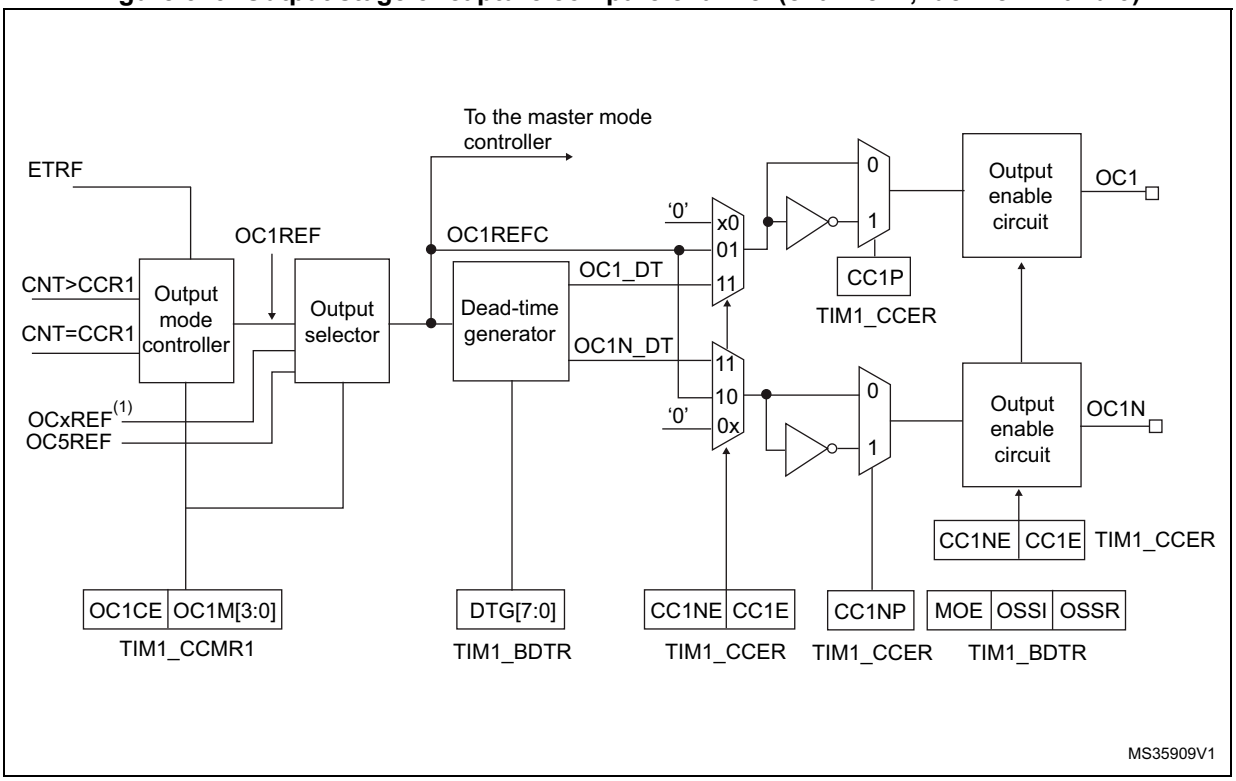


Figure 373. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 374. Output stage of capture/compare channel (channel 4)

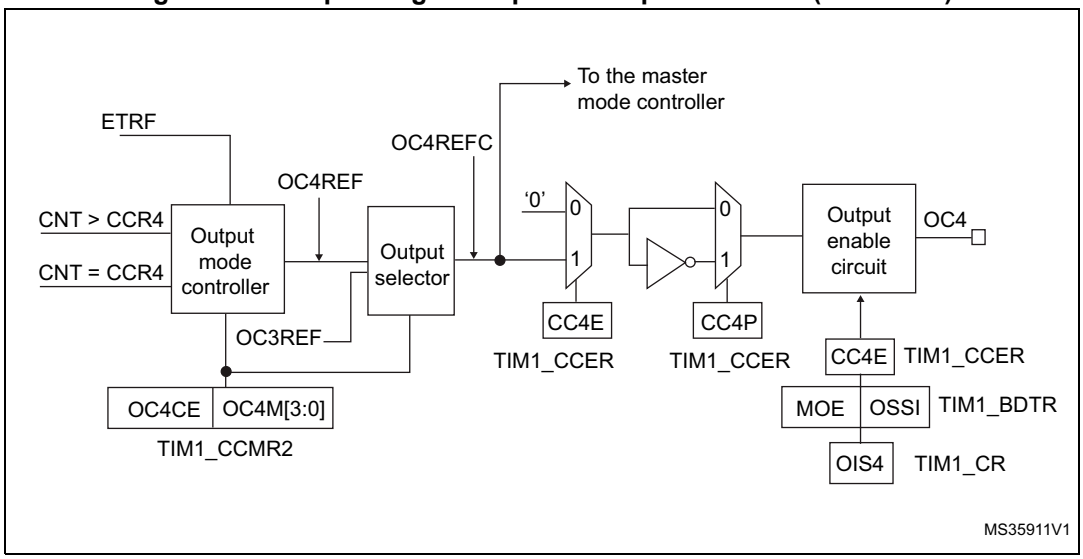
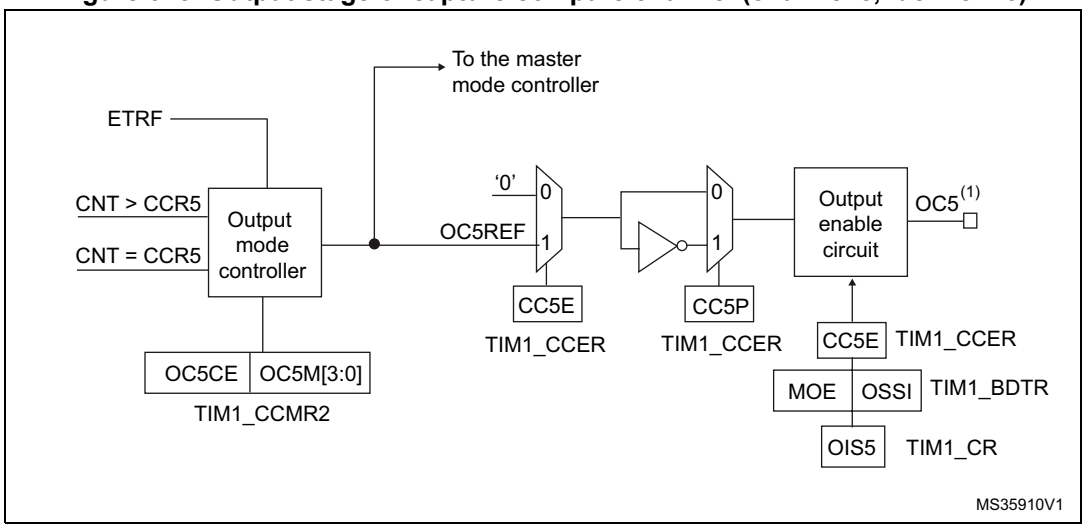


Figure 375. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

37.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or

a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

37.3.8 PWM input mode

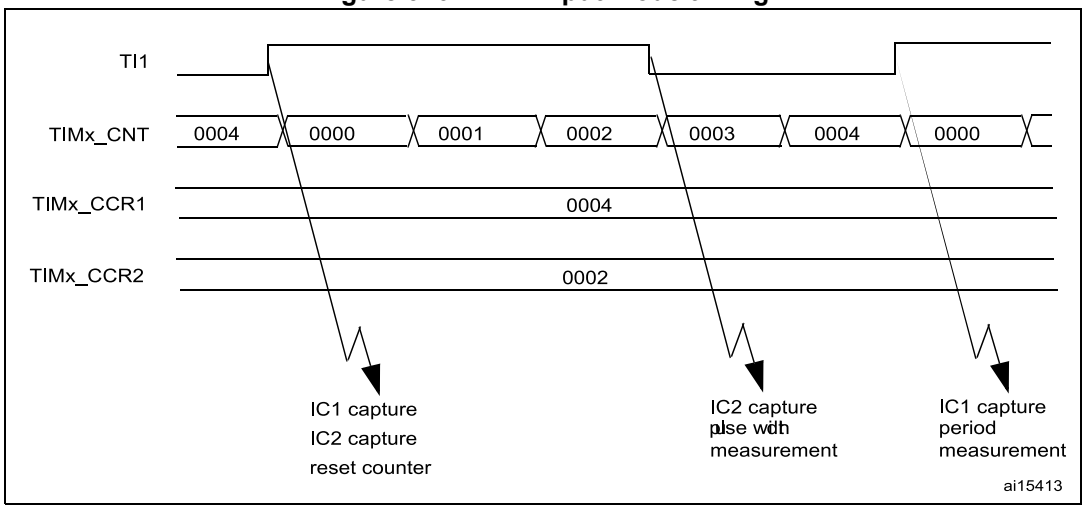
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 376. PWM input mode timing



37.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

37.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

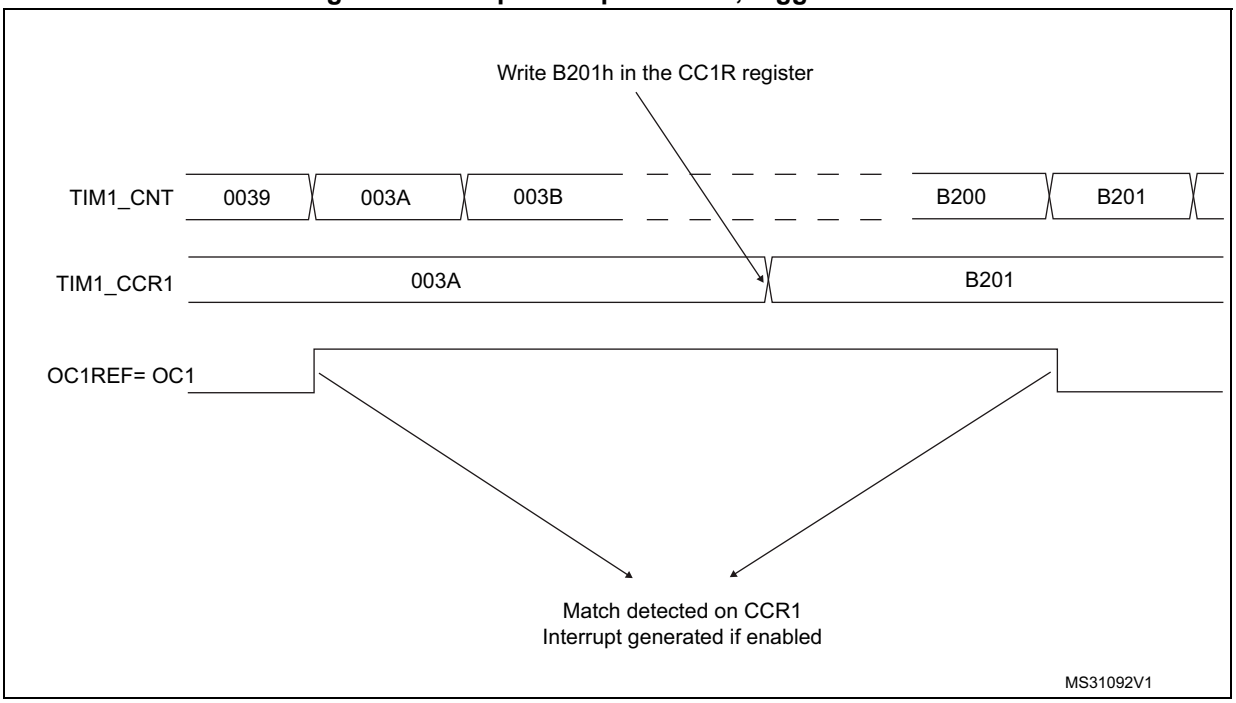
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 377*.

Figure 377. Output compare mode, toggle on OC1



37.3.11 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

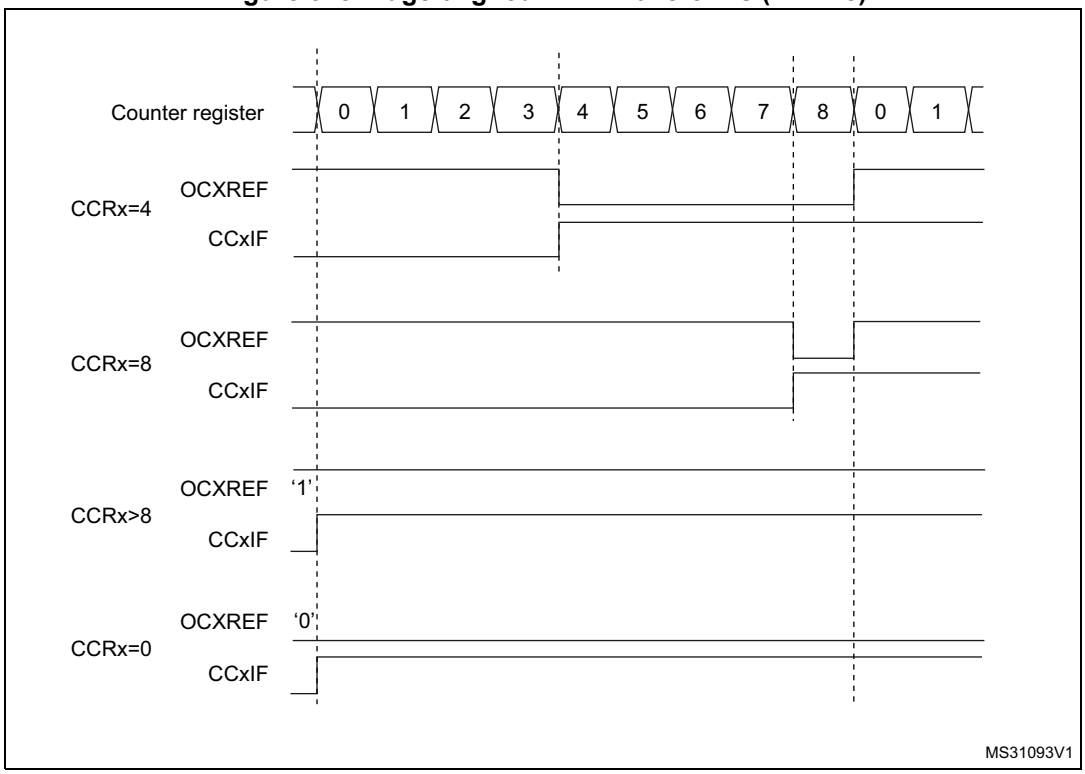
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 1518*.

  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 378* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 378. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 1522*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

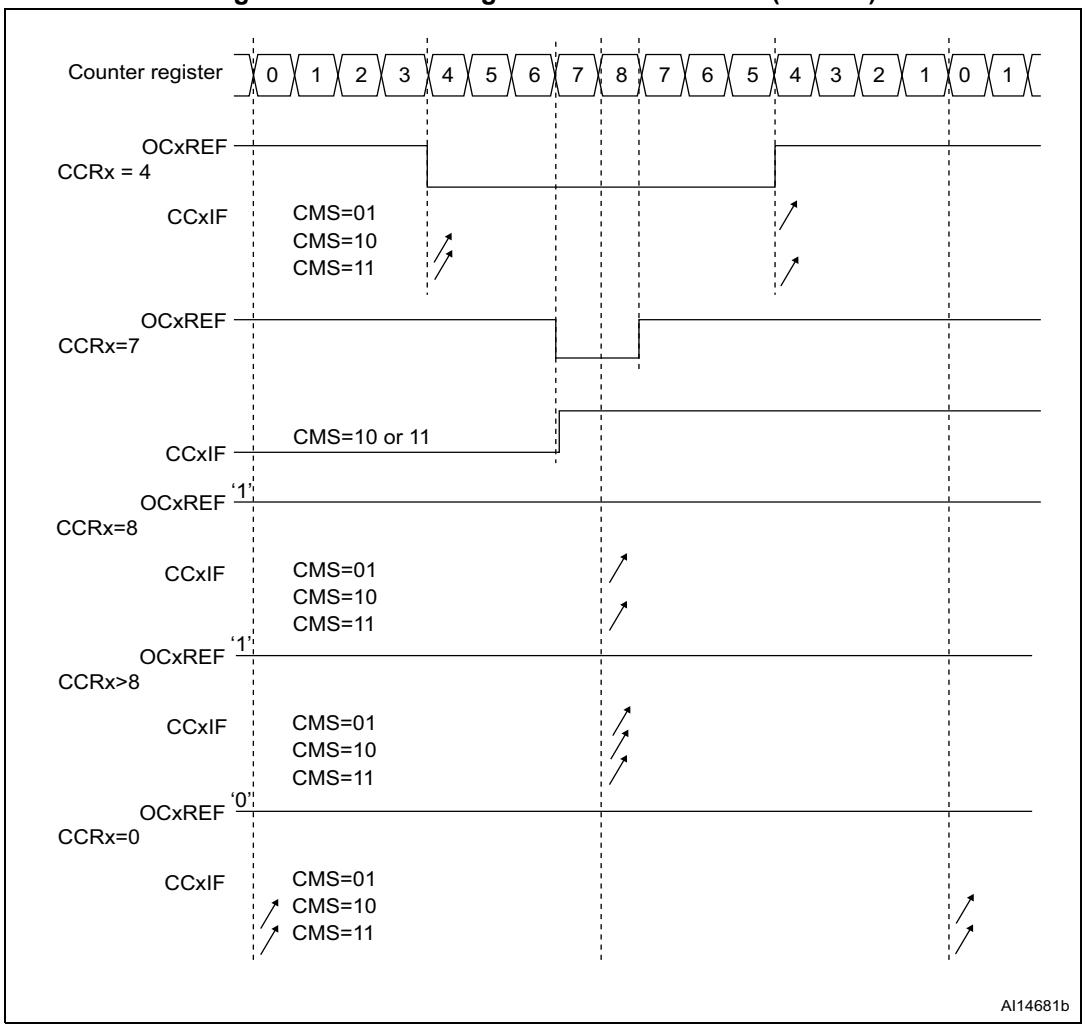
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 1525*.

Figure 379 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 379. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

37.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

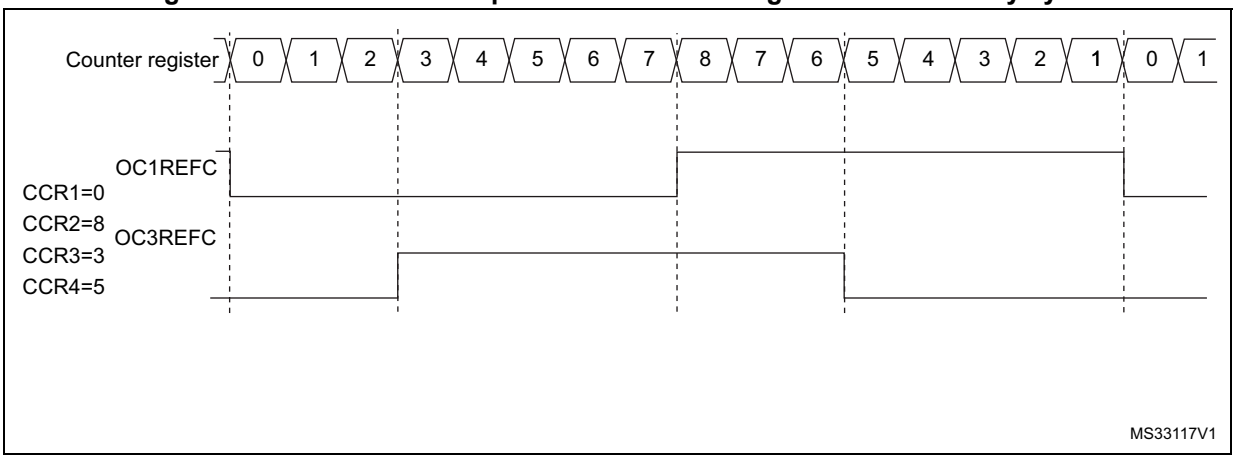
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

Figure 380 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 380. Generation of 2 phase-shifted PWM signals with 50% duty cycle



37.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

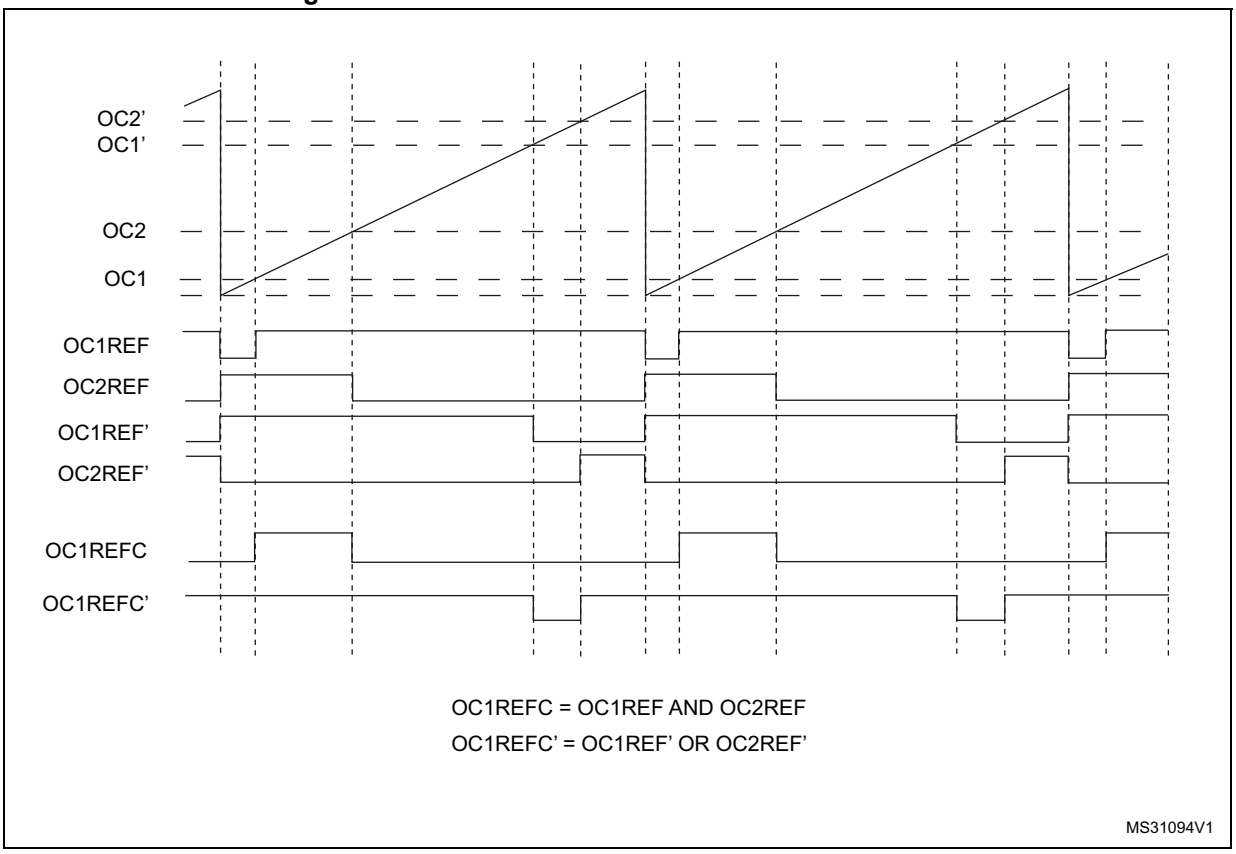
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 381 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

Figure 381. Combined PWM mode on channel 1 and 3



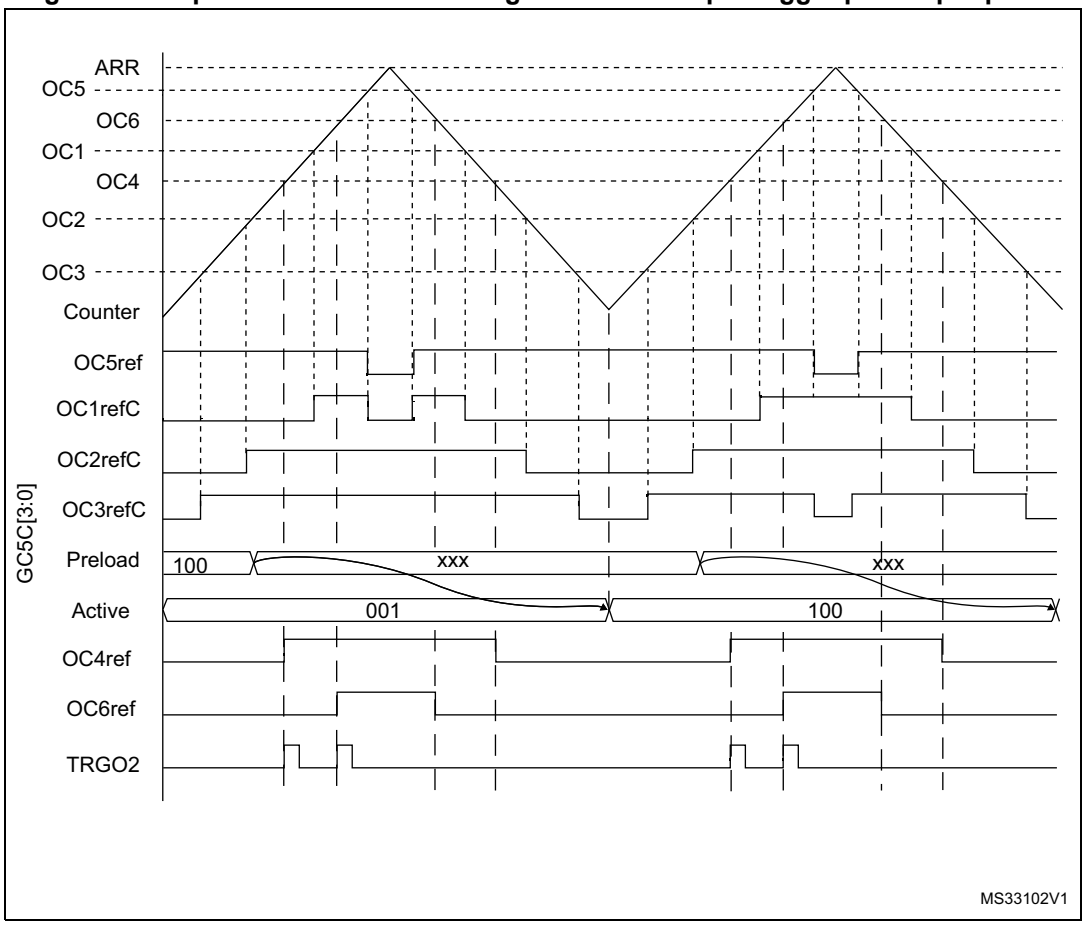
37.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 382. 3-phase combined PWM signals with multiple trigger pulses per period



The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 37.3.27: ADC synchronization* for more details.

37.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 323: Output control bits for complementary OCx and OCxN channels with break feature on page 1593* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 383. Complementary output with dead-time insertion

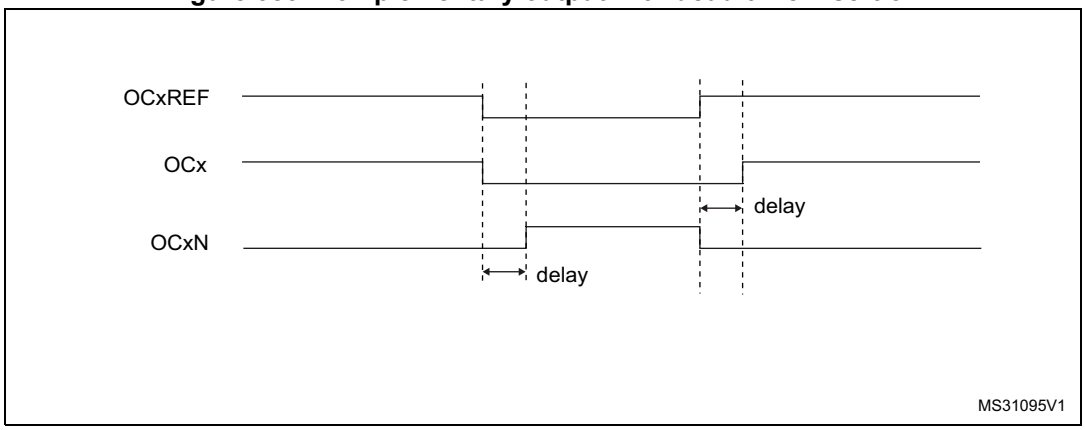


Figure 384. Dead-time waveforms with delay greater than the negative pulse

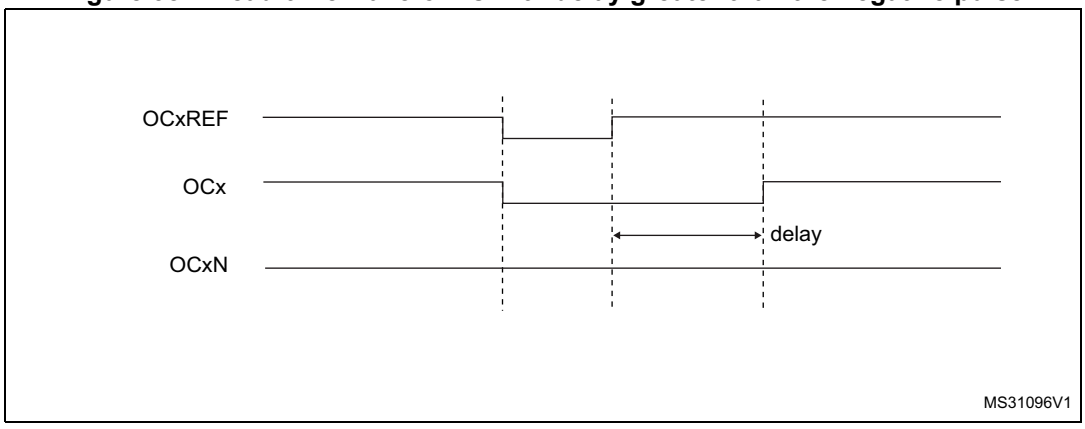
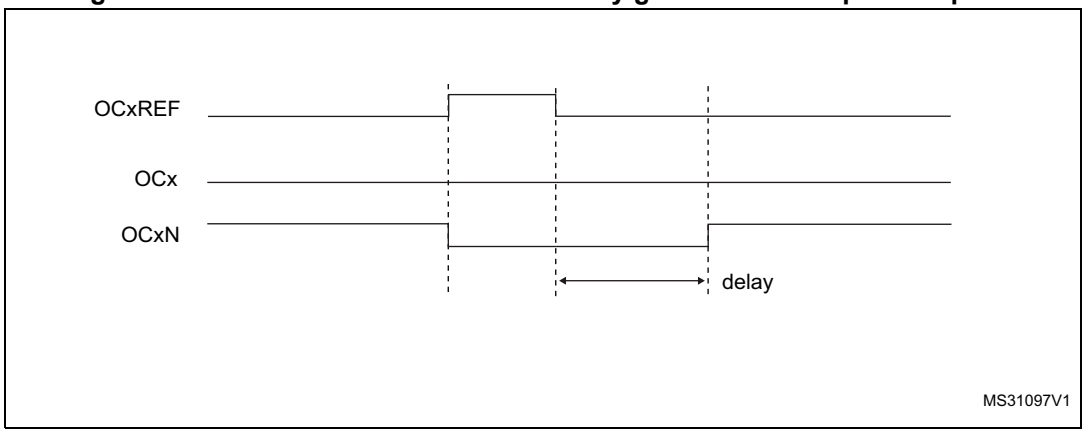


Figure 385. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 37.4.18: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

37.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 and TIM8 timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 323: Output control bits for complementary OCx and OCxN channels with break feature on page 1593* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break functions by setting the BKE and BKE2 bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BKP2 bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral
  - A system break:
    - the Cortex®-M7 LOCKUP output
    - the PVD output
    - the SRAM parity error signal
    - a Flash ECC error
    - a clock failure event generated by the CSS detector

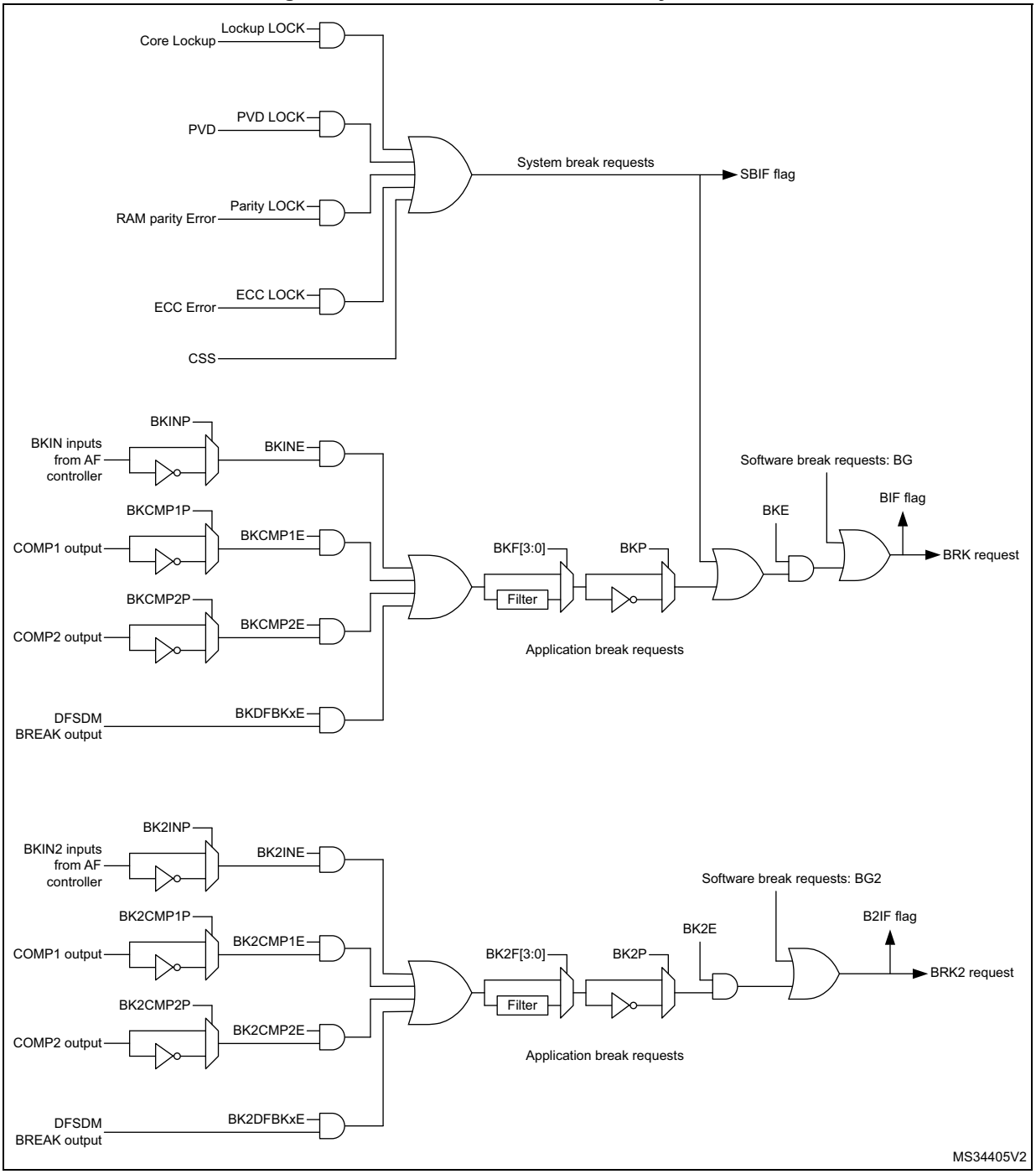
The sources for break2 (BRK2) are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and BG2 is active whatever the BKE and BKE2 enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per *Figure 386* below.

Figure 386. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

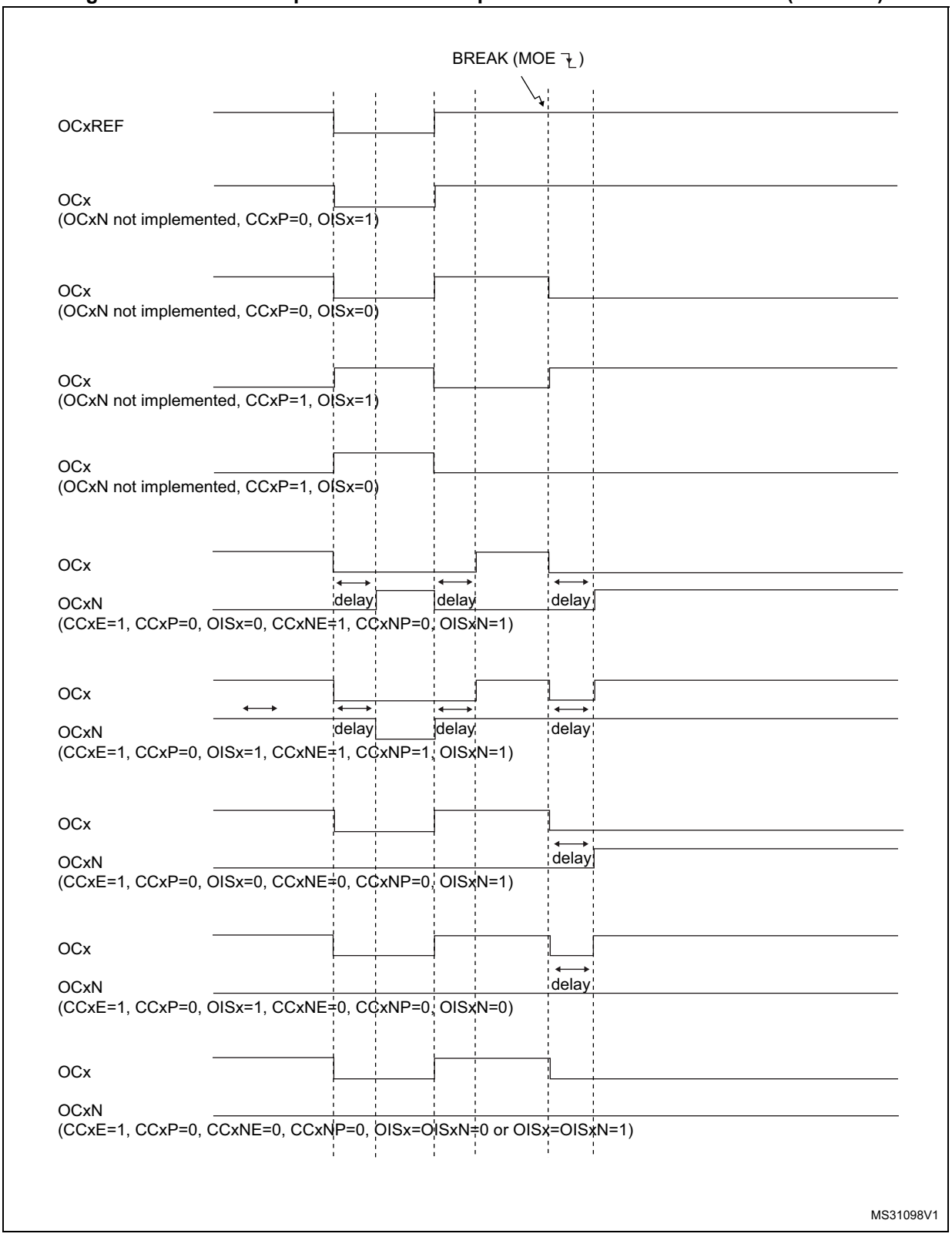
- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to ‘1’ again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 37.4.18: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)*. The LOCK bits can be written only once after an MCU reset.

Figure 387 shows an example of behavior of the outputs in response to a break.

Figure 387. Various output behavior in response to a break event on BRK (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in *Table 320*.

Note: BRK2 must only be used with OSSR = OSSI = 1.

Table 320. Behavior of timer outputs versus BRK/BRK2 inputs

BRK	BRK2	Timer outputs state	Typical use case	
			OCxN output (low side switches)	OCx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 388 gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 388. PWM output state following BRK and BRK2 pins assertion (OSSI=1)

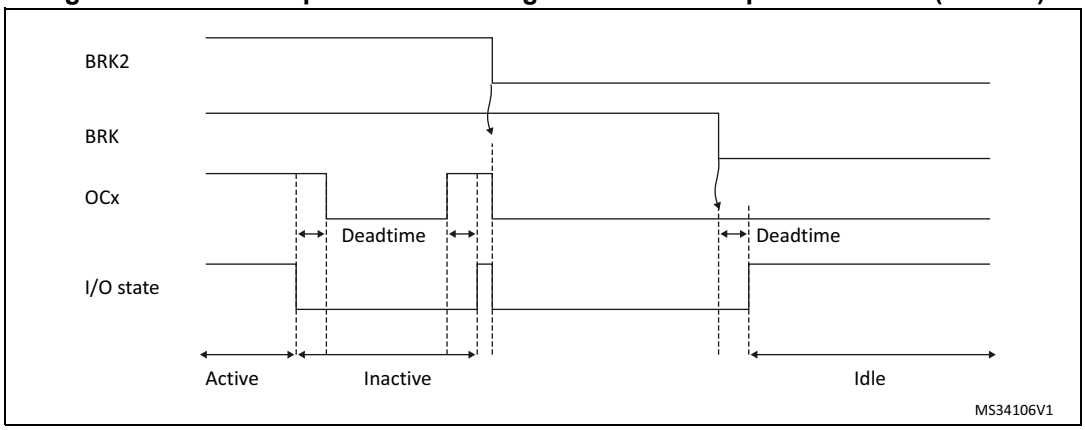
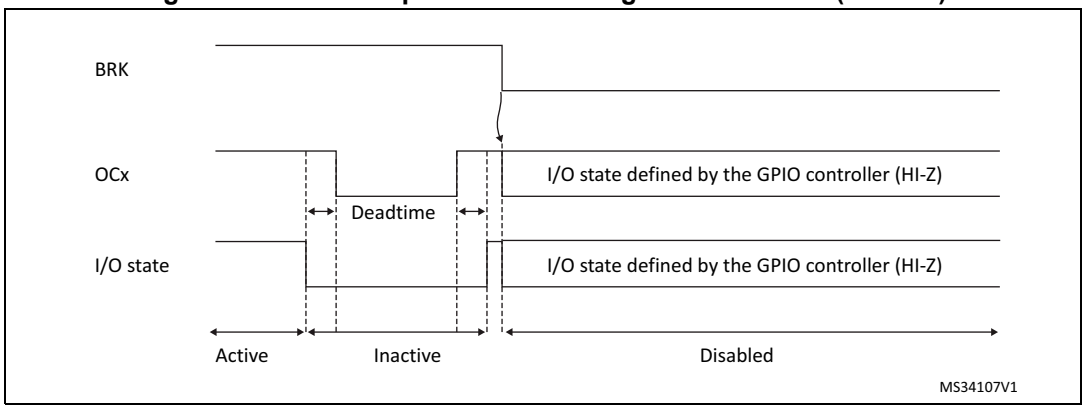


Figure 389. PWM output state following BRK assertion (OSSI=0)



37.3.17 Bidirectional break inputs

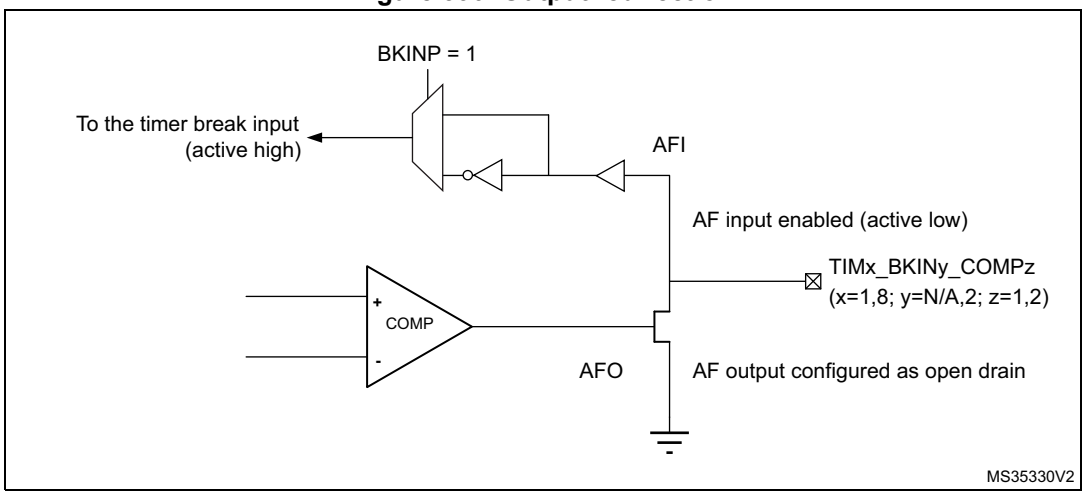
Beside regular digital break inputs and internal break events coming from the comparators, the timer 1 and 8 are featuring bidirectional break inputs/outputs combining the two sources, as represented on *Figure 390*.

The TIMx_BKINy_COMPz pins are combining the COMPz output (to be configured in open drain) and the Timerx's TIMx_BKINy input. They allow to have:

- A global break information available for external MCUs or gate drivers shut down inputs, with a single-pin.

- An internal comparator and multiple external open drain comparators outputs ORed together and triggering a break event, when the multiple internal and external break inputs must be merged.

Figure 390. Output redirection



37.3.18 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be

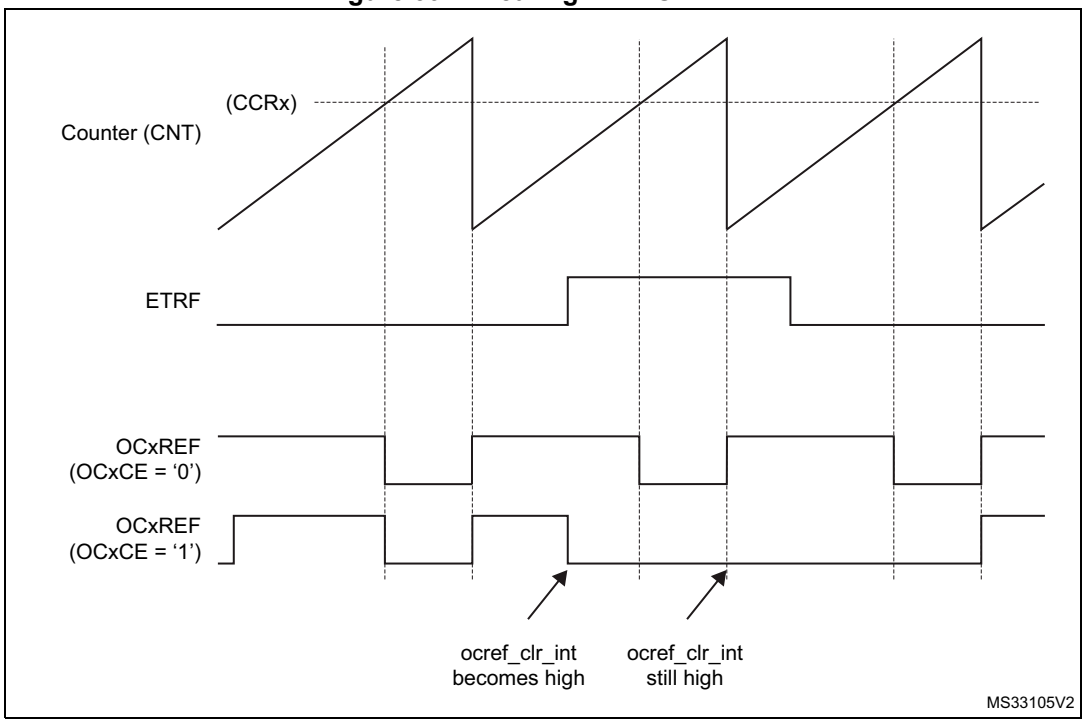
used in Output compare and PWM modes. It does not work in Forced mode. The ocref_clr_int is connected to the ETRF signal (ETR after filtering).

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 391 shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 391. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.

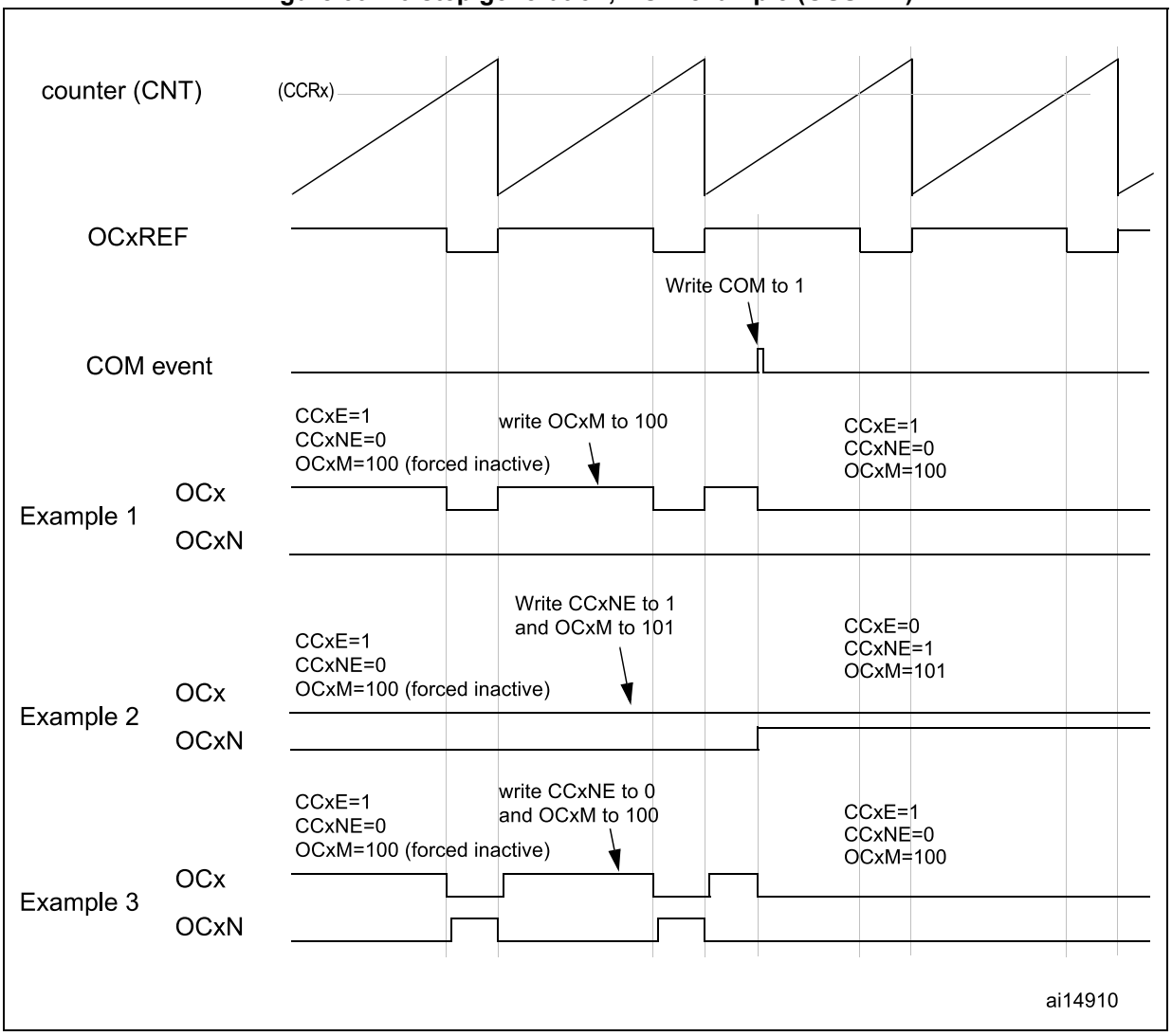
37.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus you can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 392* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 392. 6-step generation, COM example (OSSR=1)



37.3.20 One-pulse mode

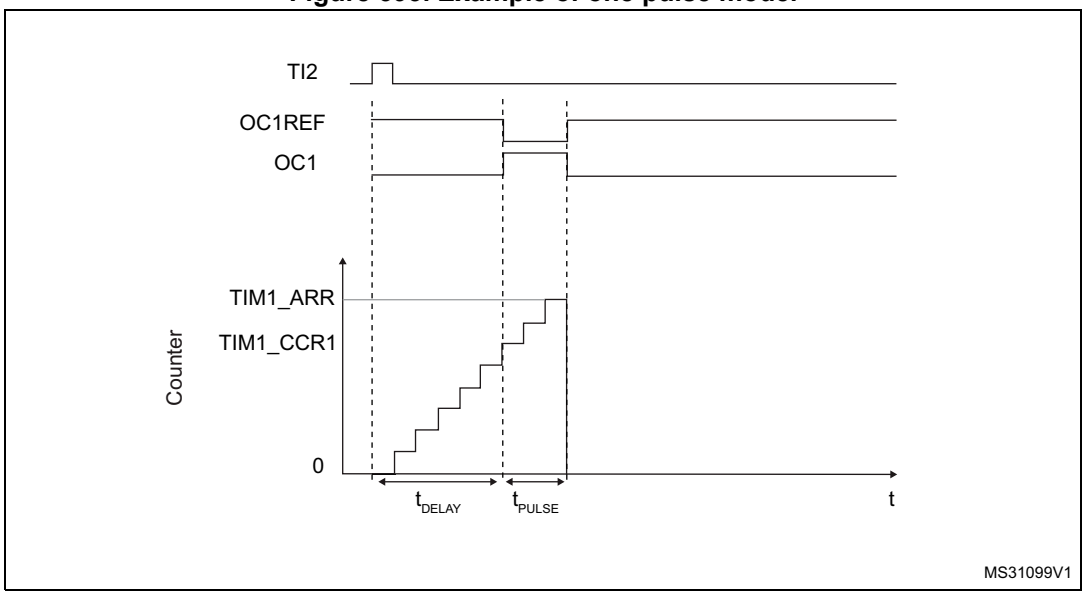
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 393. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

37.3.21 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 37.3.20*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

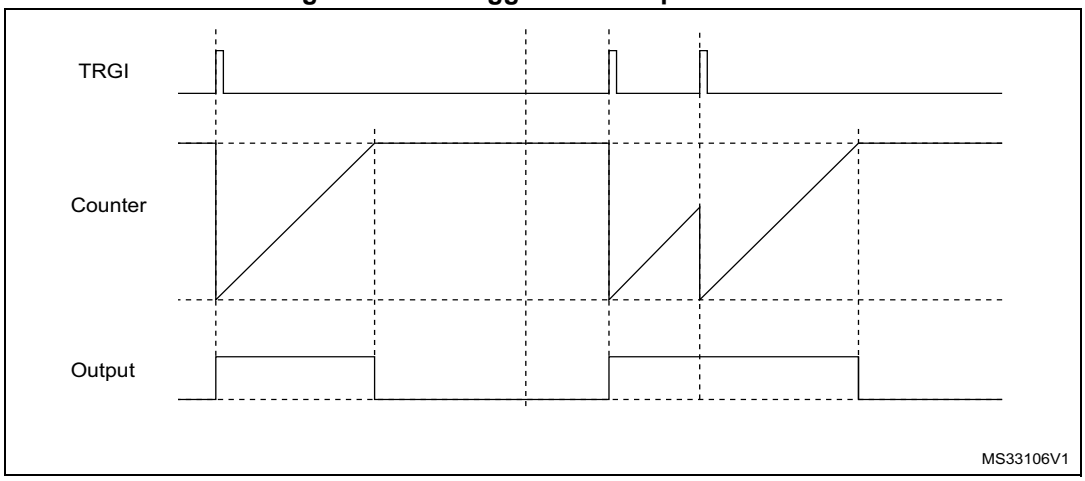
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 394. Retriggerable one pulse mode



37.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an quadrature encoder. Refer to *Table 321*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx_ARR before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don't switch at the same time.

Table 321. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 395* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 395. Example of counter operation in encoder interface mode.

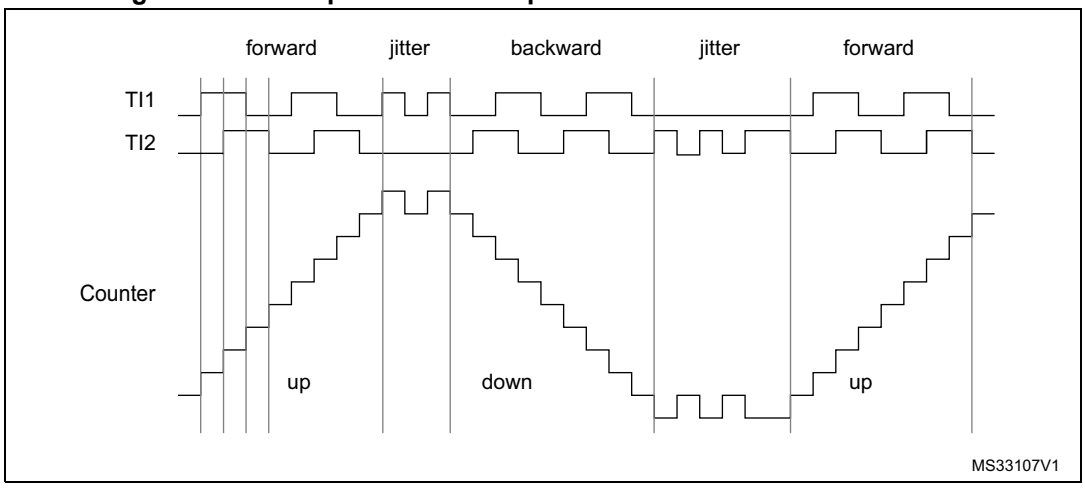
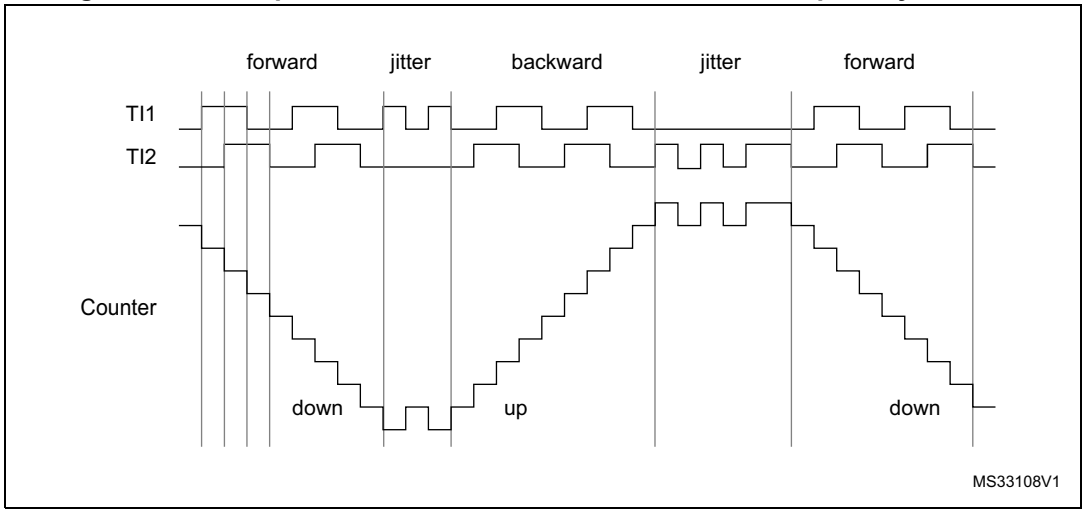


Figure 396 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 396. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

37.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

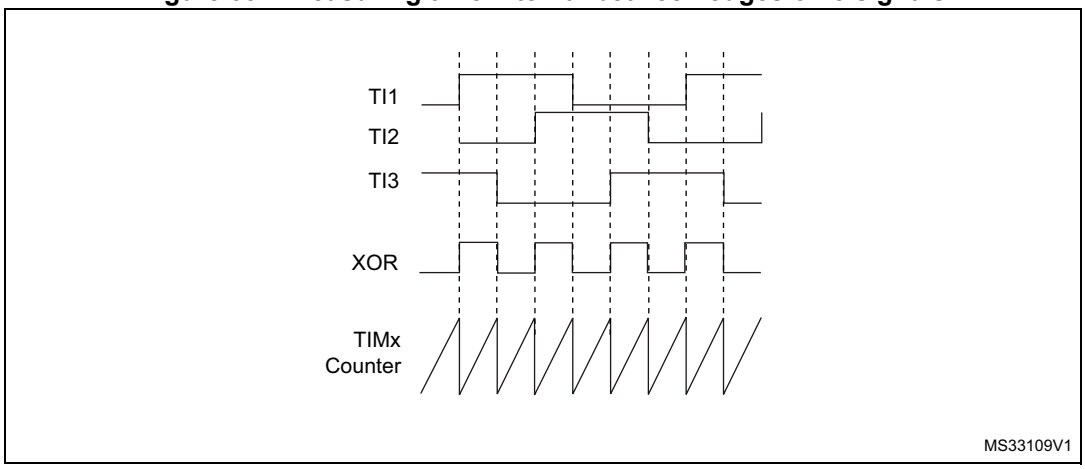
There is no latency between the UIF and UIFCPY flags assertion.

37.3.24 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 397* below.

Figure 397. Measuring time interval between edges on 3 signals



37.3.25 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4) referred to as “interfacing timer” in *Figure 398*. The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See *Figure 371: Capture/compare channel (example: channel 1 input stage) on page 1536*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1 or TIM8) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1 or TIM8) through the TRGO output.

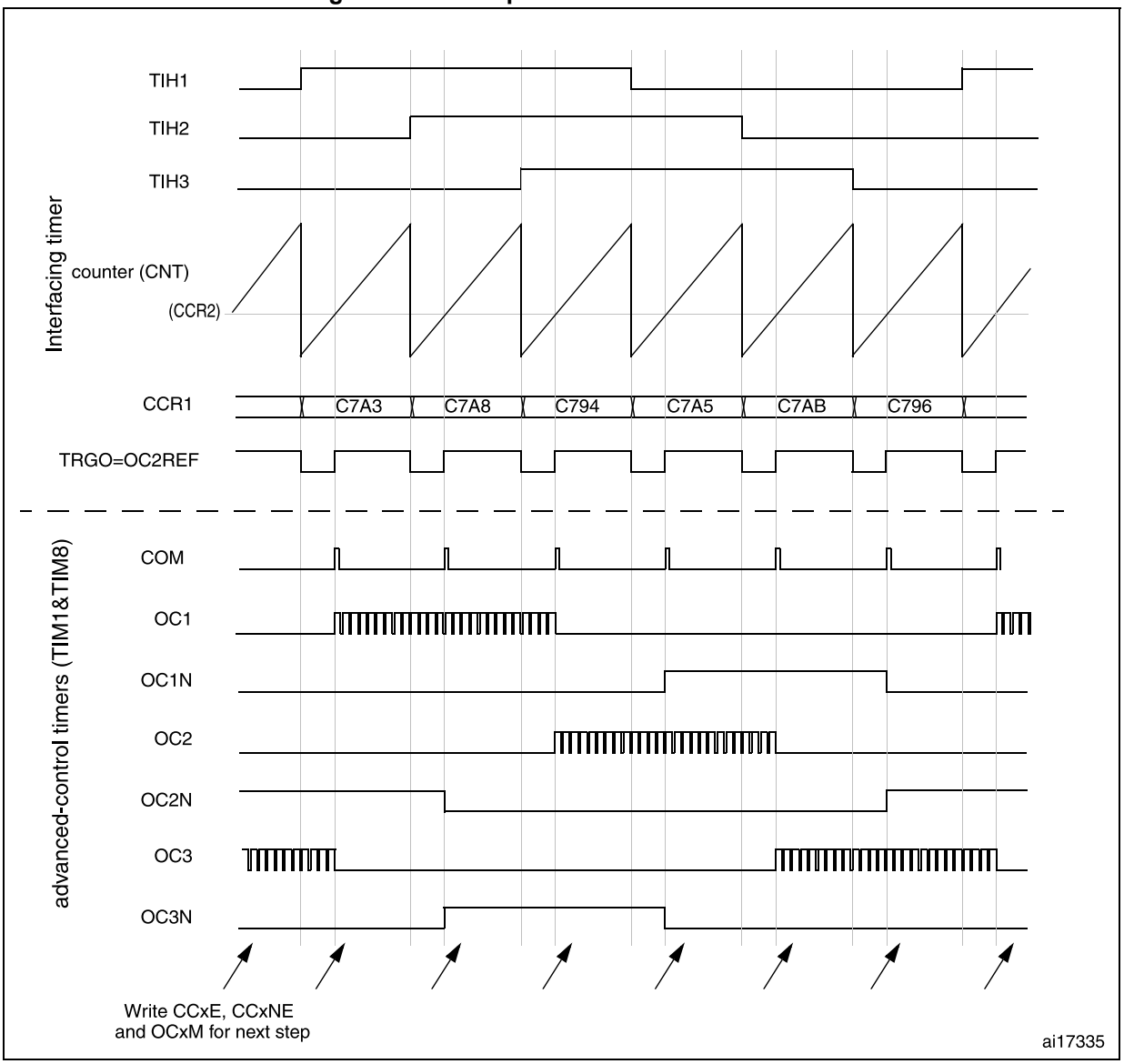
Example: you want to change the PWM configuration of your advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to '1',
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to '01'. You can also program the digital filter if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to '111' and the CC2S bits to '00' in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to '101',

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The *Figure 398* describes this example.

Figure 398. Example of Hall sensor interface



37.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

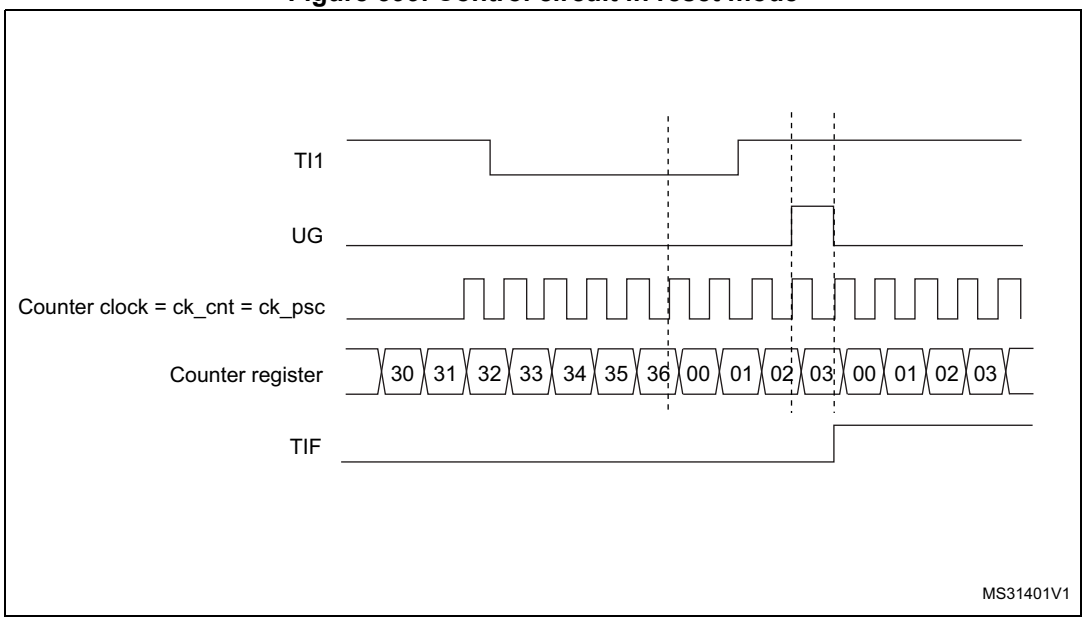
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 399. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

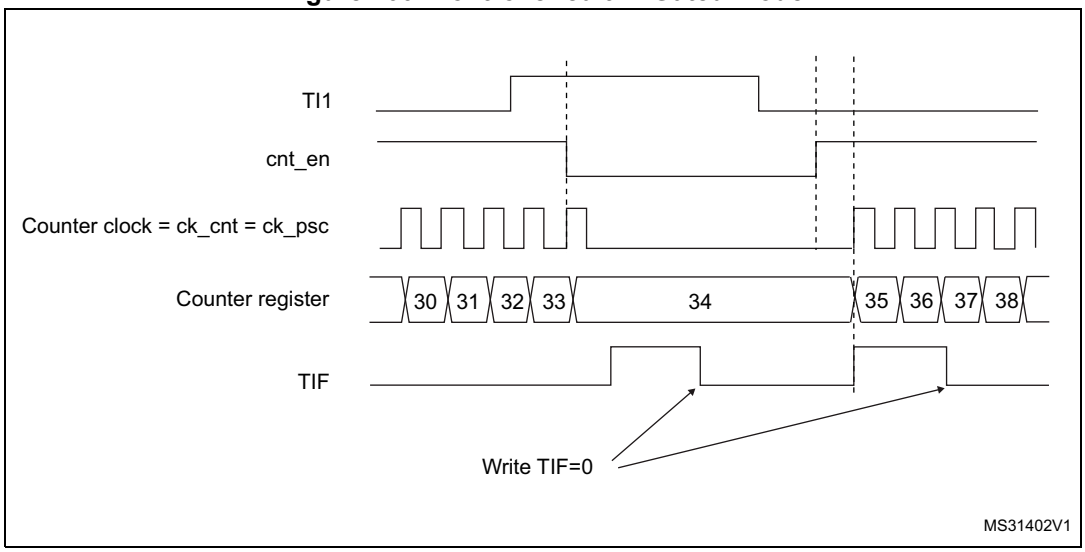
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 400. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register.

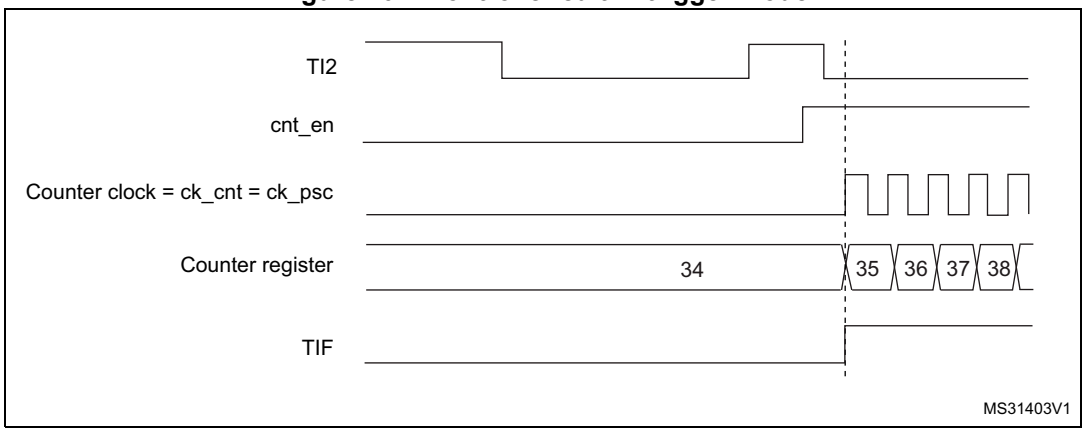
Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 401. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

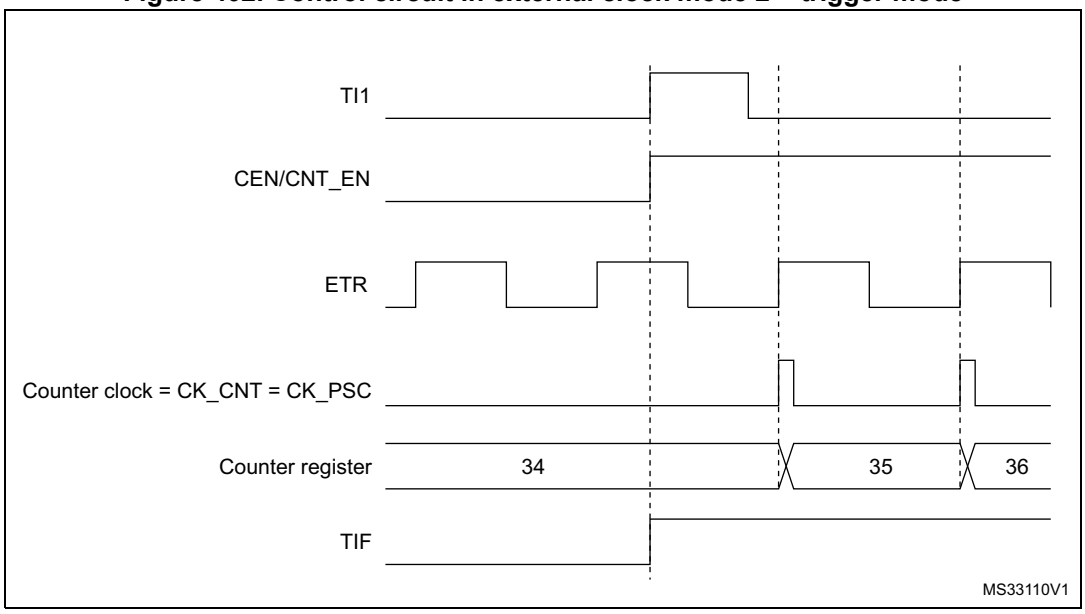
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 402. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

37.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 382 on page 1548*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

37.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

37.3.29 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 with FPU core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to *Section 59.5.8: Microcontroller debug unit (DBGMCU)*.

For safety purposes, when the counter is stopped (TIMx = 1 in DBGMCU_APB2FZ1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

37.4 TIM1/TIM8 registers

Refer to for a list of abbreviations used in register descriptions.

37.4.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx),
00: $t_{DTS}=t_{CK_INT}$
01: $t_{DTS}=2*t_{CK_INT}$
10: $t_{DTS}=4*t_{CK_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

37.4.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
								rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:
0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.
0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).
0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.
0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).
0100: **Compare** - OC1REF signal is used as trigger output (TRGO2)
0101: **Compare** - OC2REF signal is used as trigger output (TRGO2)
0110: **Compare** - OC3REF signal is used as trigger output (TRGO2)
0111: **Compare** - OC4REF signal is used as trigger output (TRGO2)
1000: **Compare** - OC5REF signal is used as trigger output (TRGO2)
1001: **Compare** - OC6REF signal is used as trigger output (TRGO2)
1010: **Compare Pulse** - OC4REF rising or falling edges generate pulses on TRGO2
1011: **Compare Pulse** - OC6REF rising or falling edges generate pulses on TRGO2
1100: **Compare Pulse** - OC4REF or OC6REF rising edges generate pulses on TRGO2
1101: **Compare Pulse** - OC4REF rising or OC6REF falling edges generate pulses on TRGO2
1110: **Compare Pulse** - OC5REF or OC6REF rising edges generate pulses on TRGO2
1111: **Compare Pulse** - OC5REF rising or OC6REF falling edges generate pulses on TRGO2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)
Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)
Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

37.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
*Note: **1:** Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).*
***2:** It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).*
***3:** If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010:
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
00111: External Trigger input (ETRF)
Others: Reserved

See *Table 322: TIMx internal trigger connection on page 1579* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Note: The other bit is at position 16 in the same register

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 322. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM1	TIM15	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

37.4.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

37.4.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System Break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 37.4.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

37.4.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

37.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3
Refer to OC2M description on bits 14:12.

Bits 23:17 Reserved, must be kept at reset value.

Bits16 **OC1M[3]**: Output Compare 1 mode - bit 3
Refer to OC1M description on bits 6:4

Bit 15 **OC2CE:** Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0’) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=’1’).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

37.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x1C

Reset value: 0x0000 0000

Refer to the above CCMR1 register description.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **OC4M[3]**: Output Compare 4 mode - bit 3

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **OC3M[3]**: Output Compare 3 mode - bit 3

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

37.4.9 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/Compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high.
1: OC1N active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high

1: OC1 active low

CC1 channel configured as input: CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

01: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

10: reserved, do not use this configuration.

11: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled.

1: Capture enabled.

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 323. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). **Note**: BRK2 can only be used if OSSI = OSSR = 1.	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

37.4.10 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

37.4.11 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

37.4.12 TIMx auto-reload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 37.3.1: Time-base unit on page 1516* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

37.4.13 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

the number of PWM periods in edge-aligned mode

the number of half PWM period in center-aligned mode.

37.4.14 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

37.4.15 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

37.4.16 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

37.4.17 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

37.4.18 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **BK2P**: Break 2 polarity

0: Break input BRK2 is active low

1: Break input BRK2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 386: Break and Break2 circuitry overview*).

0: Break2 function disabled

1: Break2 function enabled

Note: The BRKIN2 must only be used with OSSR = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK2 acts asynchronously

0001: $f_{SAMPLING}=f_{CK_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously

0001: $f_{SAMPLING}=f_{CK_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled

In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details (*Section 37.4.9: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 386: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 37.4.9: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 37.4.9: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).
0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).
1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected.
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

37.4.19 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
10001: 18 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

– If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data will be copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

– If you configure the DMA Data Size in half-words, 16-bit data will be transferred to each of the 7 registers.

– If you configure the DMA Data Size in bytes, the data will also be transferred to 7 registers: the first register will contain the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, you also have to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

37.4.20 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4
where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

37.4.21 TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8)

Address offset: 0x54

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

37.4.22 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3

Distortion on Channel 3 output:

0: No effect of OC5REF on OC3REFC

1: OC3REFC is the logical AND of OC3REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group Channel 5 and Channel 2

Distortion on Channel 2 output:

0: No effect of OC5REF on OC2REFC

1: OC2REFC is the logical AND of OC2REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group Channel 5 and Channel 1

Distortion on Channel 1 output:

0: No effect of OC5REF on OC1REFC5

1: OC1REFC is the logical AND of OC1REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.

37.4.23 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC6 output.

37.4.24 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	BK CMP2P	BK CMP1P	BKINP	BKDF1 BK0E	Res.	Res.	Res.	Res.	Res.	BK CMP2E	BK CMP1E	BKINE
rw	rw			rw	rw	rw	rw						rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: COMP1 output
0010: COMP2 output
0011: ADC1 AWD1
0100: ADC1 AWD2
0101: ADC1 AWD3
0110: ADC3 AWD1
0111: ADC3 AWD2
1000: ADC3 AWD3
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active high
1: COMP2 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active high
1: COMP1 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active high
1: BKIN input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK0E**: BRK dfsdm1_break[0] enable
This bit enables the dfsdm1_break[0] for the timer's BRK input. dfsdm1_break[0] output is 'ORed' with the other BRK sources.
0: dfsdm1_break[0] input disabled
1: dfsdm1_break[0] input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable
This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.
0: COMP2 input disabled
1: COMP2 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable
This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.
0: COMP1 input disabled
1: COMP1 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 365: TIM1/TIM8 ETR input circuitry and to Figure 386: Break and Break2 circuitry overview.

37.4.25 TIM1 Alternate function register 2 (TIM1_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BK2 CMP2 P	BK2 CMP1 P	BK2 INP	BK2DF1 BK1E	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP2 polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP2 polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BKP2 polarity bit.

0: BKIN2 input is active low

1: BKIN2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2DF1BK1E**: BRK2 dfsdm1_break[1] enable

This bit enables the dfsdm1_break[1] for the timer's BRK2 input. dfsdm1_break[1] output is 'ORed' with the other BRK2 sources.

0: dfsdm1_break[1] input disabled

1: dfsdm1_break[1] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable

This bit enables the COMP2 for the timer's BRK2 input. COMP2 output is 'ORed' with the other BRK2 sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable

This bit enables the COMP1 for the timer's BRK2 input. COMP1 output is 'ORed' with the other BRK2 sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 386: Break and Break2 circuitry overview.

37.4.26 TIM8 Alternate function option register 1 (TIM8_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	BK CMP2 P	BK CMP1 P	BKINP	BKDF1 BK2E	Res.	Res.	Res.	Res.	Res.	BK CMP2E	BK CMP1E	BKINE
rw	rw			rw	rw	rw	rw						rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: COMP1 output
0010: COMP2 output
0011: ADC2 AWD1
0100: ADC2 AWD2
0101: ADC2 AWD3
0110: ADC3 AWD1
0111: ADC3 AWD2
1000: ADC3 AWD3
Others: Reserved
Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity
This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP2 input is active high
1: COMP2 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: COMP1 input is active high
1: COMP1 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active high
1: BKIN input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable
This bit enables the dfsdm1_break[2] for the timer's BRK input. dfsdm1_break[2] output is 'ORed' with the other BRK sources.
0: dfsdm1_break[2] input disabled
1: dfsdm1_break[2] input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 365: TIM1/TIM8 ETR input circuitry and to Figure 386: Break and Break2 circuitry overview.

37.4.27 TIM8 Alternate function option register 2 (TIM8_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BK2 CMP2 P	BK2 CMP1 P	BK2 INP	BK2DF1 BK3E	Res.	Res.	Res.	Res.	Res.	BK2 CMP2E	BK2 CMP1E	BK2INE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP2 polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity
This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP2 polarity bit.
0: COMP1 input is active low
1: COMP1 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity
This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BKP2 polarity bit.
0: BKIN2 input is active low
1: BKIN2 input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2DF1BK3E**: BRK2 dfsdm1_break[3] enable
This bit enables the dfsdm1_break[3] for the timer's BRK2 input. dfsdm1_break[3] output is 'ORed' with the other BRK2 sources.
0: dfsdm1_break[3] input disabled
1: dfsdm1_break[3] input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **BK2CMP2E**: BRK2 COMP2 enable
This bit enables the COMP2 for the timer's BRK2 input. COMP2 output is 'ORed' with the other BRK2 sources.
0: COMP2 input disabled
1: COMP2 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: BRK2 COMP1 enable
This bit enables the COMP1 for the timer's BRK2 input. COMP1 output is 'ORed' with the other BRK2 sources.
0: COMP1 input disabled
1: COMP1 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: BRK2 BKIN input enable
This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.
0: BKIN2 input disabled
1: BKIN2 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 386: Break and Break2 circuitry overview.

37.4.28 TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw						rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input
0000: TIM1_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input
0000: TIM1_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
0000: TIM1_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM1_CH1 input
0001: COMP1 output
Others: Reserved

37.4.29 TIM8 timer input selection register (TIM8_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw						rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input
0000: TIM8_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input
0000: TIM8_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
0000: TIM8_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM8_CH1 input
0001: COMP2 output
Others: Reserved

37.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 324. TIM1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM1_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM1_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIM1_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIM1_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**TIM1_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**TIM1_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM1_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM1_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM1_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	Res.	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 324. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIM1_CNT**	UIFCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM1_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM1_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM1_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**TIM1_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM1_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM1_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM1_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM1_BDTR**	Res.	Res.	Res.	Res.	Res.	Res.	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM1_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM1_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	**TIM1_CCMR3** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x58	**TIM1_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 324. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C	**TIM1_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM1_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]						BKCMP2P	BKCMP1P	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0			0	0	0	0						0	0	1
0x64	**TIM1_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	BK2DF1BK1E	Res.	Res.	Res.	Res.	Res.	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																					0	0	0	0						0	0	1
0x68	**TIM1_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0						0	0	0	0

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

37.4.31 TIM8 register map

TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

Table 325. TIM8 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM8_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM8_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIM8_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIM8_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**TIM8_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**TIM8_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM8_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM8_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM8_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM8_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM8_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	Res.	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**TIM8_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 325. TIM8 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**TIM8_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM8_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM8_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM8_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM8_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM8_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM8_BDTR**	Res.	Res.	Res.	Res.	Res.	Res.	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM8_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM8_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	**TIM8_CCMR3** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x58	**TIM8_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 325. TIM8 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C	**TIM8_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM8_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0			0	0	0	0						0	0	1
0x64	**TIM8_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP2P	BK2CMP1P	BK2INP	BK2DF1BK3E	Res.	Res.	Res.	Res.	Res.	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																					0	0	0	0						0	0	1
0x68	**TIM8_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

38 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

38.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

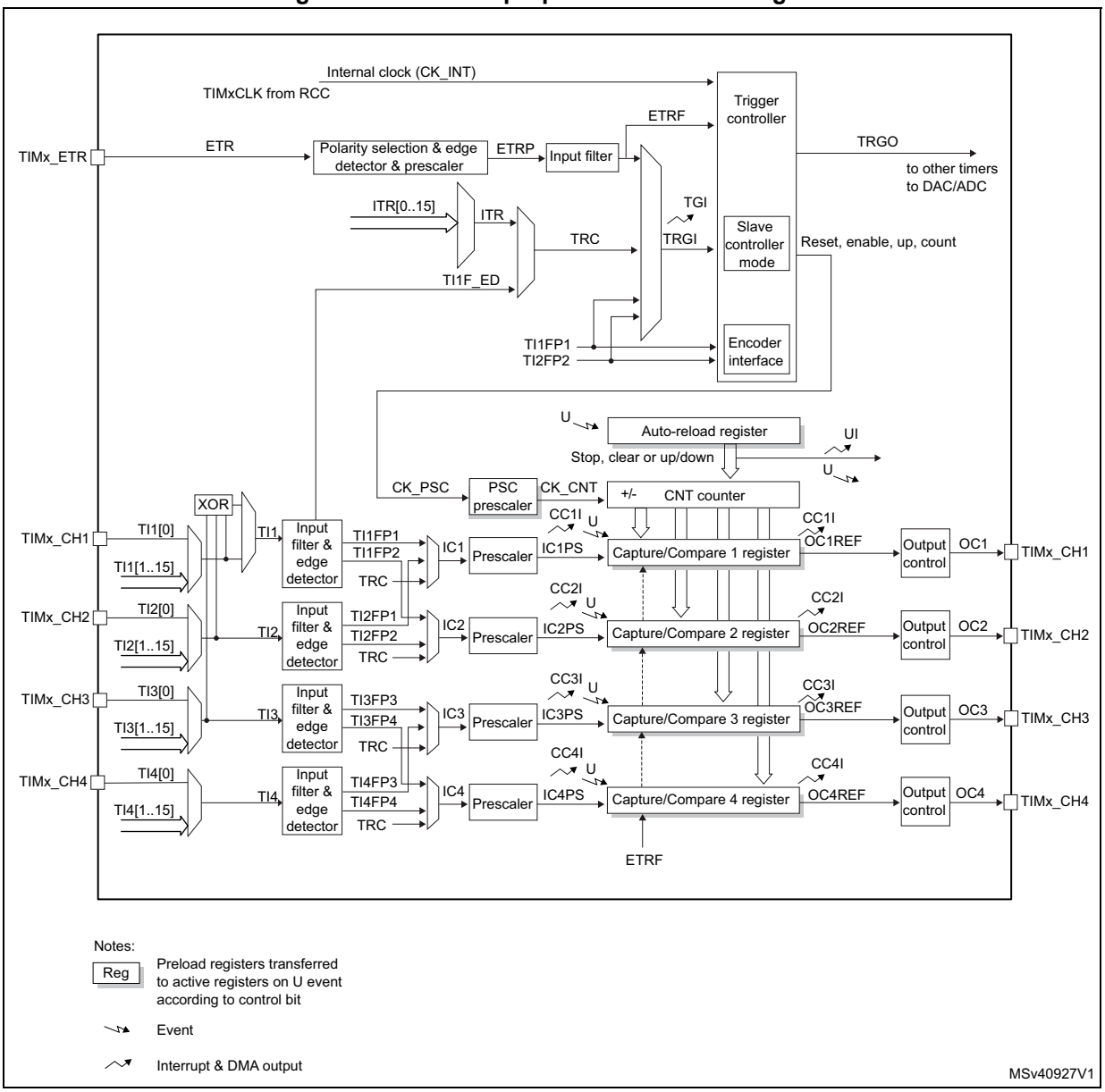
The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 38.3.19: Timer synchronization*.

38.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 403. General-purpose timer block diagram



38.3 TIM2/TIM3/TIM4/TIM5 functional description

38.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 404 and *Figure 405* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 404. Counter timing diagram with prescaler division change from 1 to 2

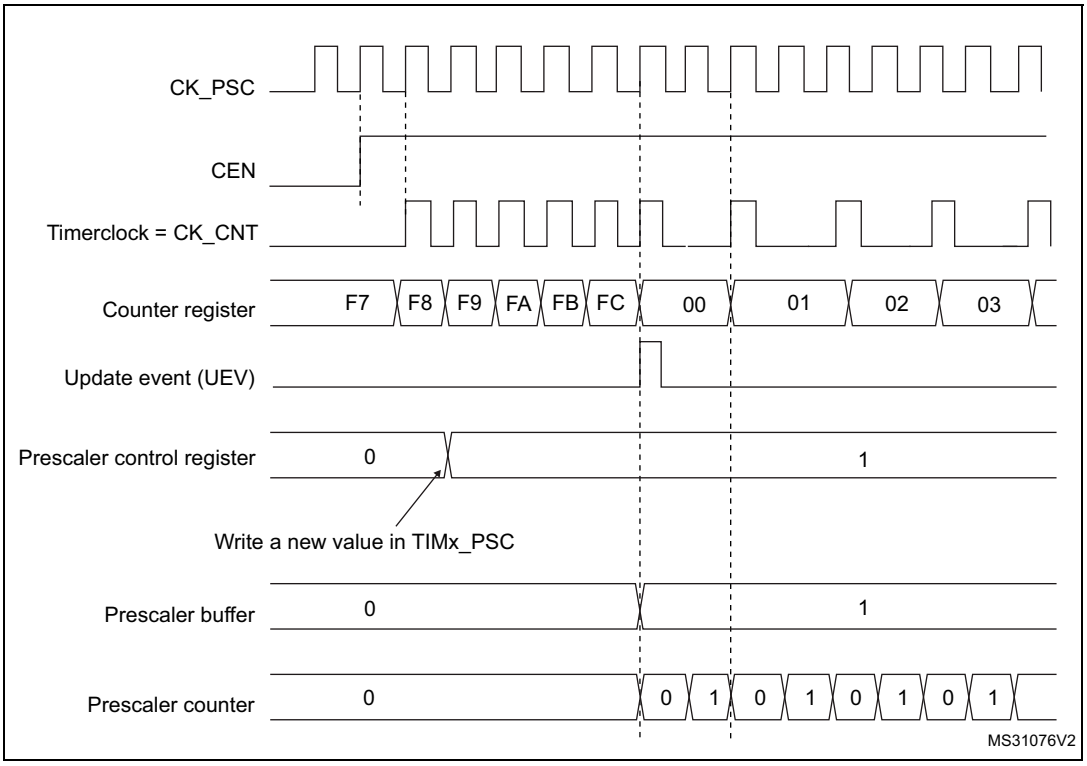
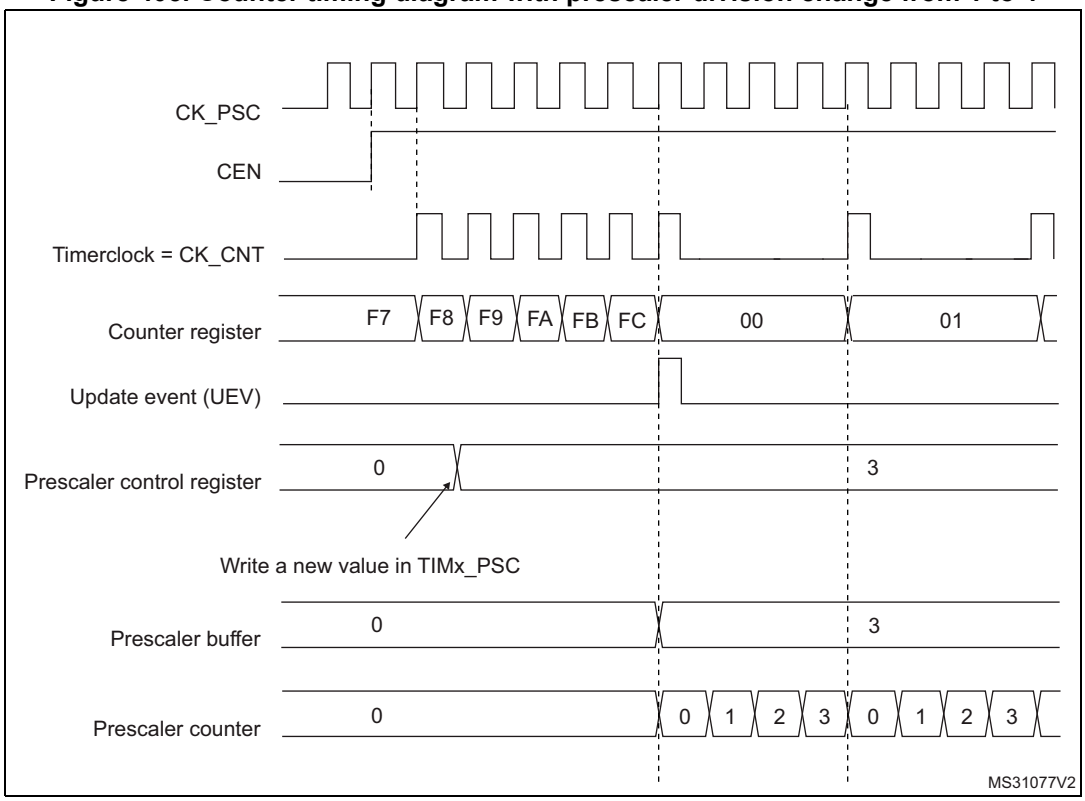


Figure 405. Counter timing diagram with prescaler division change from 1 to 4



38.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 406. Counter timing diagram, internal clock divided by 1

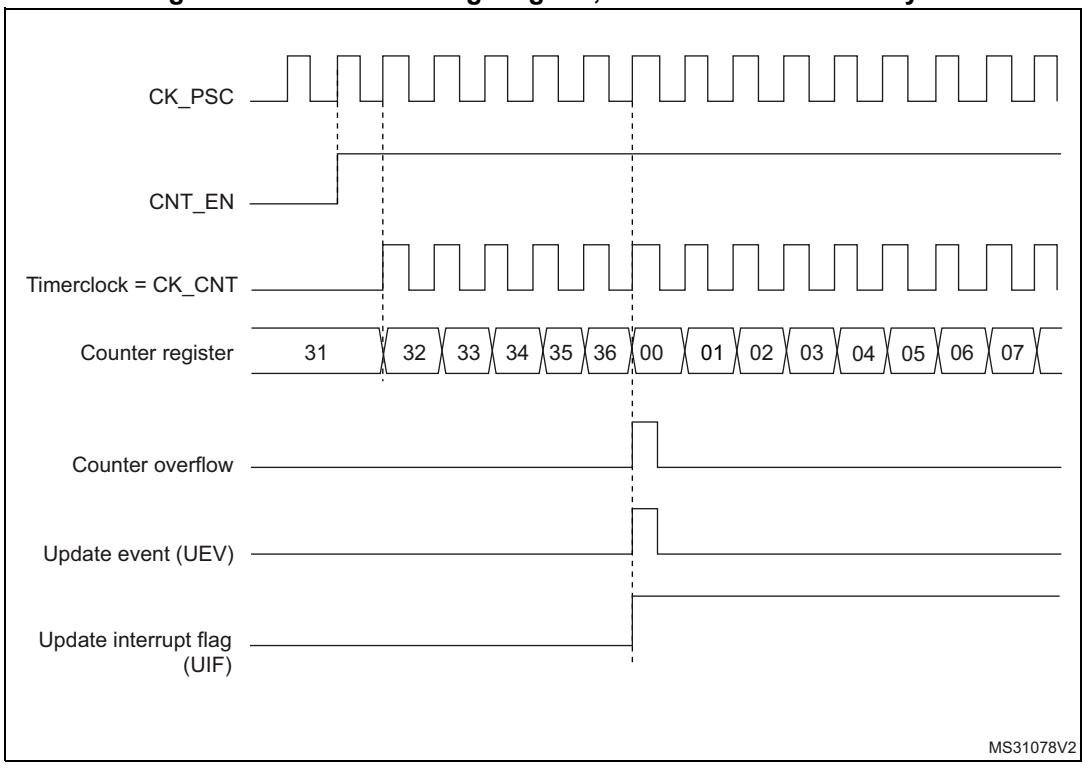


Figure 407. Counter timing diagram, internal clock divided by 2

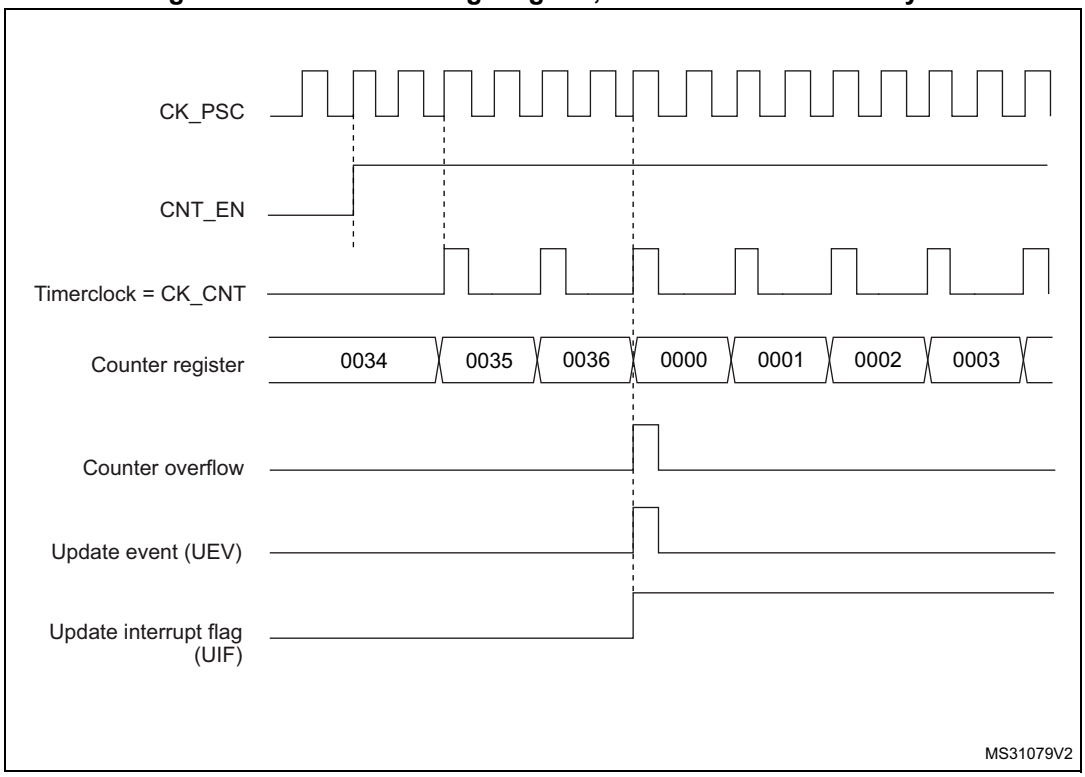


Figure 408. Counter timing diagram, internal clock divided by 4

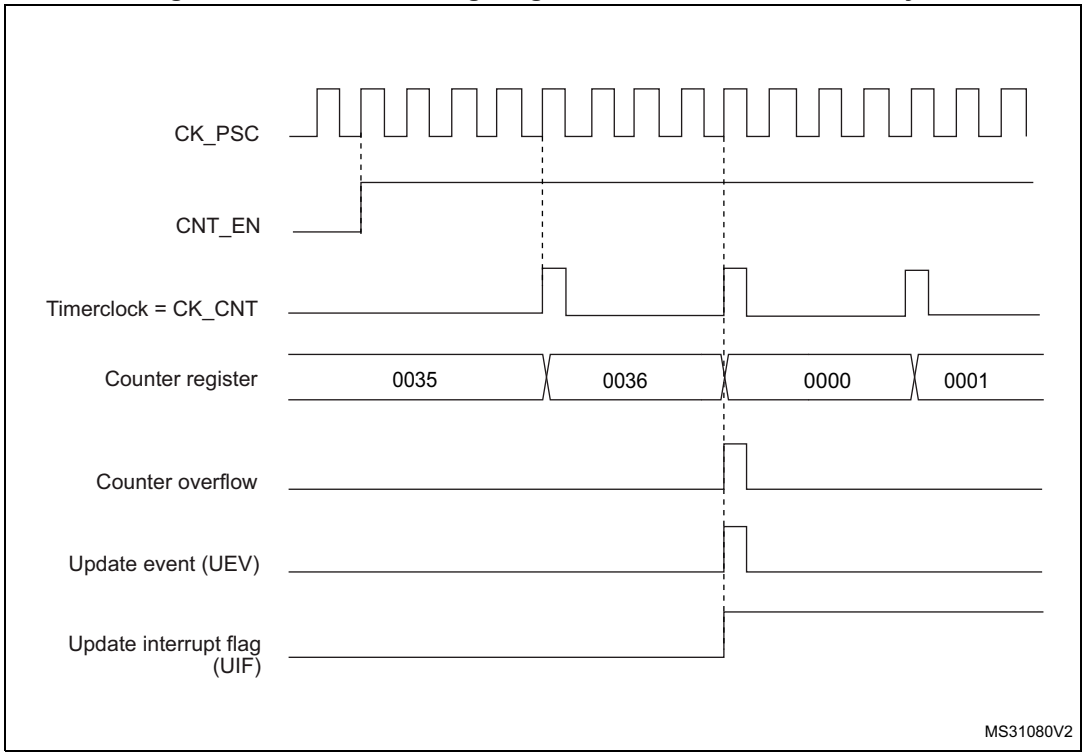


Figure 409. Counter timing diagram, internal clock divided by N

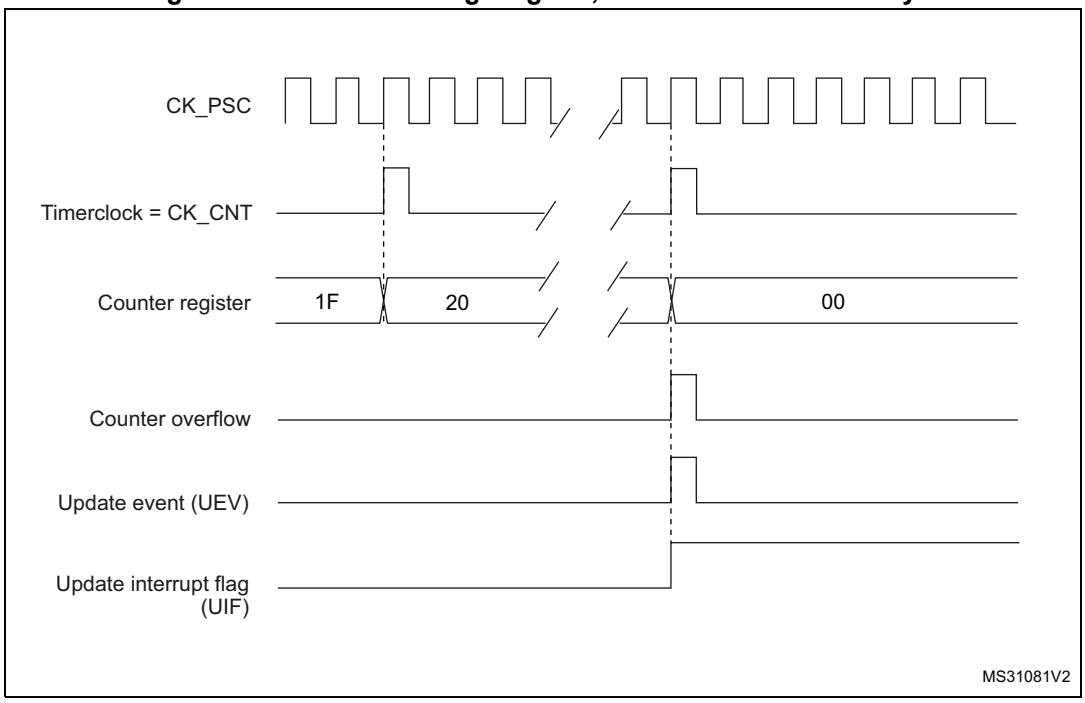


Figure 410. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

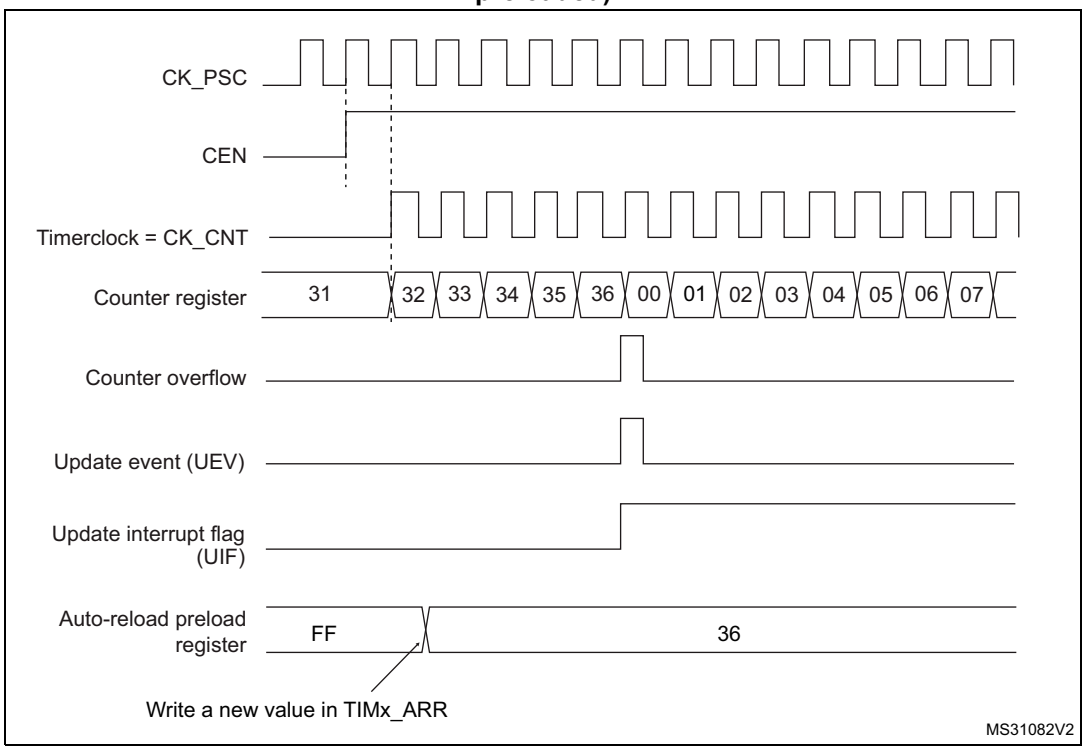
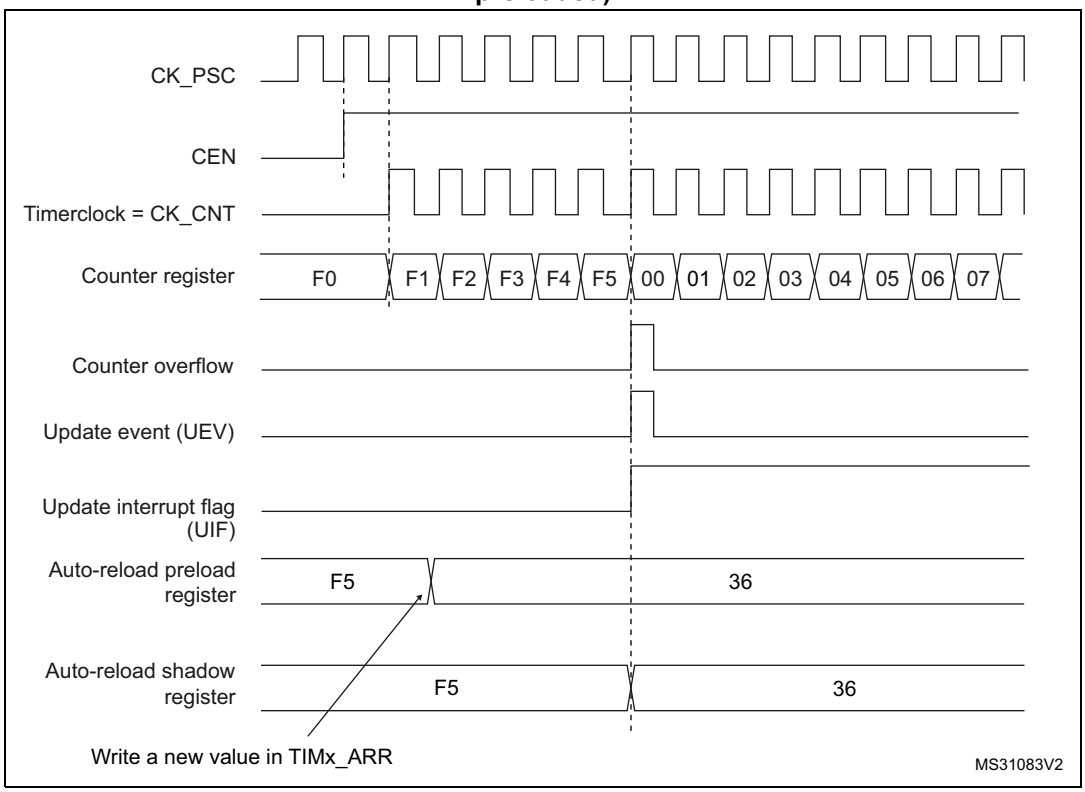


Figure 411. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 412. Counter timing diagram, internal clock divided by 1

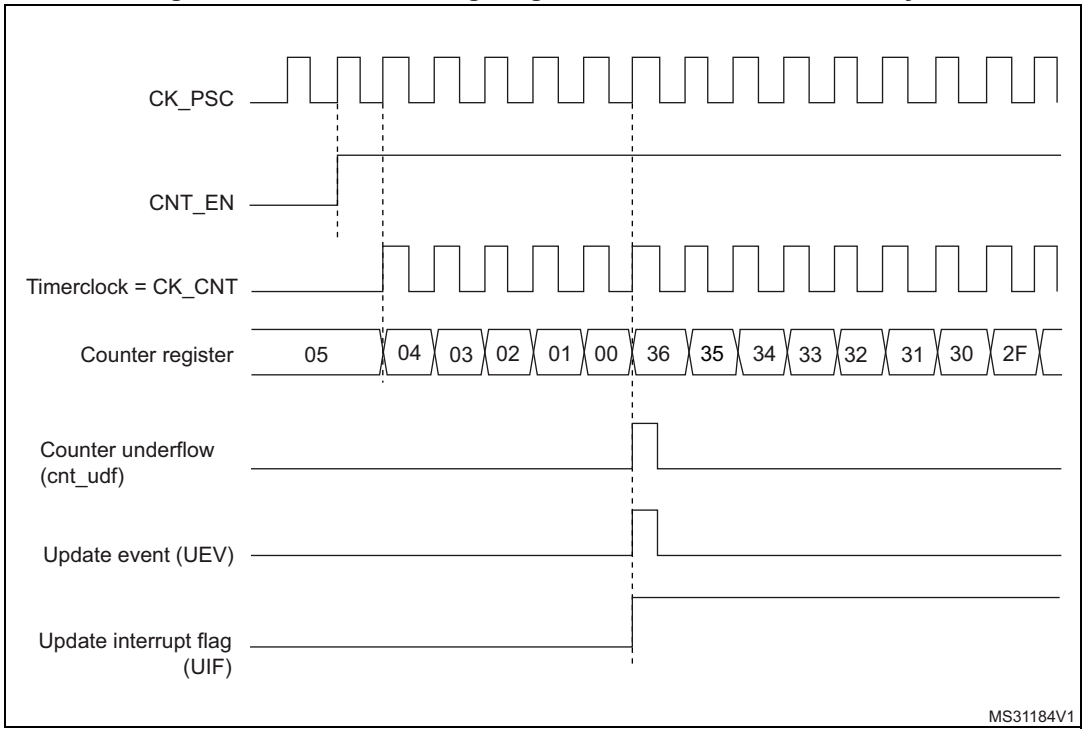


Figure 413. Counter timing diagram, internal clock divided by 2

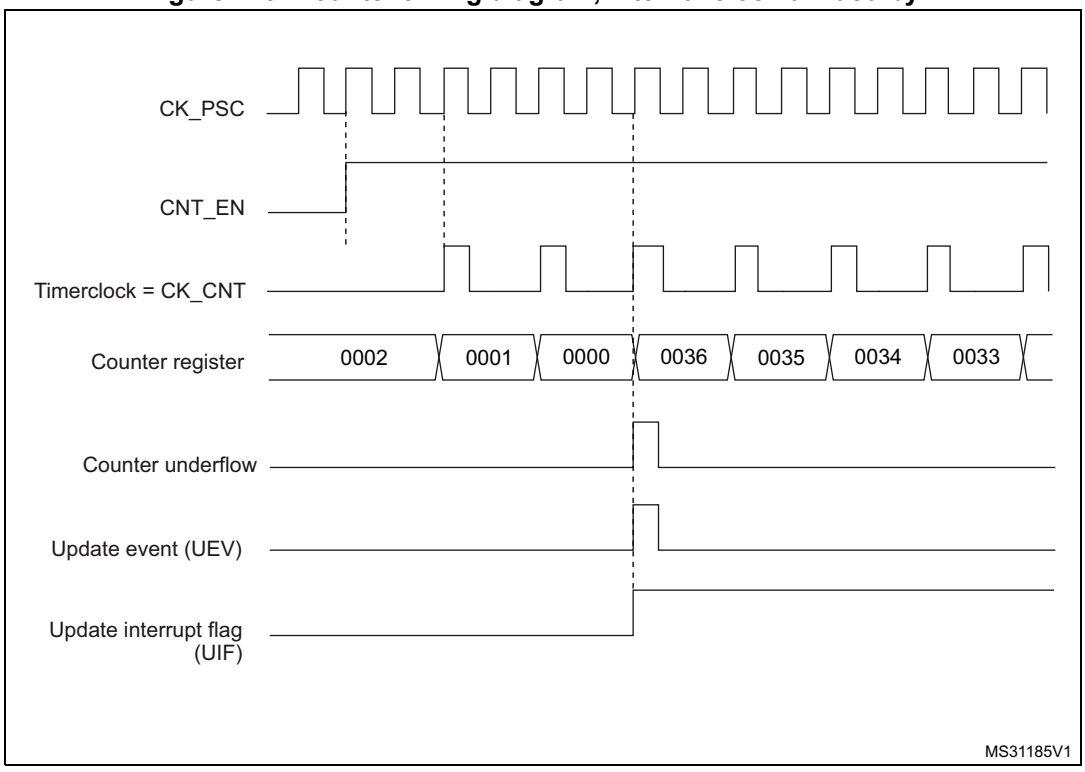


Figure 414. Counter timing diagram, internal clock divided by 4

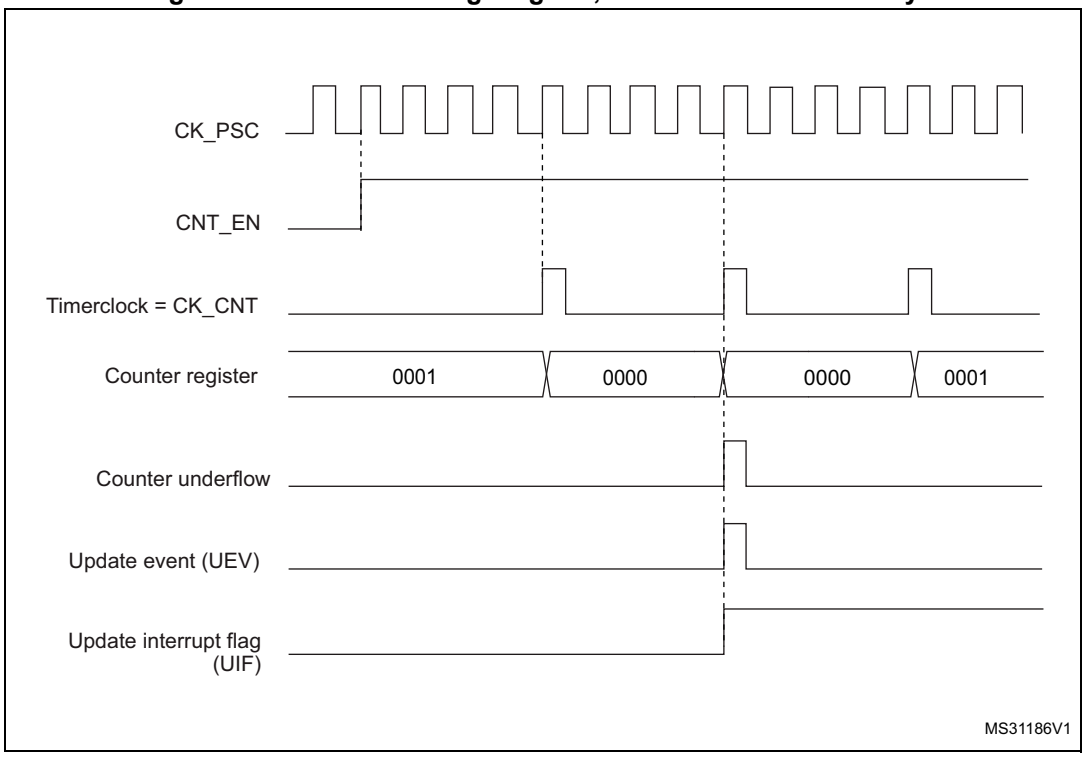


Figure 415. Counter timing diagram, internal clock divided by N

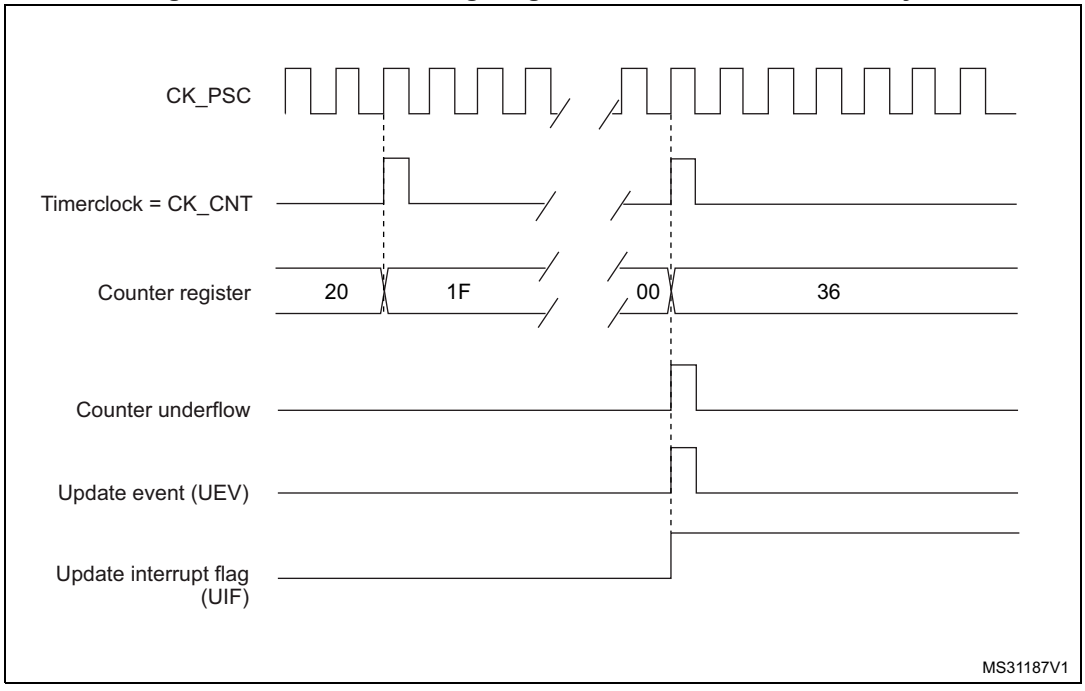
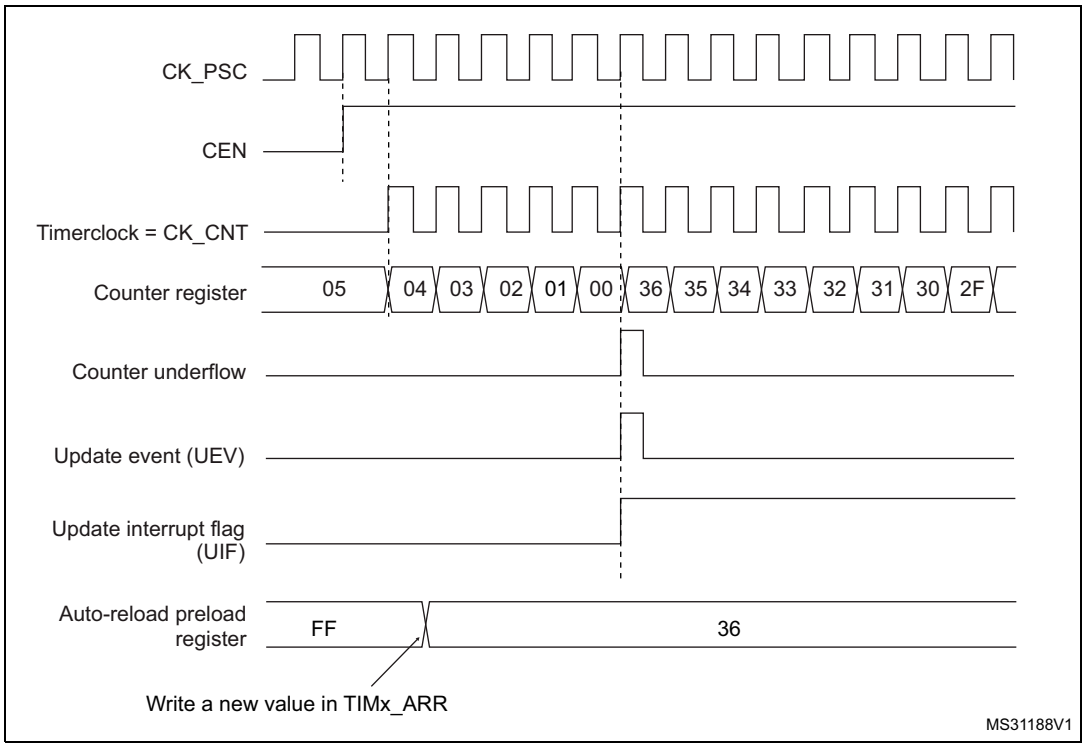


Figure 416. Counter timing diagram, Update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

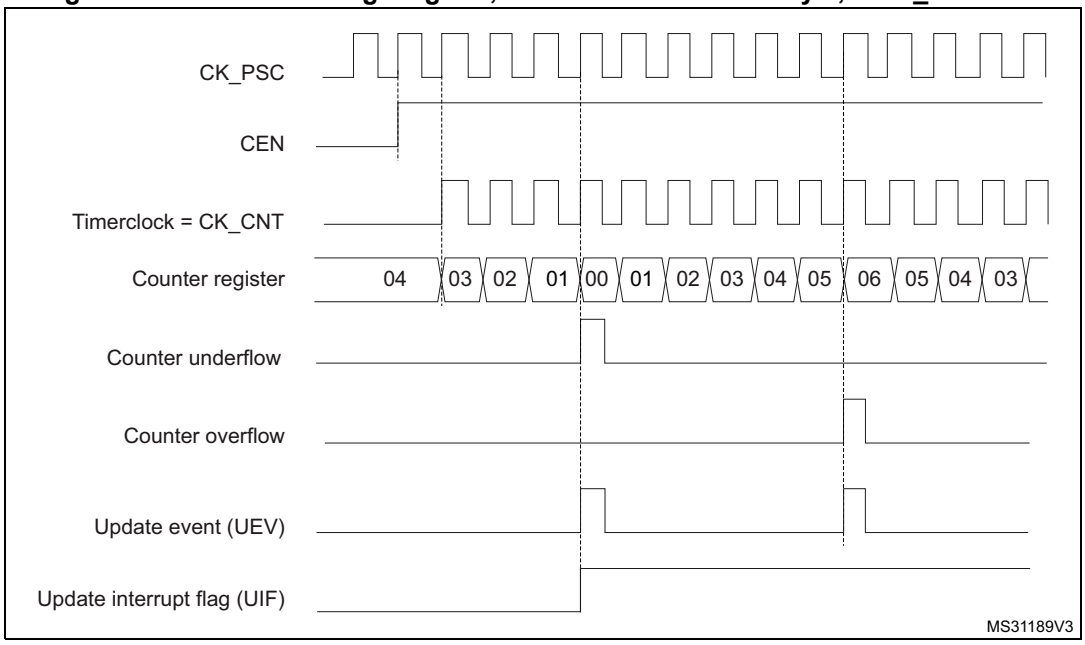
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 417. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 38.4.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5) on page 1664*).

Figure 418. Counter timing diagram, internal clock divided by 2

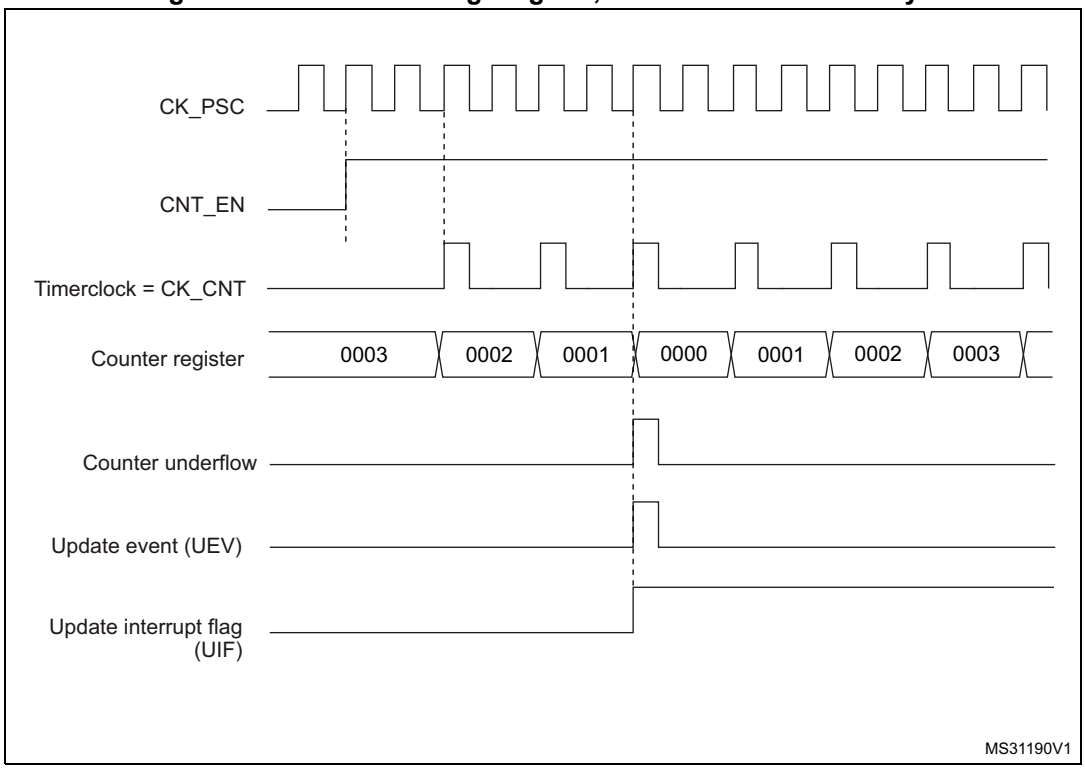
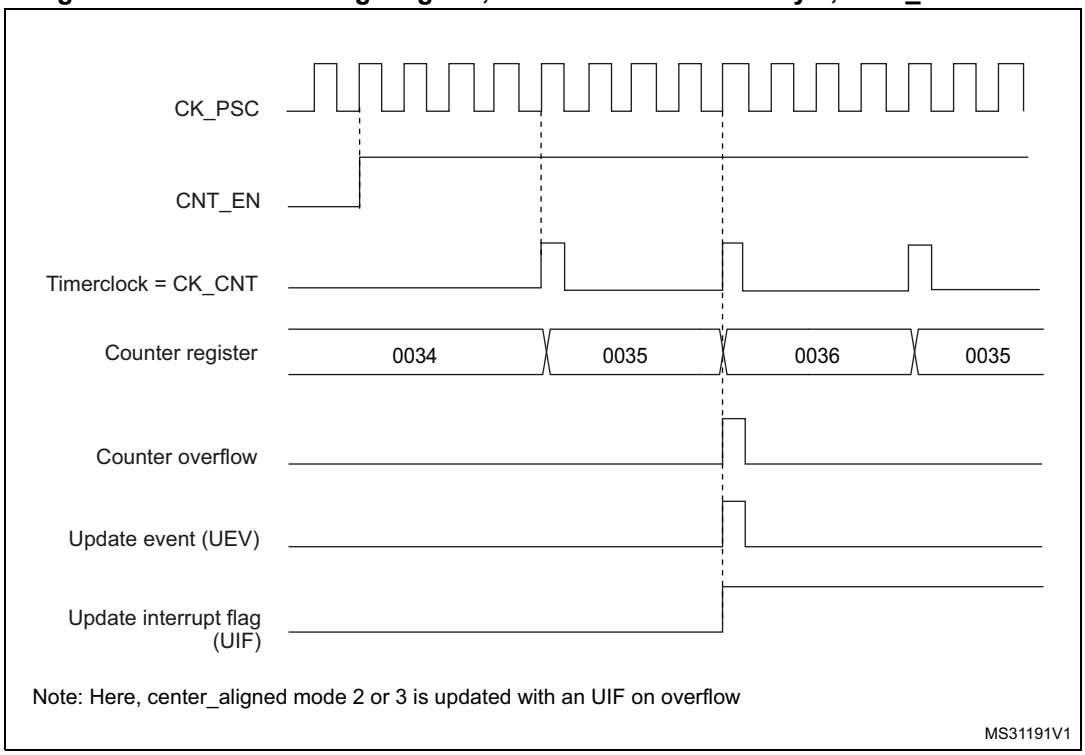


Figure 419. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 420. Counter timing diagram, internal clock divided by N

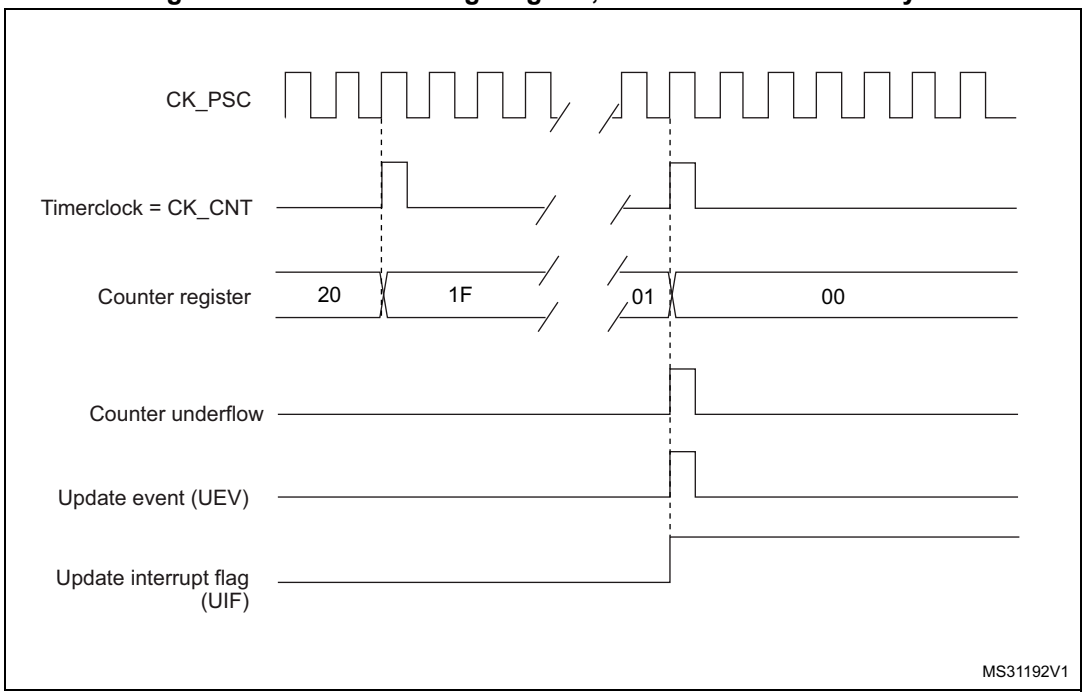


Figure 421. Counter timing diagram, Update event with ARPE=1 (counter underflow)

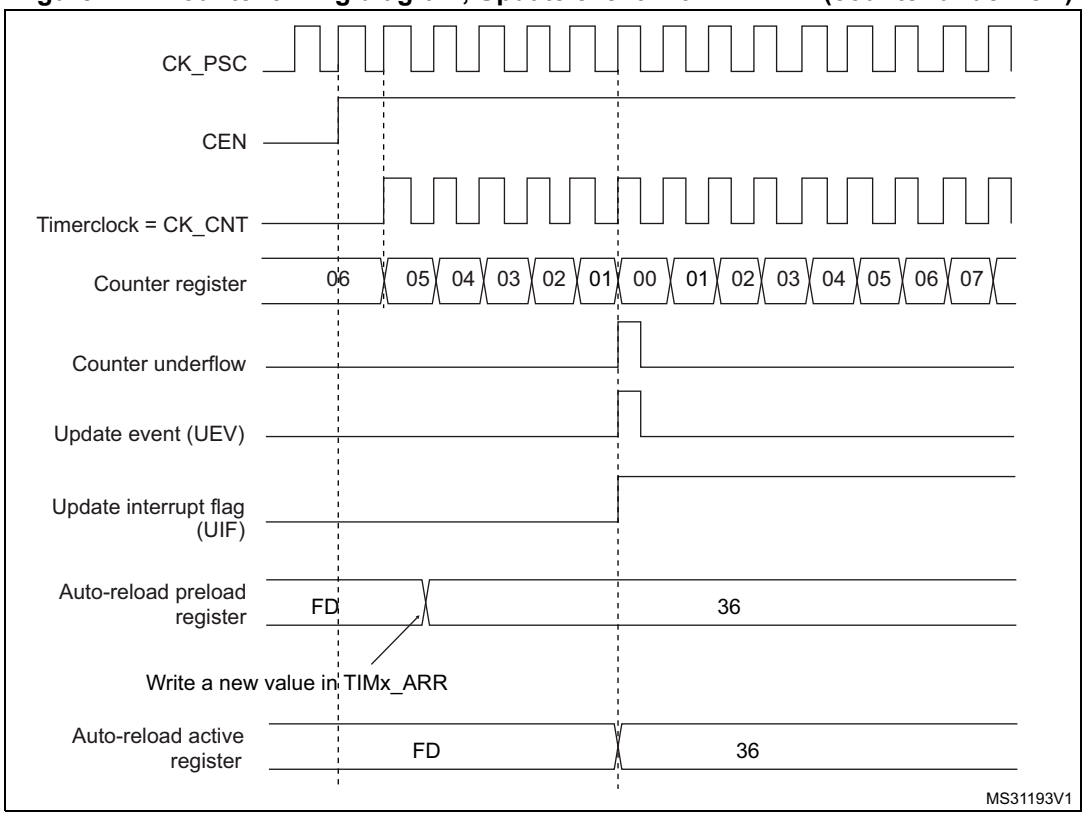
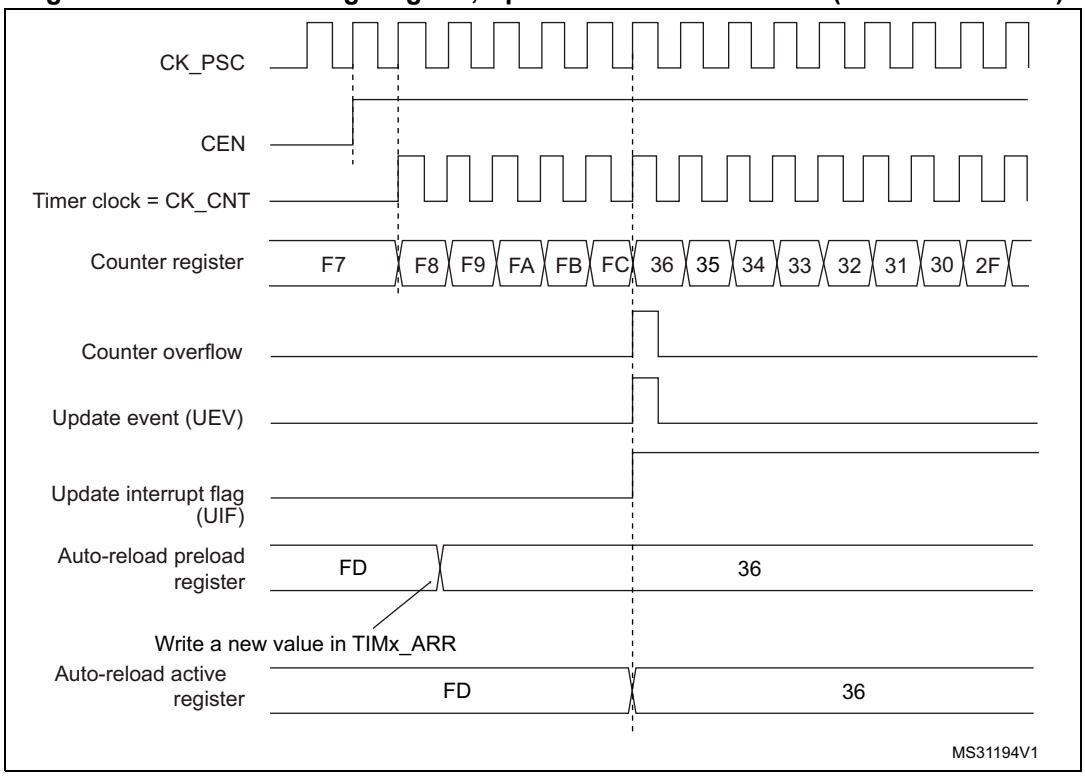


Figure 422. Counter timing diagram, Update event with ARPE=1 (counter overflow)



38.3.3 Clock selection

The counter clock can be provided by the following clock sources:

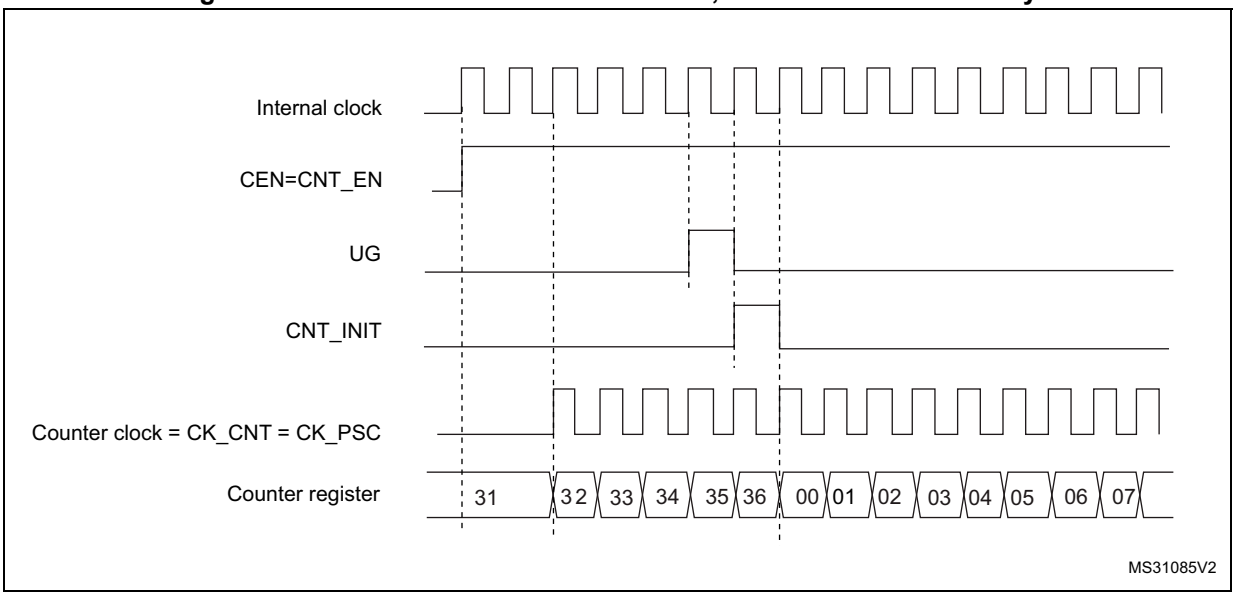
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer 13 to act as a prescaler for Timer 2. Refer to *: Using one timer as prescaler for another timer on page 1658* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 423 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

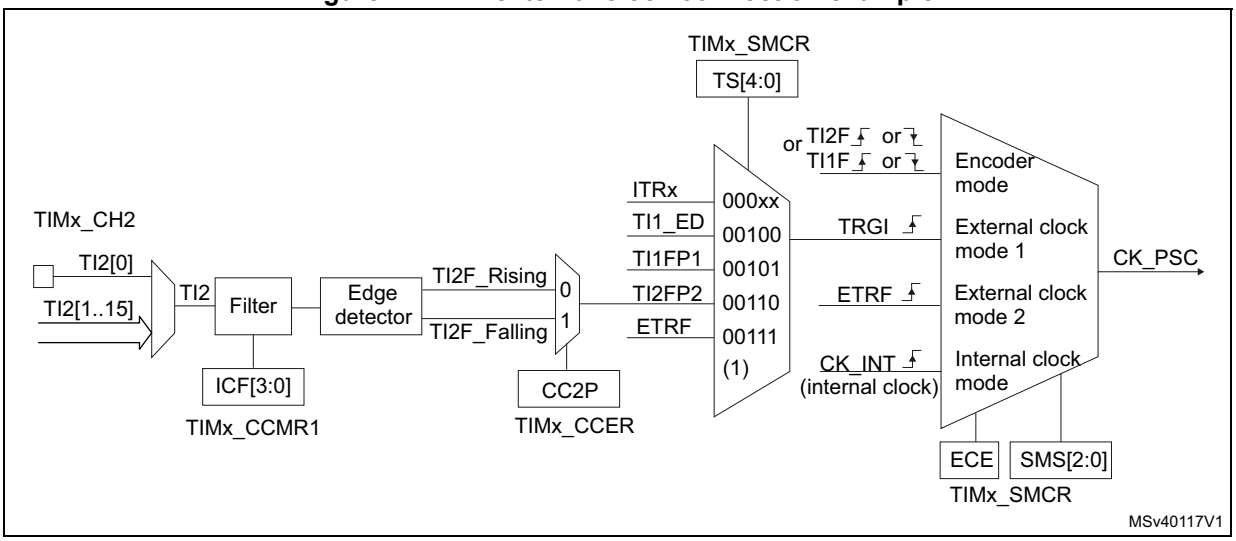
Figure 423. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 424. TI2 external clock connection example



1. Codes ranging from 01000 to 11111: ITRy.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

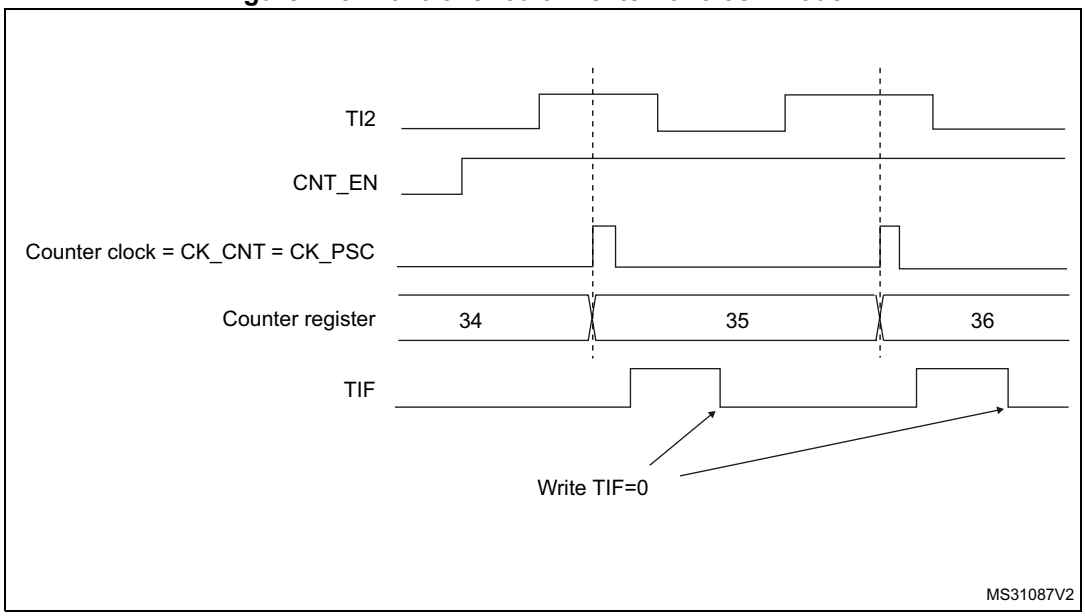
Note: The capture prescaler is not used for triggering, so you don’t need to configure it.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 425. Control circuit in external clock mode 1



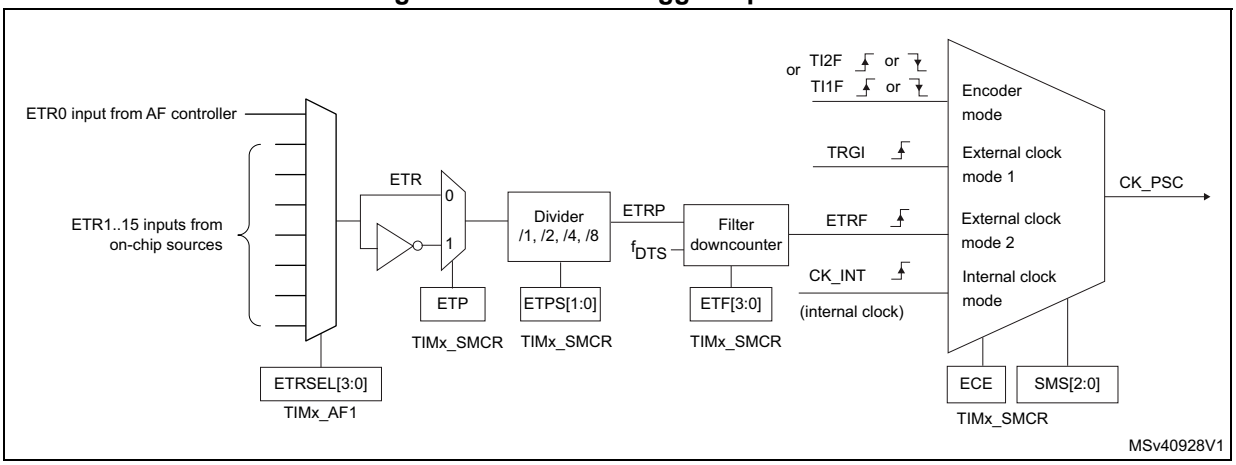
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Figure 426 gives an overview of the external trigger input block.

Figure 426. External trigger input block



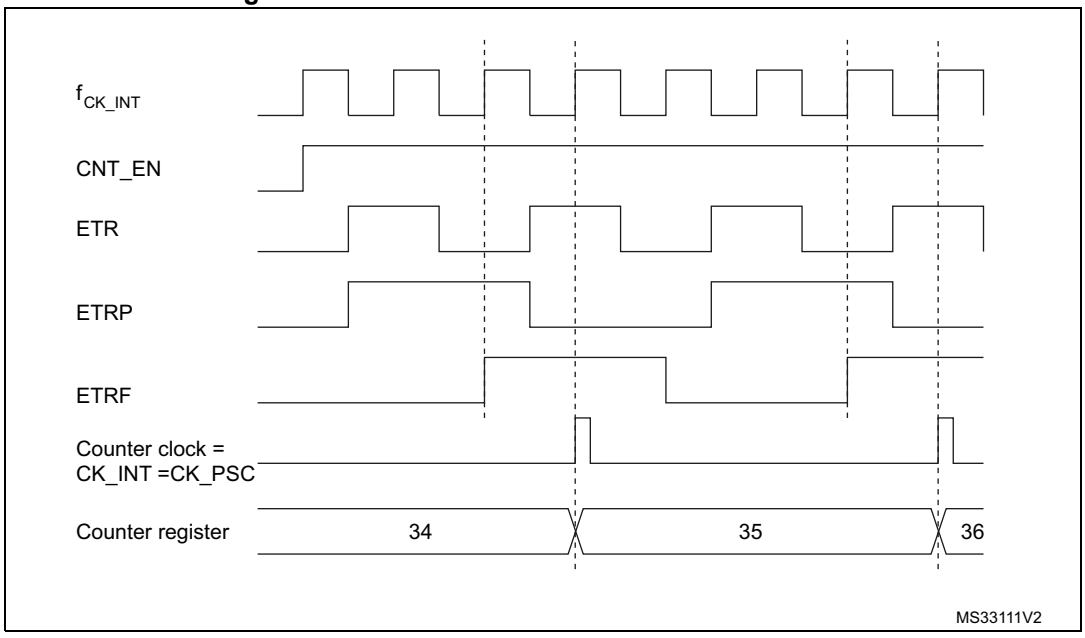
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 427. Control circuit in external clock mode 2



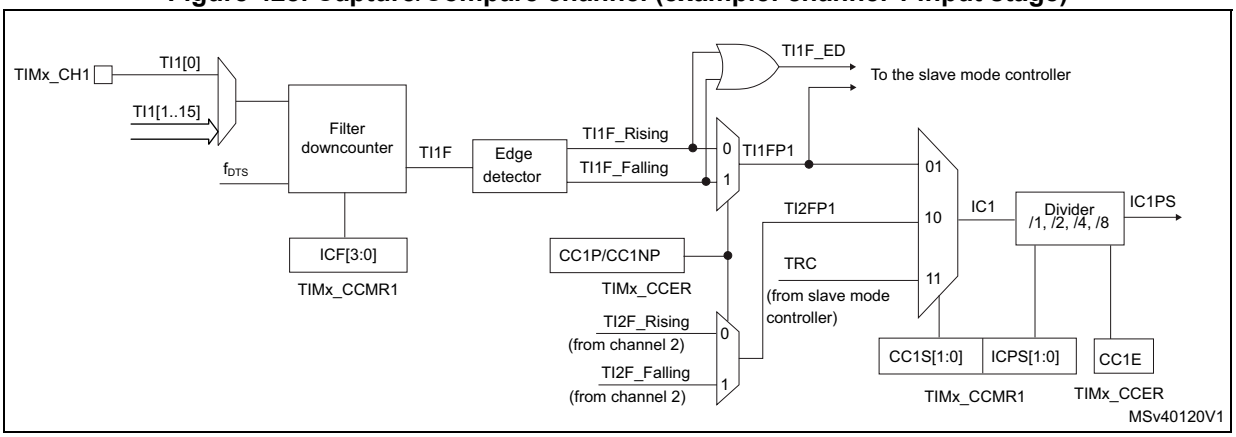
38.3.4 Capture/Compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 428. Capture/Compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 429. Capture/Compare channel 1 main circuit

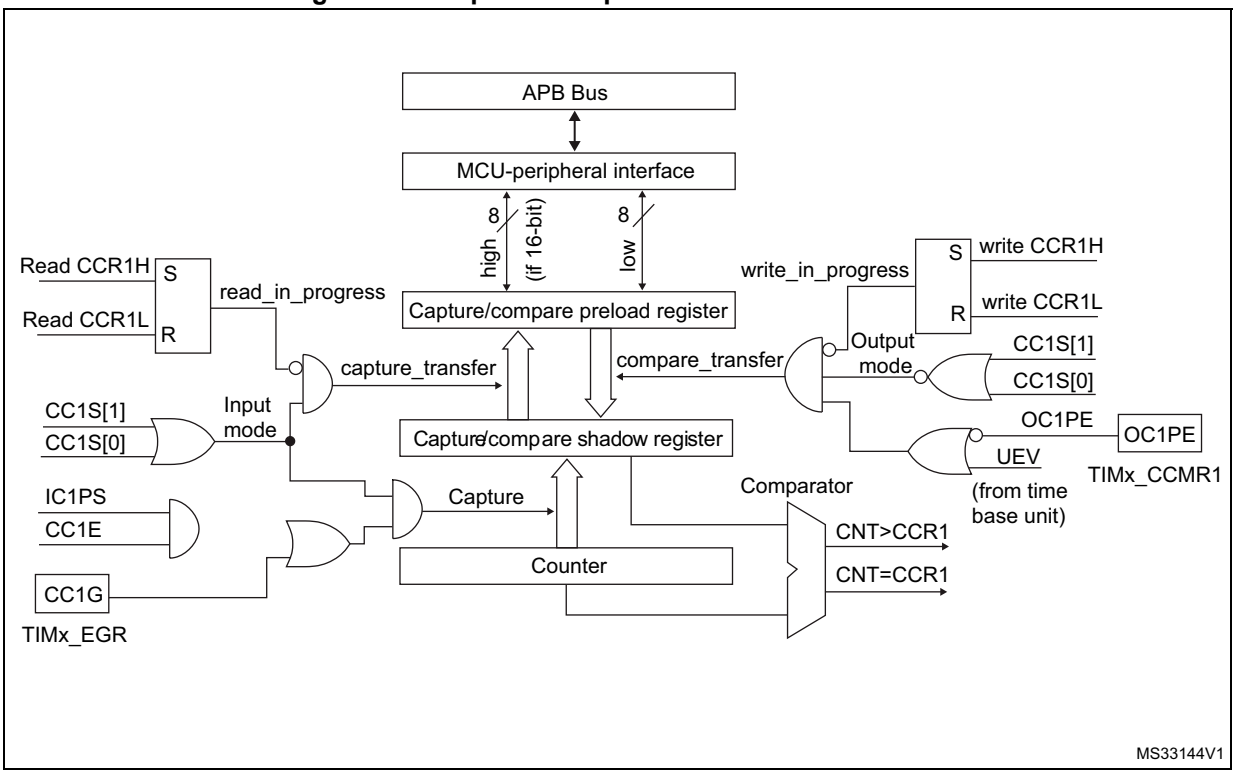
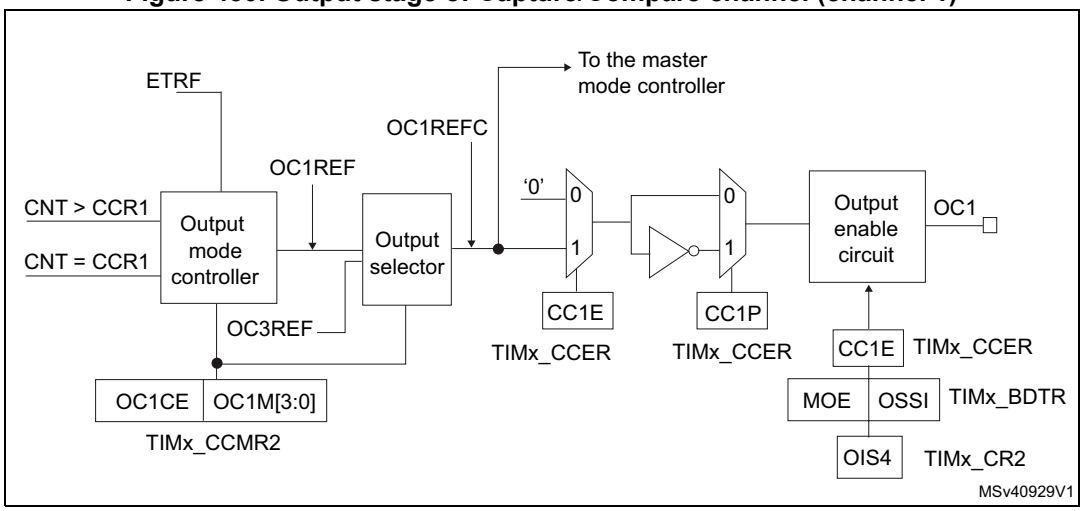


Figure 430. Output stage of Capture/Compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

38.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

38.3.6 PWM input mode

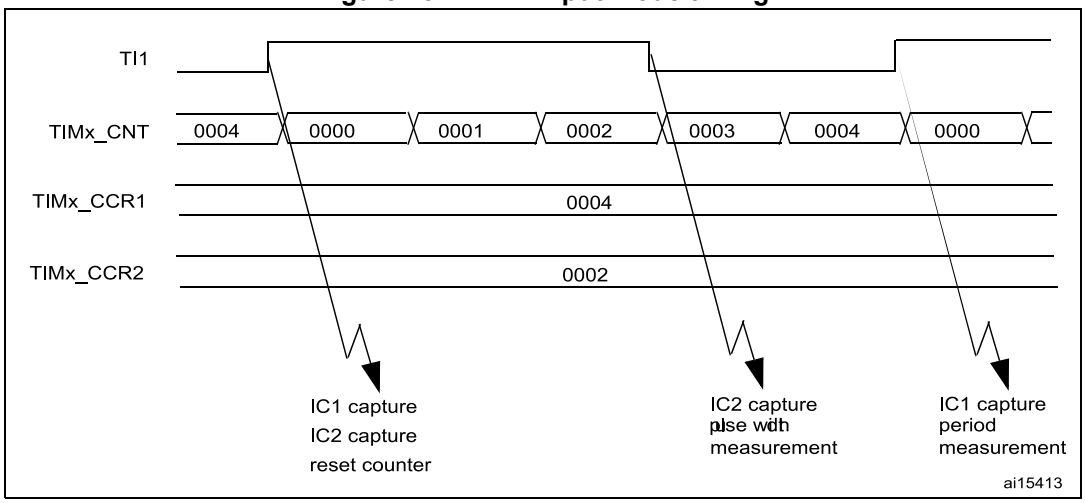
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ’0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1 in the TIMx_CCER register.

Figure 431. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

38.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

38.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

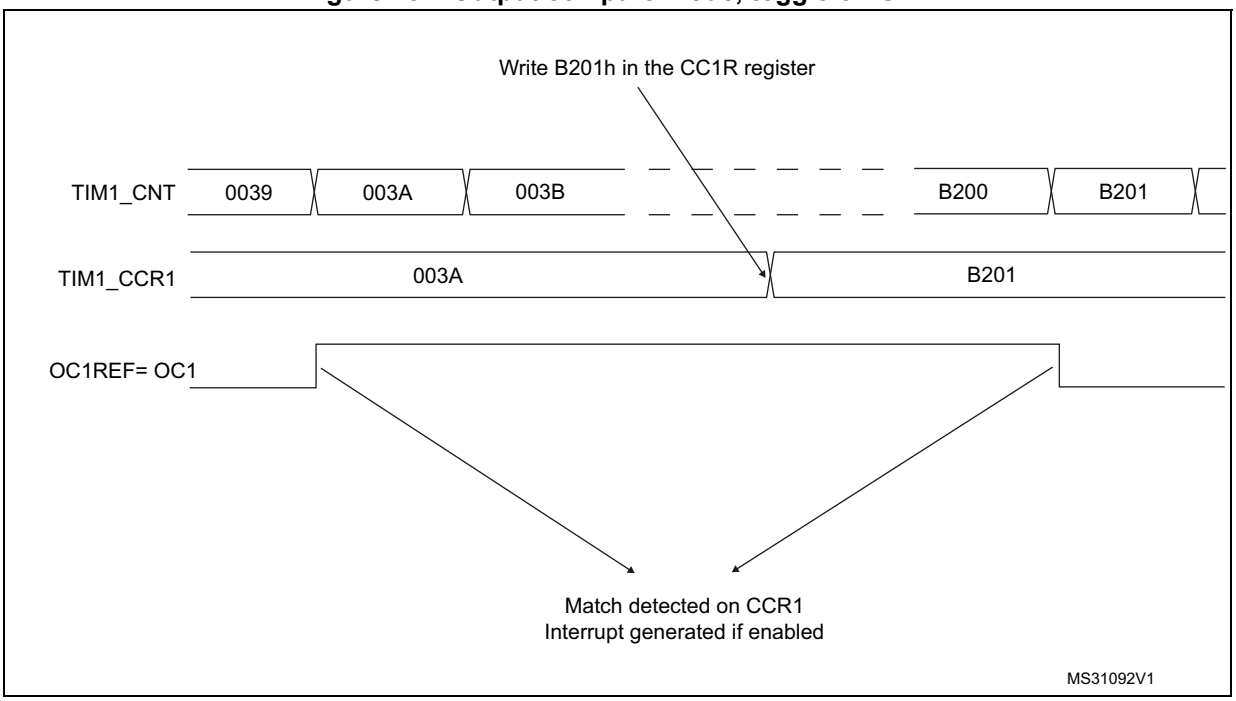
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, you must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 432*.

Figure 432. Output compare mode, toggle on OC1



38.3.9 PWM mode

Pulse width modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be

cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

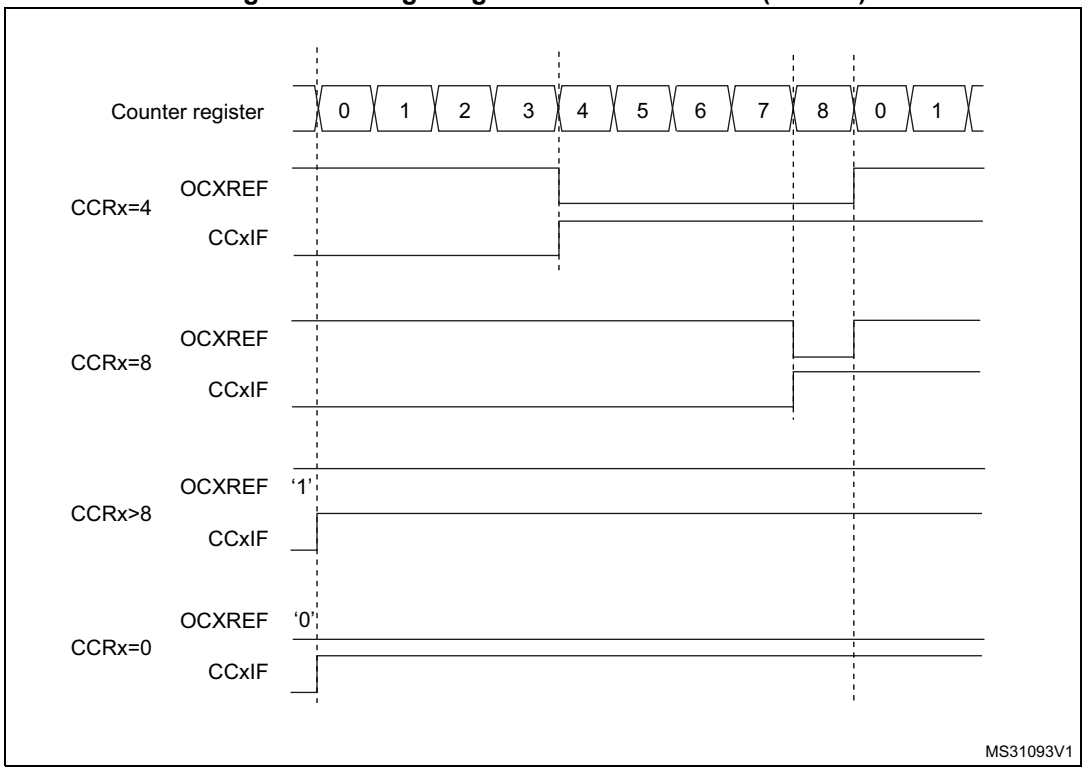
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 1623*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1. If the compare value is 0 then OCxREF is held at ‘0. *Figure 433* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 433. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 1626*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

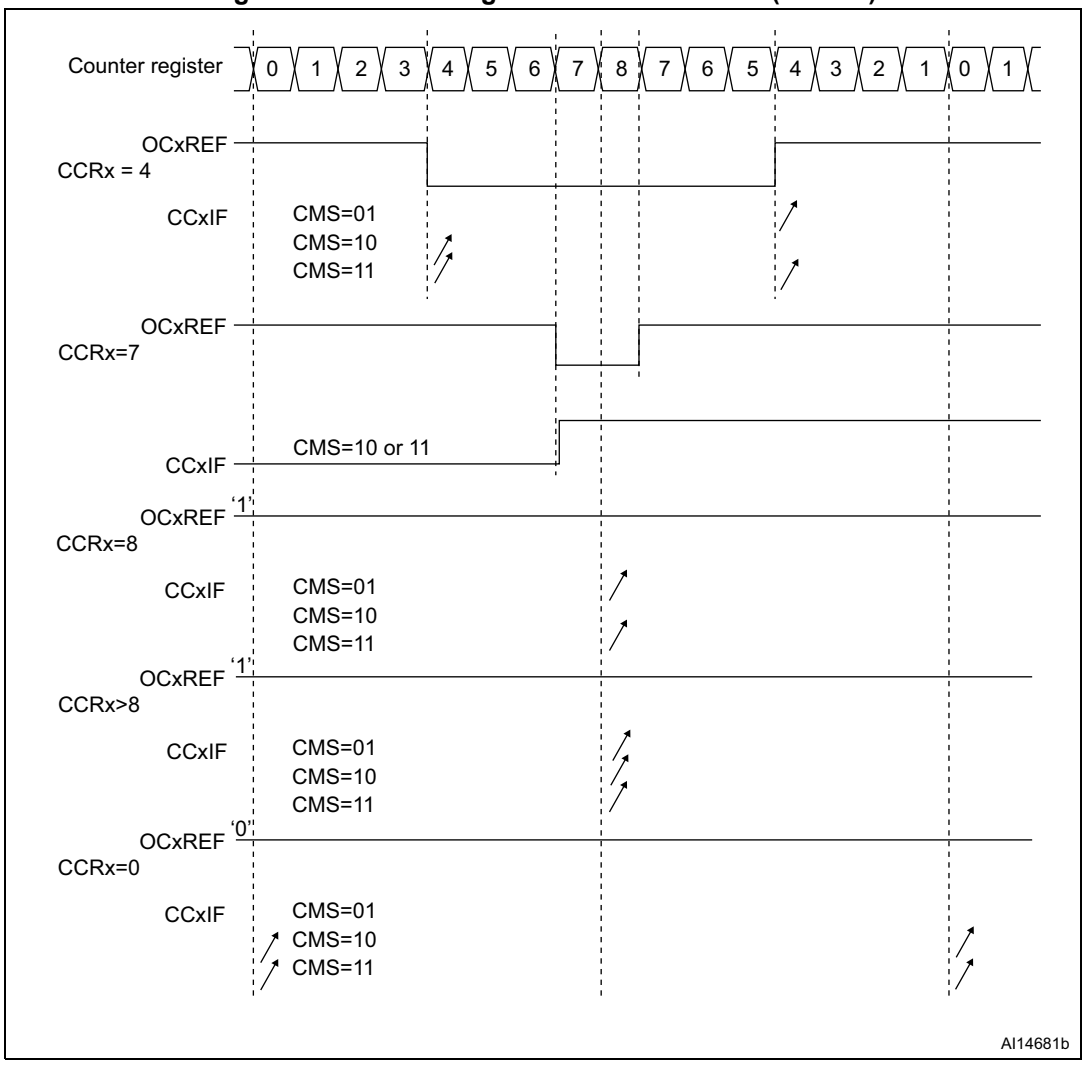
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 1629*.

Figure 434 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 434. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

38.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

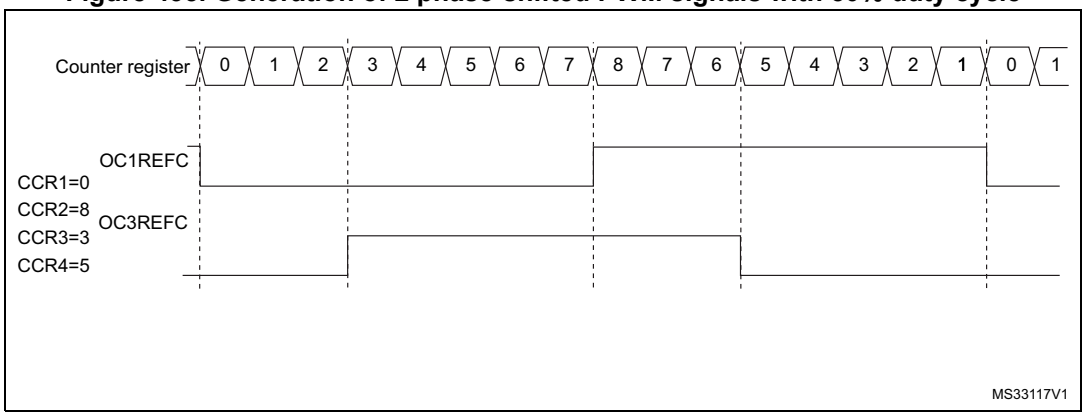
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

Figure 435 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

Figure 435. Generation of 2 phase-shifted PWM signals with 50% duty cycle



38.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

– OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2

– OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

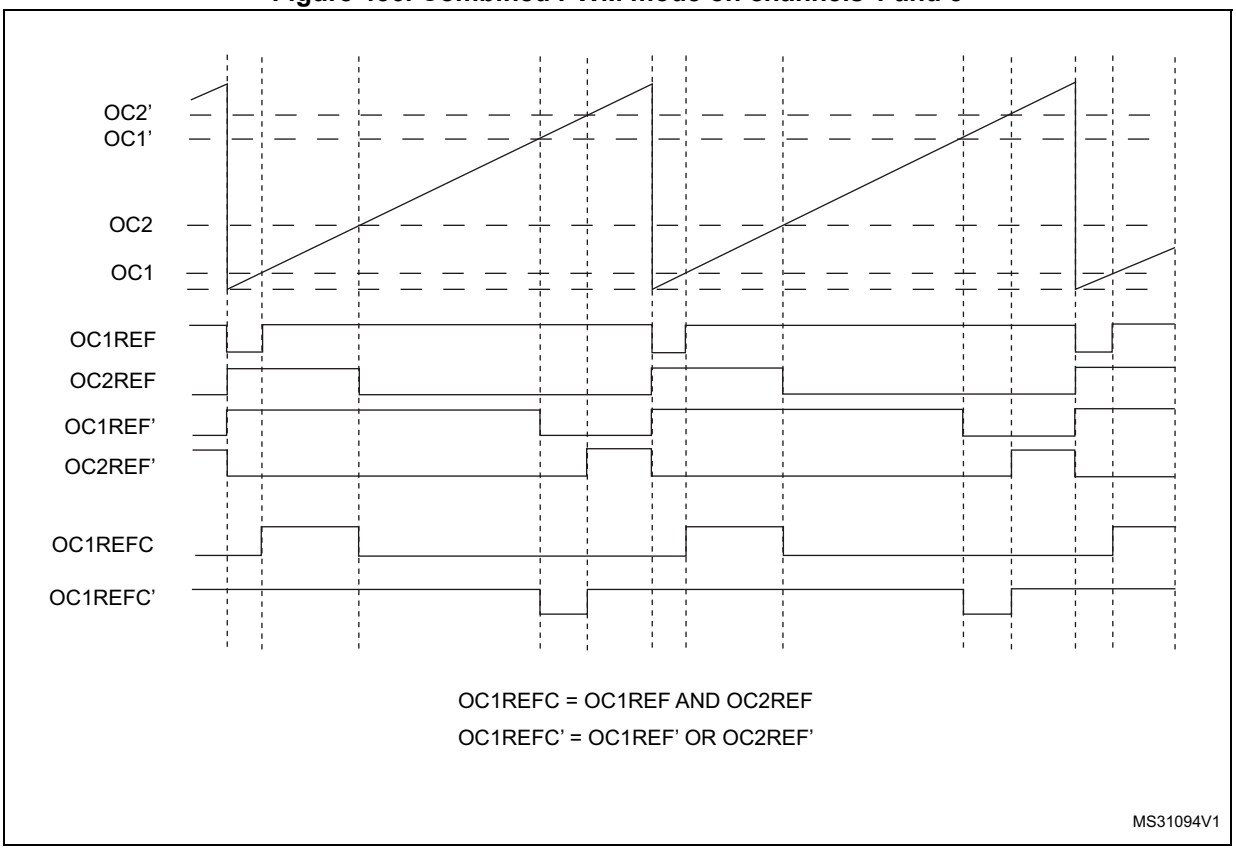
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 436 shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

Figure 436. Combined PWM mode on channels 1 and 3



38.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The ocref_clr_int is connected to the ETRF signal (ETR after filtering).

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

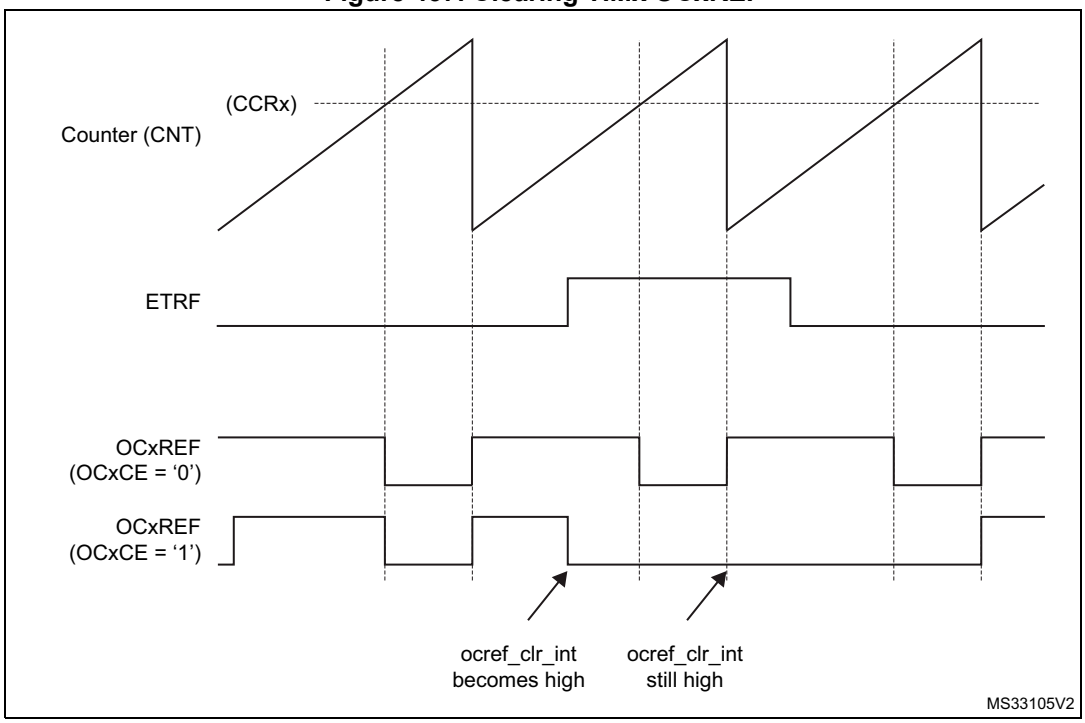
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 437 shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 437. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

38.3.13 One-pulse mode

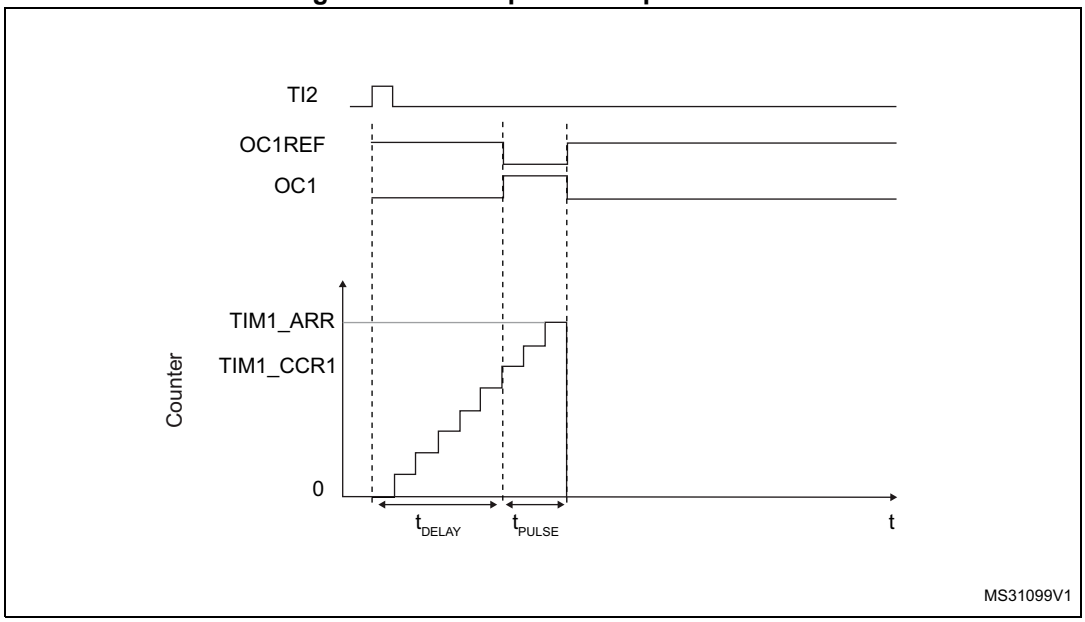
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

Figure 438. Example of one-pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

38.3.14 Retriggerable one pulse mode (OPM)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 38.3.13*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

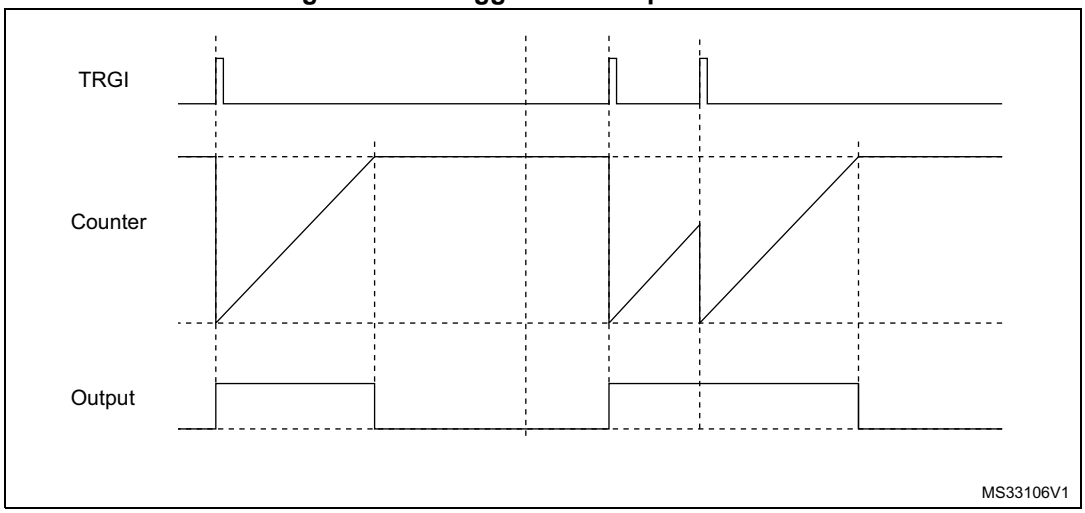
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 439Retriggerable one pulse mode



38.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, you can program the input filter as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 326*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don't switch at the same time.

Table 326. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 440 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 440. Example of counter operation in encoder interface mode

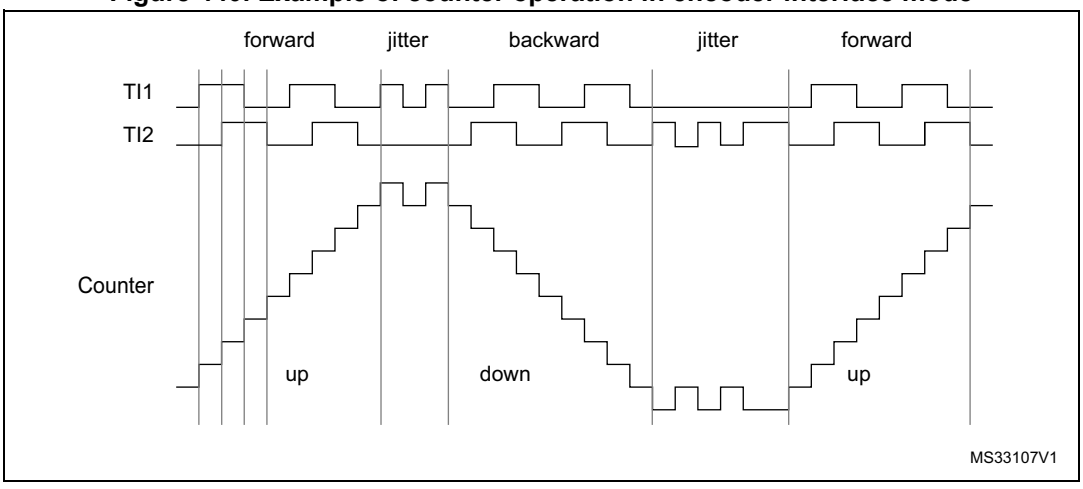
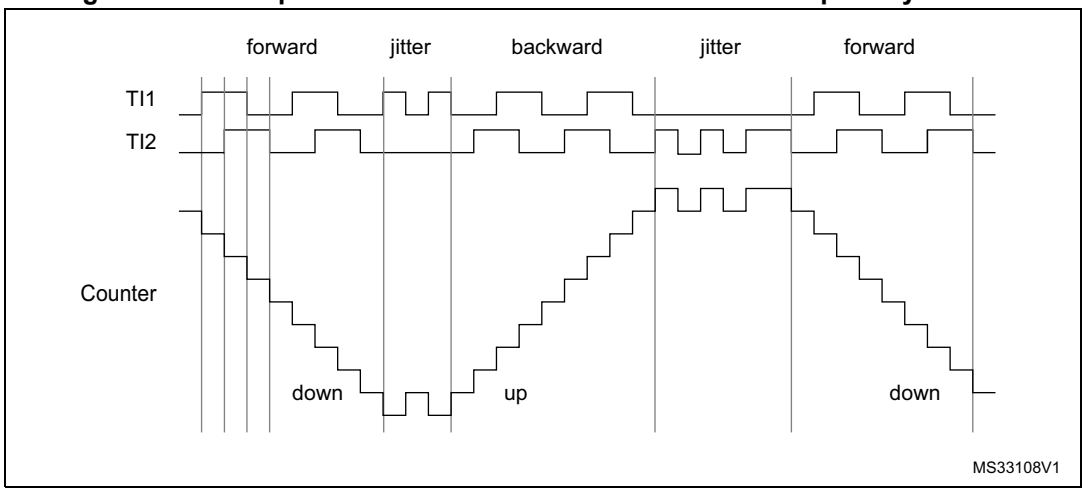


Figure 441 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 441. Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

38.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

38.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 37.3.25: Interfacing with Hall sensors on page 1564*.

38.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

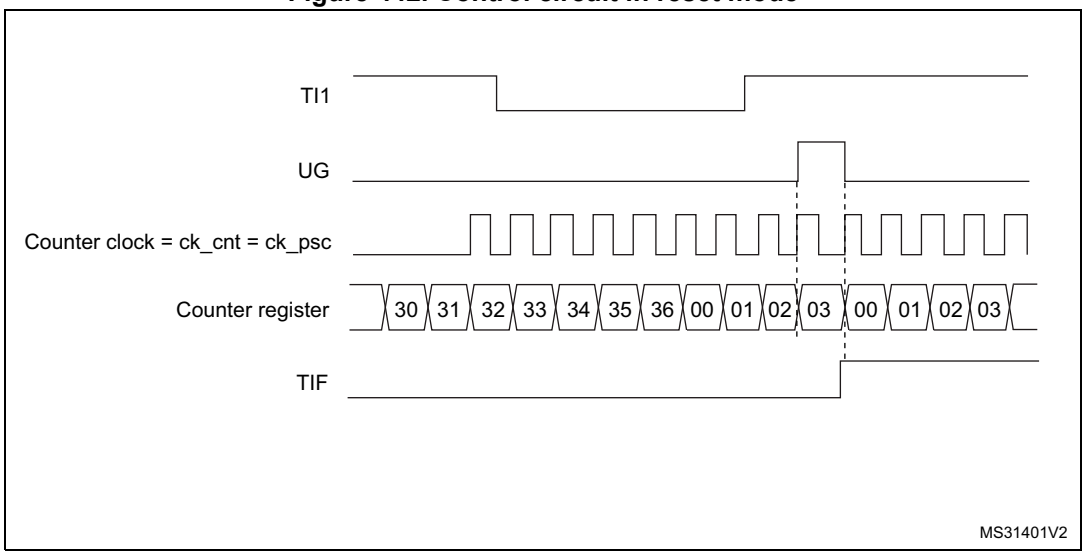
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 442. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

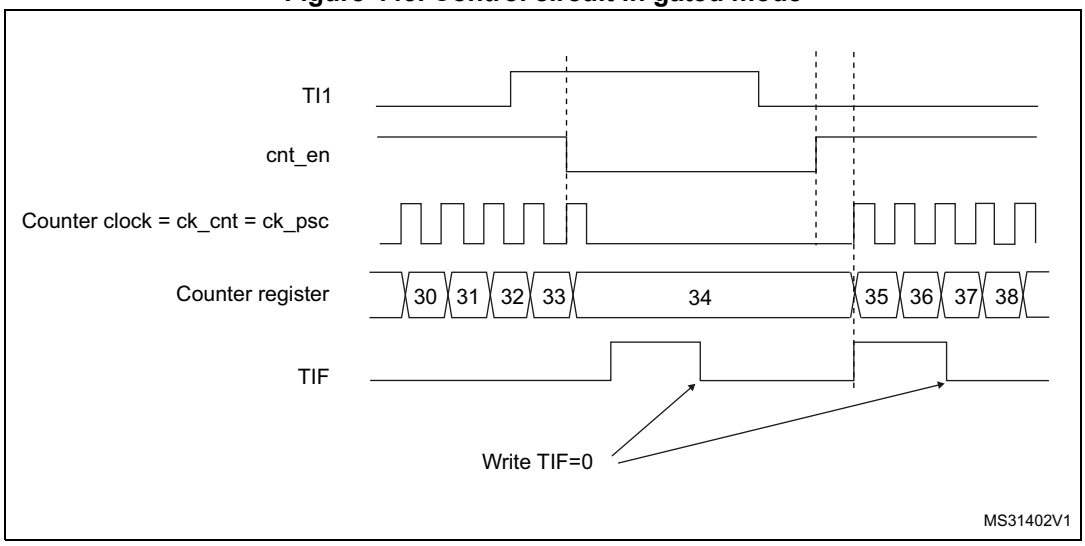
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 443. Control circuit in gated mode



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write

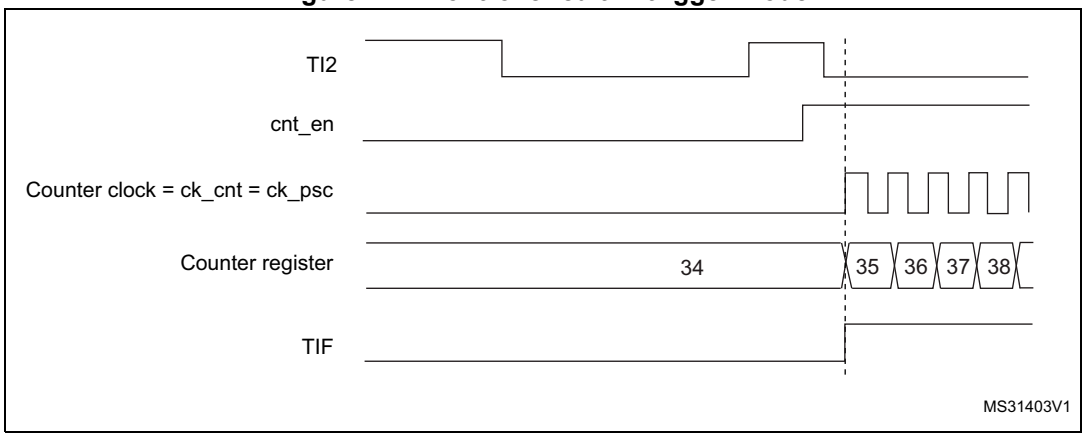
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 444. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

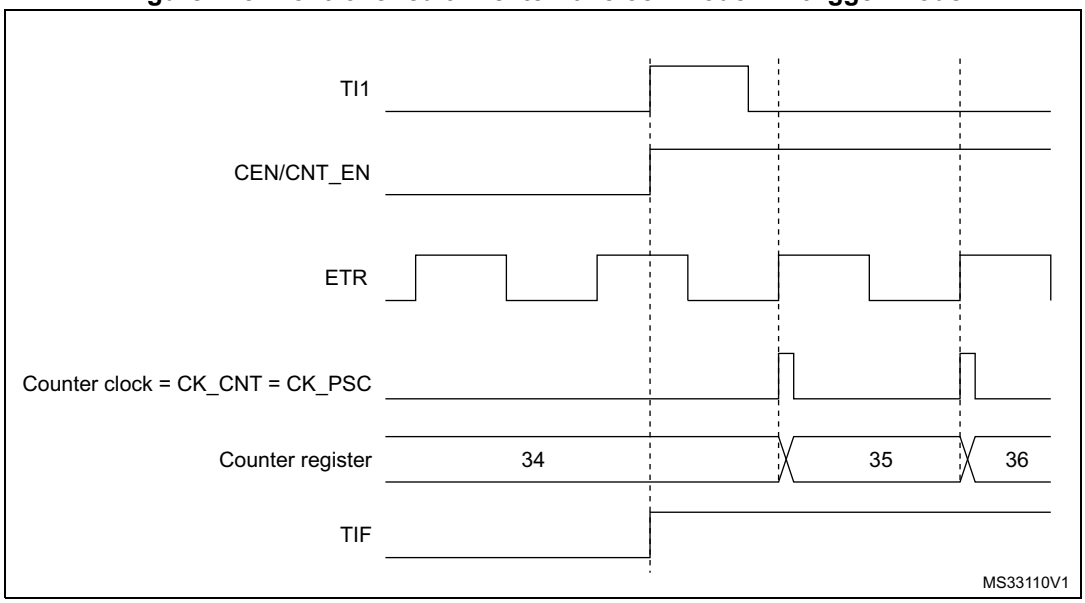
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 445. Control circuit in external clock mode 2 + trigger mode

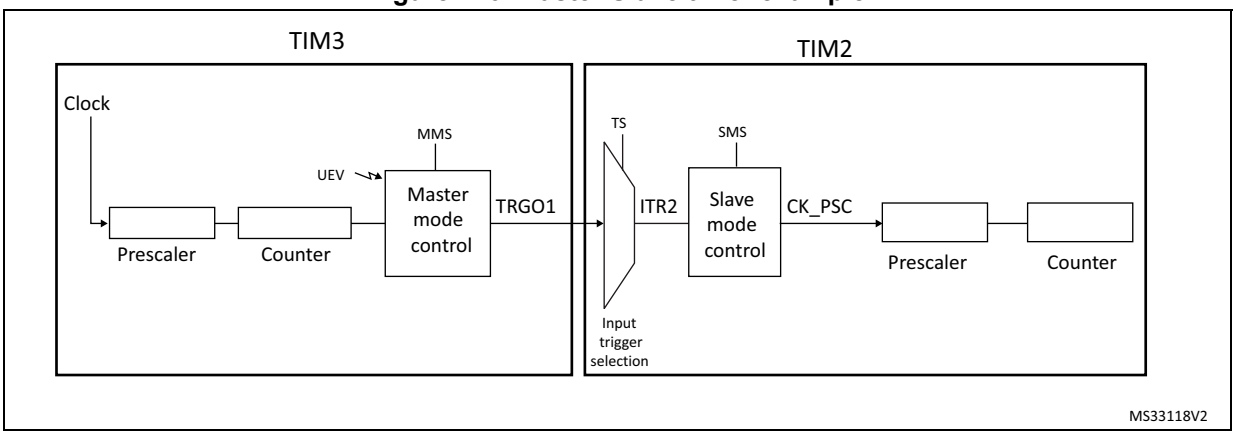


38.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 446: Master/Slave timer example presents an overview of the trigger selection and the master mode selection blocks.

Figure 446. Master/Slave timer example



Using one timer as prescaler for another timer

For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to *Figure 446*. To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM3_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. You select this through the TS bits in the TIM2_SMCR register (writing TS=00010).
3. Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

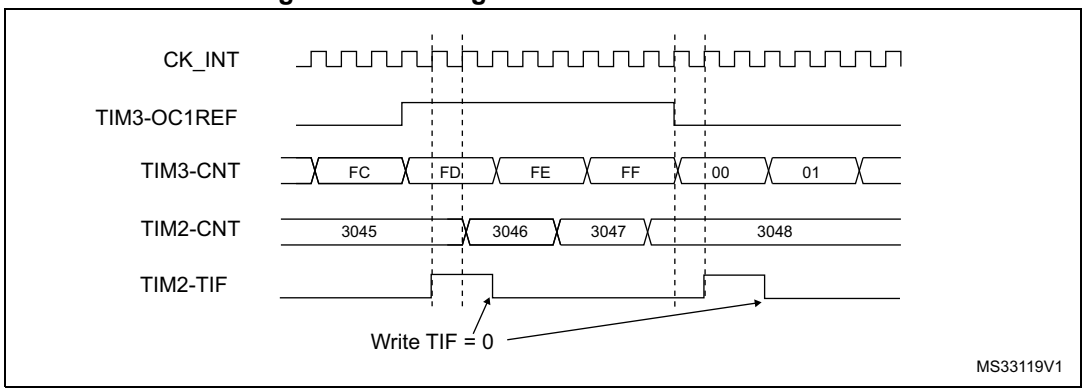
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to *Figure 446* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK_CNT} = f_{CK_INT}/3$).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
6. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 447. Gating TIM2 with OC1REF of TIM3

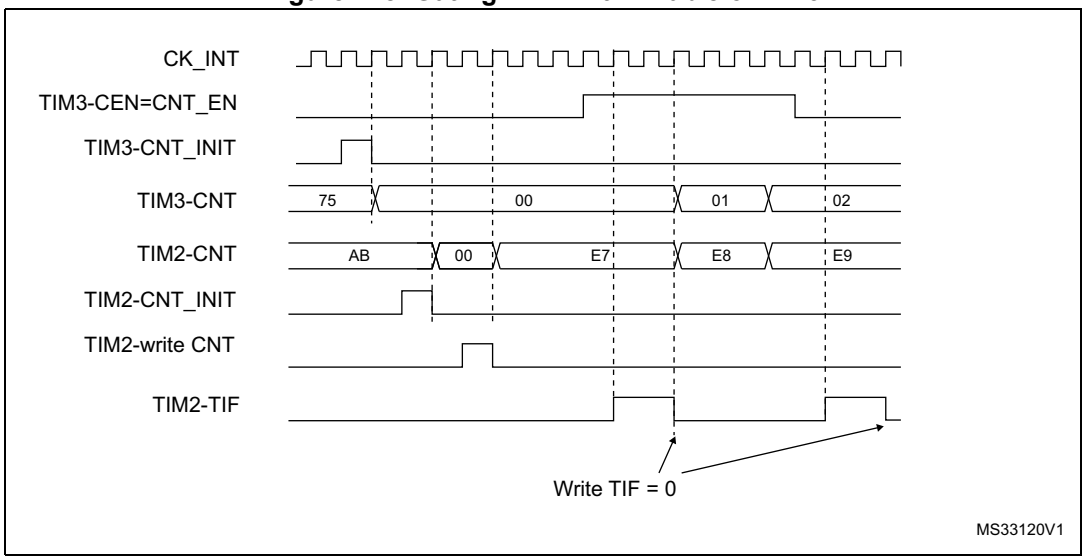


In the example in *Figure 447*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. You can then write any value you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 448*), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing ‘0 to the CEN bit in the TIM3_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM3 by writing ‘1 in UG bit (TIM3_EGR register).
6. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
9. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).
10. Stop TIM3 by writing ‘0 in the CEN bit (TIM3_CR1 register).

Figure 448. Gating TIM2 with Enable of TIM3

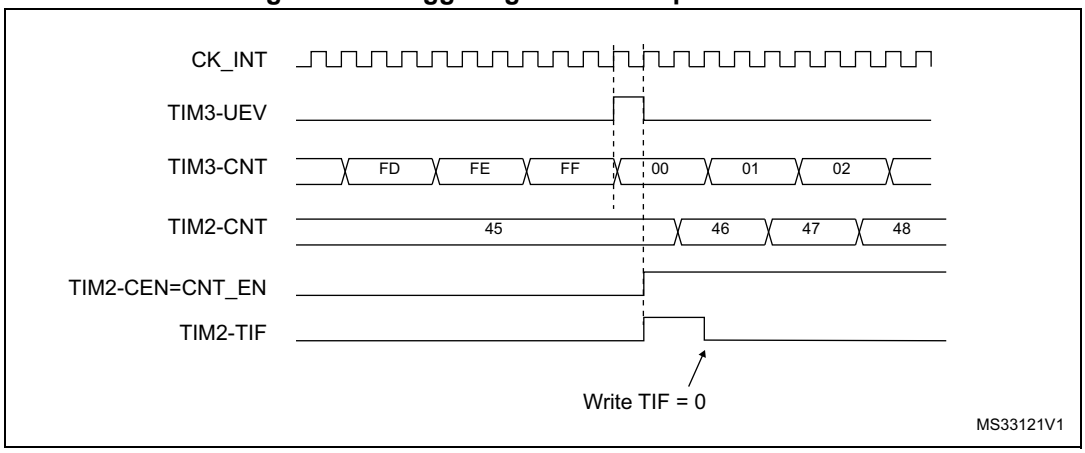


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to *Figure 446* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK_CNT} = f_{CK_INT}/3$).

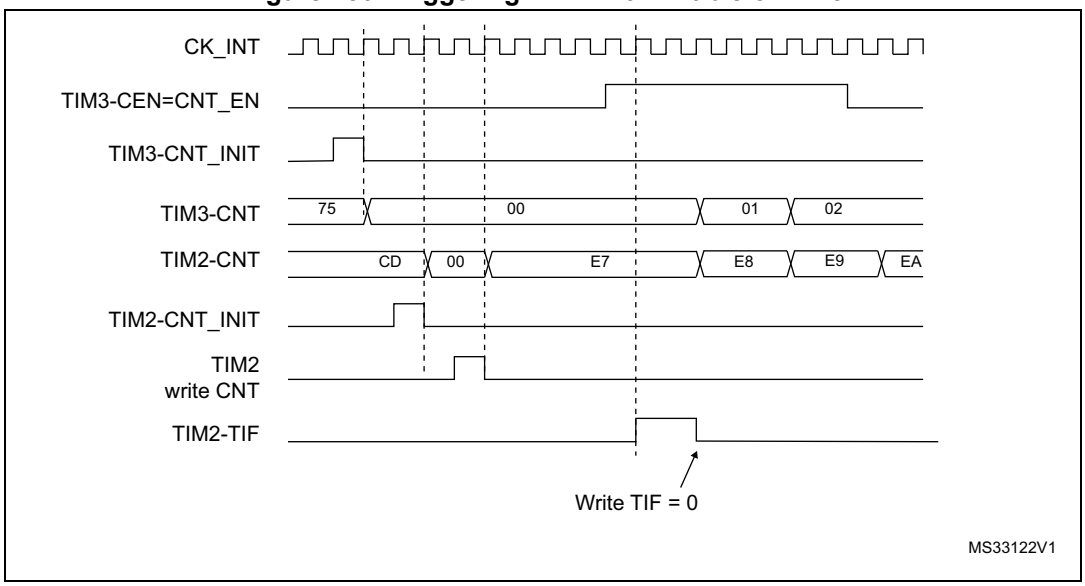
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Figure 449. Triggering TIM2 with update of TIM3



As in the previous example, you can initialize both counters before starting counting. *Figure 450* shows the behavior with the same configuration as in *Figure 449* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 450. Triggering TIM2 with Enable of TIM3



Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to *Figure 446* for connections. To ensure the counters are

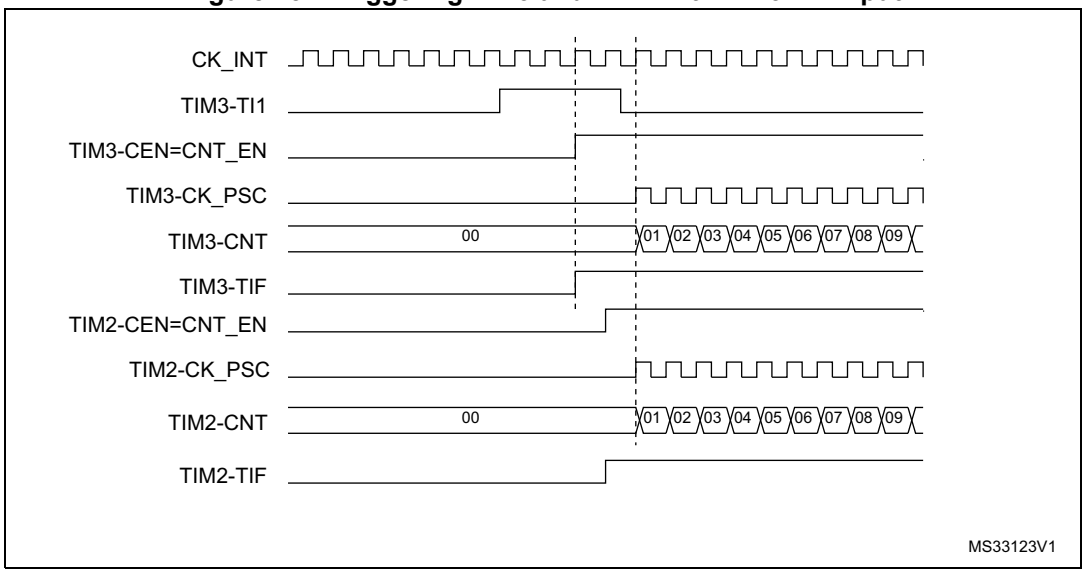
aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=00100 in the TIM3_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=00000 in the TIM2_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx_CNT). You can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.

Figure 451. Triggering TIM3 and TIM2 with TIM3 TI1 input



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

38.3.21 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 with FPU core halted), the TIMx counter either continues to work normally or stops, depending on TIMx configuration bit in DBGMCU module. For more details, refer to *Section 59.5.8: Microcontroller debug unit (DBGMCU)*.

For safety purposes, when the counter is stopped (TIMx = 1 in DBGMCU_APB1FZ2), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an

inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

38.4 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

38.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),
00: $t_{DTS} = t_{CK_INT}$
01: $t_{DTS} = 2 \times t_{CK_INT}$
10: $t_{DTS} = 4 \times t_{CK_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

38.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
See also *Section 37.3.25: Interfacing with Hall sensors on page 1564*

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)
100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

38.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **SMS[3]:** Slave mode selection - bit 3
Refer to SMS description - bits 2:0.

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge
1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).
2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).
3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler
External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.
00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection (see bits 21:20 for TS[4:3])

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
00111: External Trigger input (ETRF)
01000: Internal Trigger 4 (ITR4)
01001: Internal Trigger 5 (ITR5)
01010: Internal Trigger 6 (ITR6)
01011: Internal Trigger 7 (ITR7)
01100: Internal Trigger 8 (ITR8)
Others: Reserved

See *Table 327: TIMx internal trigger connection on page 1670* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.
0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 327. TIMx internal trigger connection

Slave TIM	TIM2	TIM3	TIM4	TIM5
ITR0	TIM1	TIM1	TIM1	TIM1
ITR1	TIM8	TIM2	TIM2	TIM8
ITR2	TIM3	TIM15	TIM3	TIM3
ITR3	TIM4	TIM4	TIM8	TIM4
ITR4	ETH PPS	ETH PPS	-	-
ITR5	USB1 OTG_HS_SOF	-	-	-
ITR6	USB2 OTG_FS_SOF	-	-	fdcan1_soc
ITR7	-	-	-	USB1 OTG_HS_SOF
ITR8	-	-	-	USB2 OTG_FS_SOF

38.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

38.4.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description) and in retriggerable one pulse mode. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT has matched the content of the TIMx_CCR1 register.
If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow (for TIM2 to TIM4) and if UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

38.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

38.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							Res.								Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).*

***2:** In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

38.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							Res.								Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0.

Bit 24 **OC4M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0.

Bit 16 **OC3M[3]**: Output Compare 1 mode - bit 3

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0.

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

38.4.9 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output: CC1NP must be kept cleared in this case.
CC1 channel configured as input: This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input: CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

CC1 channel configured as output:
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin
CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 328. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Polarity, OCx_EN=1

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

38.4.10 TIMx counter (TIMx_CNT)(x = 2 to 5)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31] or UIFCPY	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value (on TIM2 and TIM5)
Reserved on other timers
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 **CNT[30:16]**: Most significant part counter value (on TIM2 and TIM5)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

38.4.11 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

38.4.12 TIMx auto-reload register (TIMx_ARR)(x = 2 to 5)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **ARR[31:16]**: High auto-reload value (on TIM2 and TIM5)

Bits 15:0 **ARR[15:0]**: Low Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 38.3.1: Time-base unit on page 1621* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

38.4.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (on TIM2 and TIM5)

Bits 15:0 **CCR1[15:0]**: Low Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

38.4.14 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (on TIM2 and TIM5)

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

38.4.15 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (on TIM2 and TIM5)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

If channel CC3 is configured as output:
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.
If channel CC3is configured as input:
CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

38.4.16 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (on TIM2 and TIM5)

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):
   CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
   It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

38.4.17 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...
Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

38.4.18 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

38.4.19 TIM2 alternate function option register 1 (TIM2_AF1)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: COMP1 output
0010: COMP2 output
0011: LSE
0100: SAI1 FS_A
0101: SAI1 FS_B
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:0 Reserved, must be kept at reset value.

38.4.20 TIM3 alternate function option register 1 (TIM3_AF1)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: COMP1 output
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:0 Reserved, must be kept at reset value.

38.4.21 TIM4 alternate function option register 1 (TIM4_AF1)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: ETR input is connected to I/O
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:0 Reserved, must be kept at reset value.

38.4.22 TIM5 alternate function option register 1 (TIM5_AF1)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: SAI2 FS_A connected to ETR input
0010: SAI2 FS_B connected to ETR input
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:0 Reserved, must be kept at reset value.

38.4.23 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM2_CH4 input
0001: COMP1 output
0010: COMP2 output
0011: COMP1 output OR COMP2 output
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM2_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM2_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM2_CH1 input
Others: Reserved

38.4.24 TIM3 timer input selection register (TIM3_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM3_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM3_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM3_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM3_CH1 input
0001: COMP1 output
0010: COMP2 output
0011: COMP1 output OR COMP2 output
Others: Reserved

38.4.25 TIM4 timer input selection register (TIM4_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM4_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM4_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM4_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM4_CH1 input
Others: Reserved

38.4.26 TIM5 timer input selection register (TIM5_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM5_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM5_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM5_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM5_CH1 input
0001: fdcan1_tmp
0010: fdcan1_rtp
Others: Reserved

38.4.27 TIMx register map

TIMx registers are mapped as described in the table below:

Table 329. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Table 329. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIMx_CNT**	CNT[31] or UIFCPY	CNT[30:16] (TIM2 and TIM5 only, reserved on the other timers)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[15:0]															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	Reserved																																
0x34	**TIMx_CCR1**	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM2_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x60	**TIM3_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														

Table 329. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	**TIM4_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x60	**TIM5_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x68	**TIM2_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x68	**TIM3_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x68	**TIM4_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x68	**TIM5_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

39 General-purpose timers (TIM12/TIM13/TIM14)

39.1 TIM12/TIM13/TIM14 introduction

The TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 39.3.17: Timer synchronization (TIM12)*.

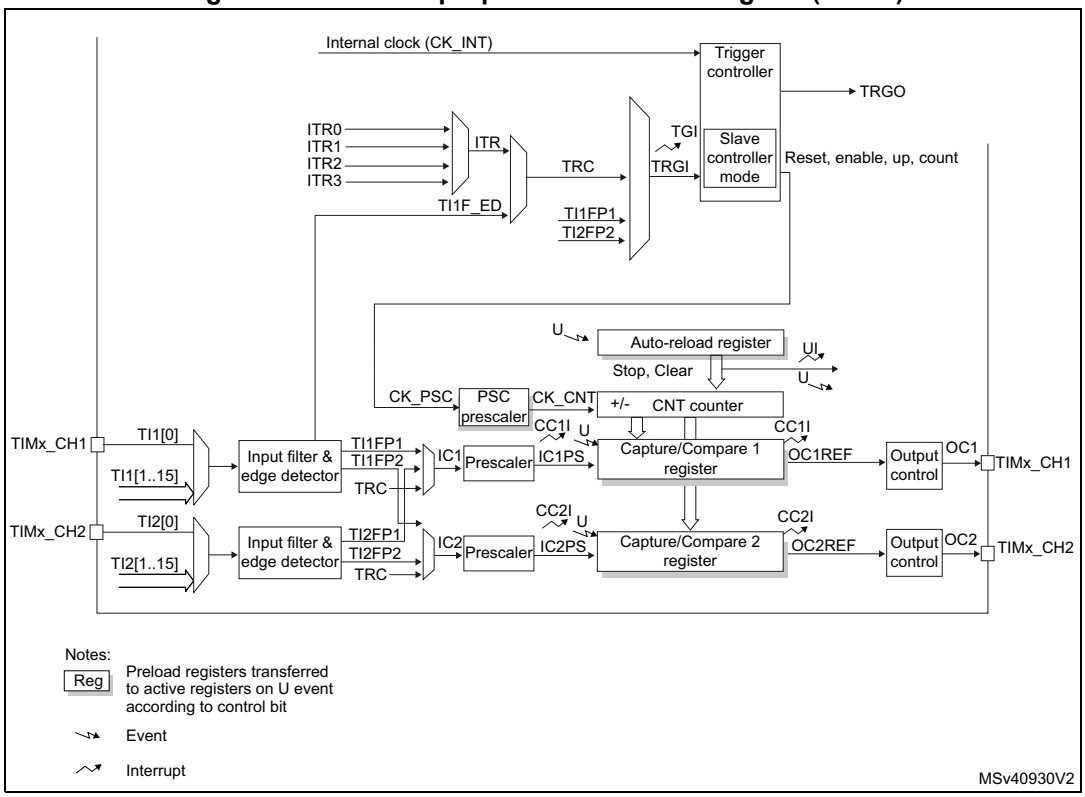
39.2 TIM12/TIM13/TIM14 main features

39.2.1 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare

Figure 452. General-purpose timer block diagram (TIM12)

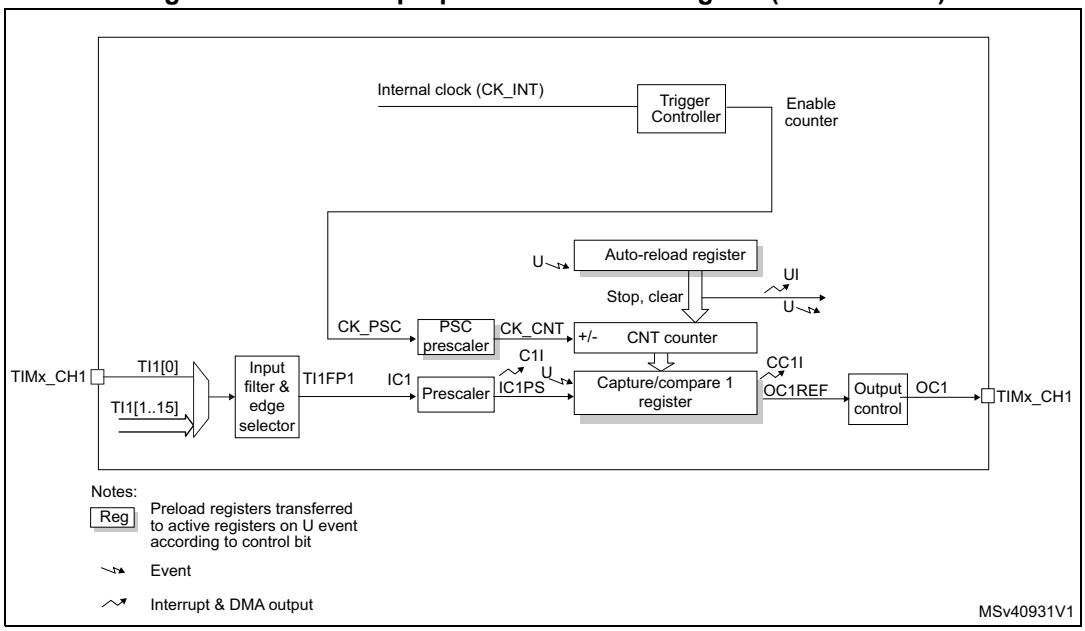


39.2.2 TIM13/TIM14 main features

The features of general-purpose timers TIM13/TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

Figure 453. General-purpose timer block diagram (TIM13/TIM14)



39.3 TIM12/TIM13/TIM14 functional description

39.3.1 Time-base unit

The main block of the timer is a 16-bit up-counter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 454 and *Figure 455* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 454. Counter timing diagram with prescaler division change from 1 to 2

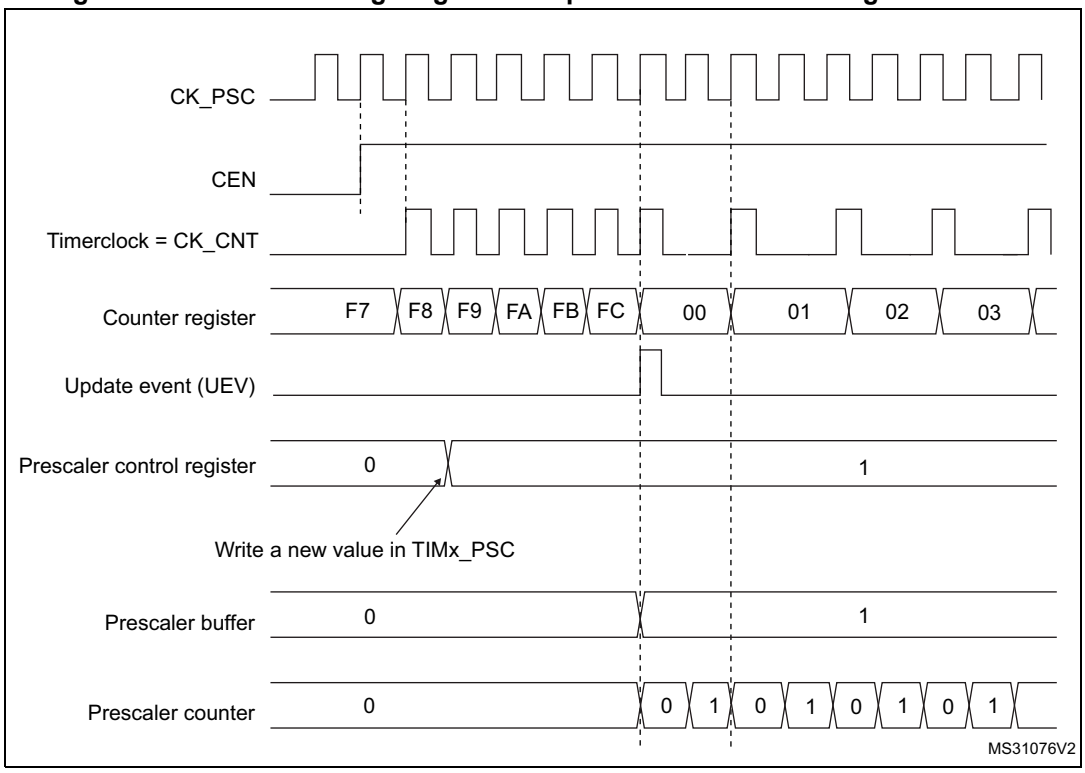
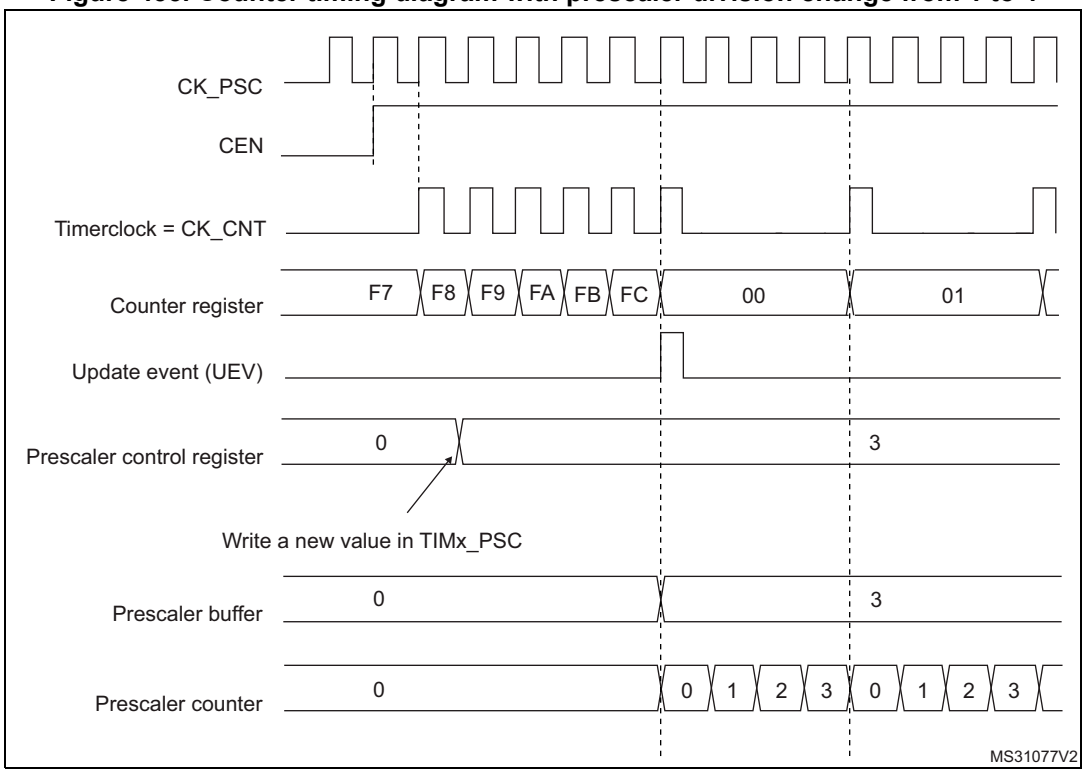


Figure 455. Counter timing diagram with prescaler division change from 1 to 4



39.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 456. Counter timing diagram, internal clock divided by 1

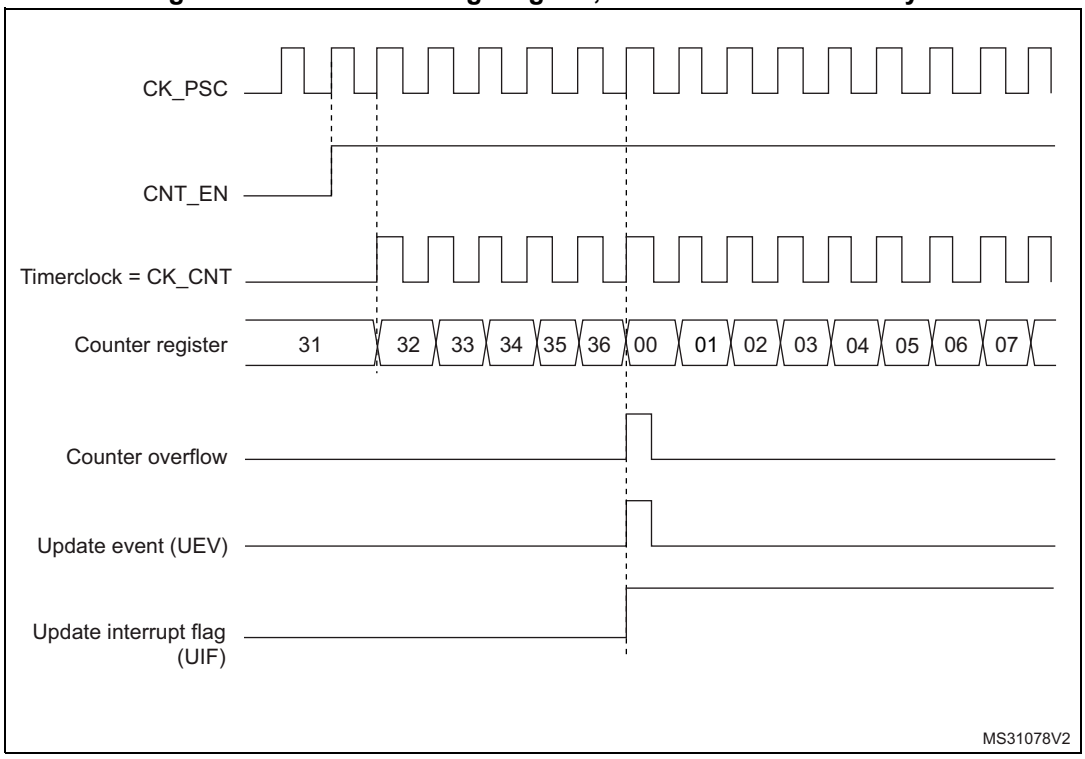


Figure 457. Counter timing diagram, internal clock divided by 2

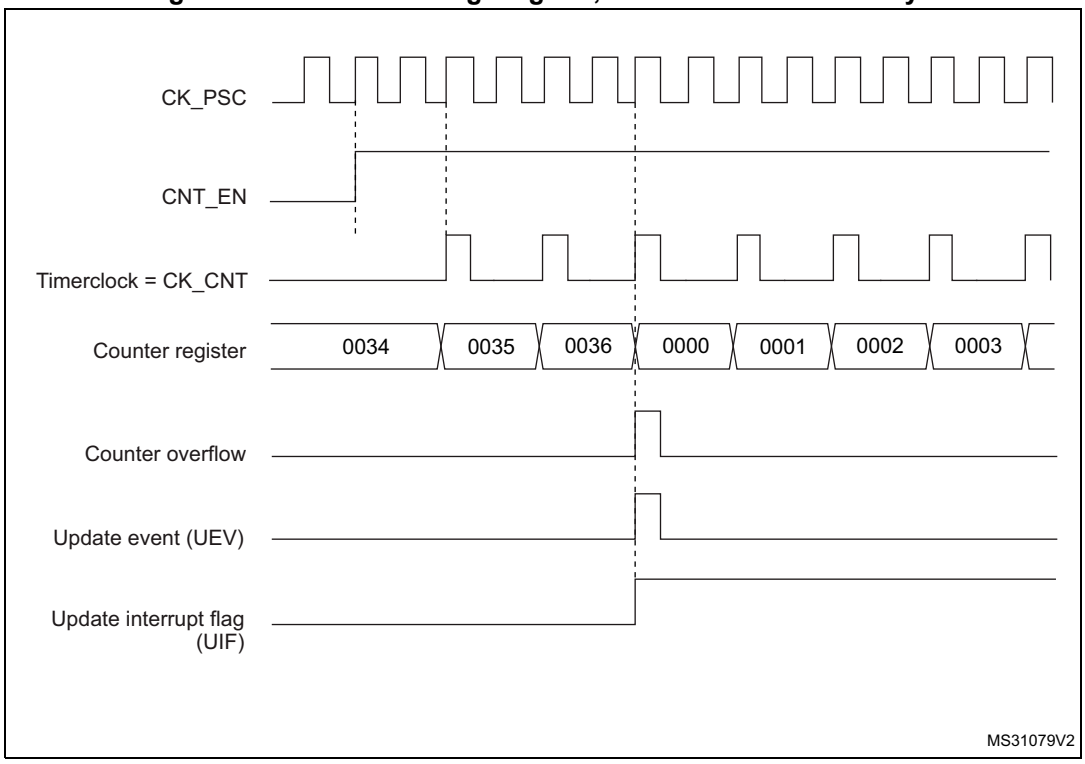


Figure 458. Counter timing diagram, internal clock divided by 4

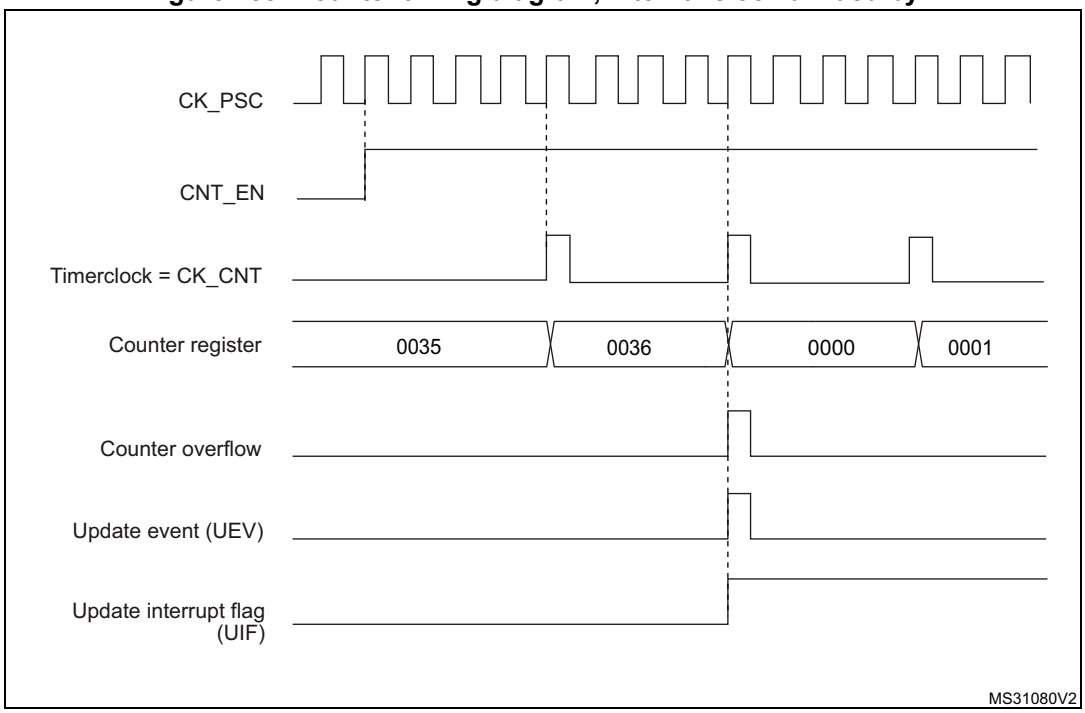


Figure 459. Counter timing diagram, internal clock divided by N

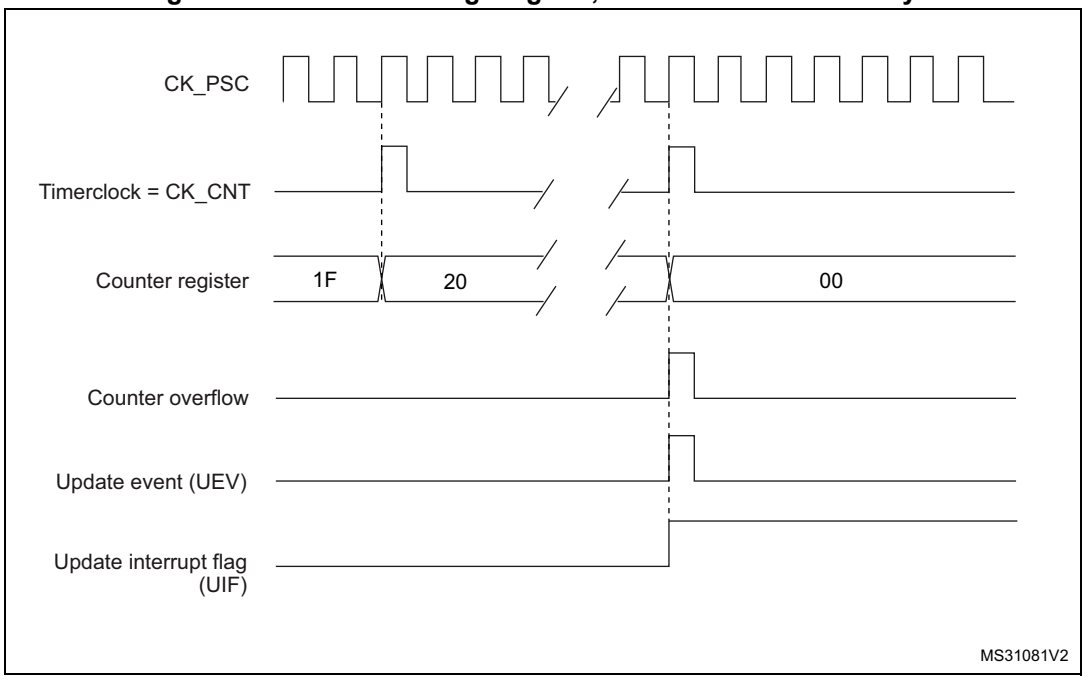


Figure 460. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

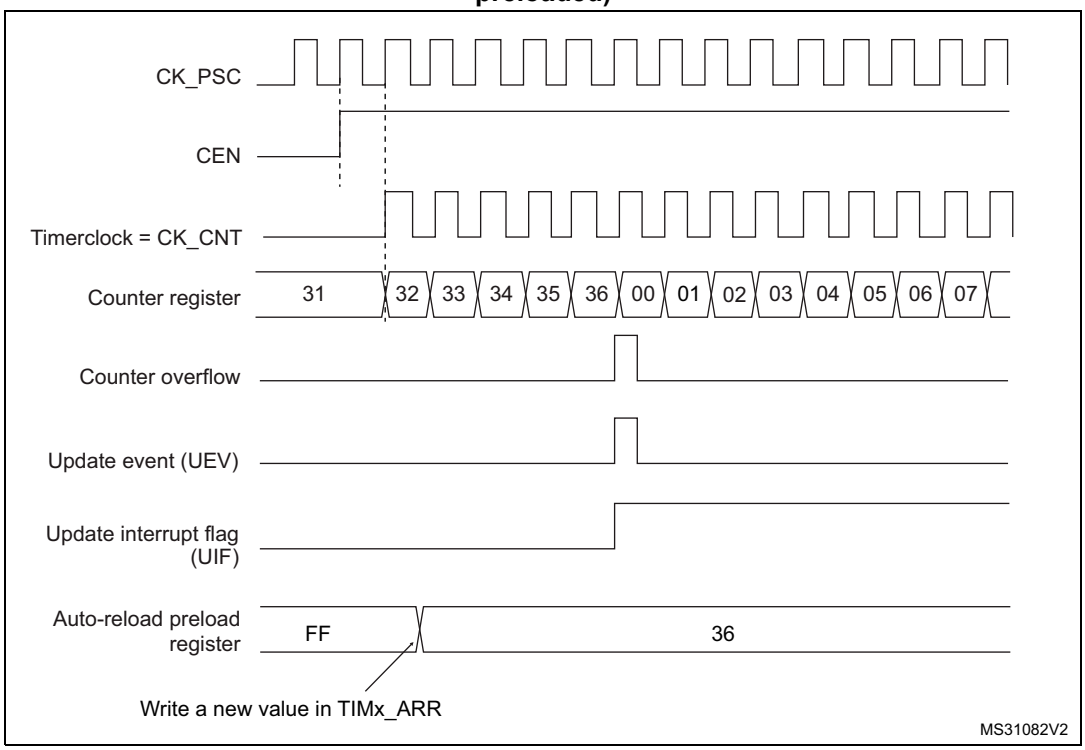
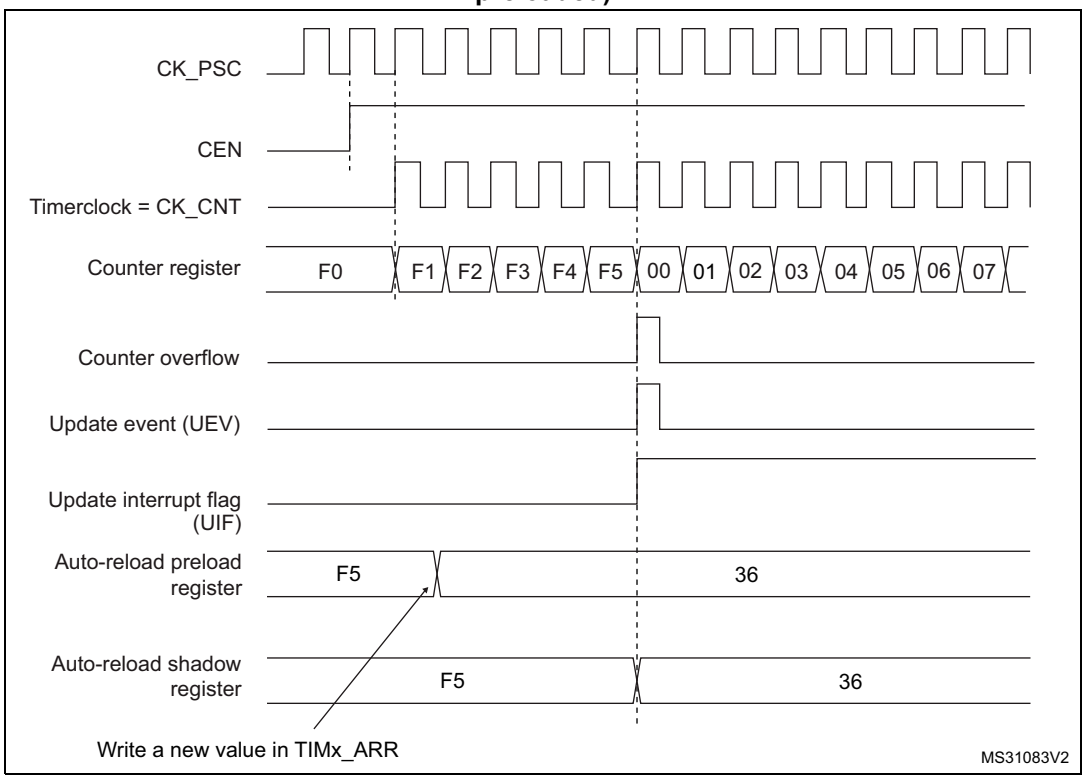


Figure 461. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



39.3.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1 (for TIM12): external input pin (TIx)
- Internal trigger inputs (ITRx) (for TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

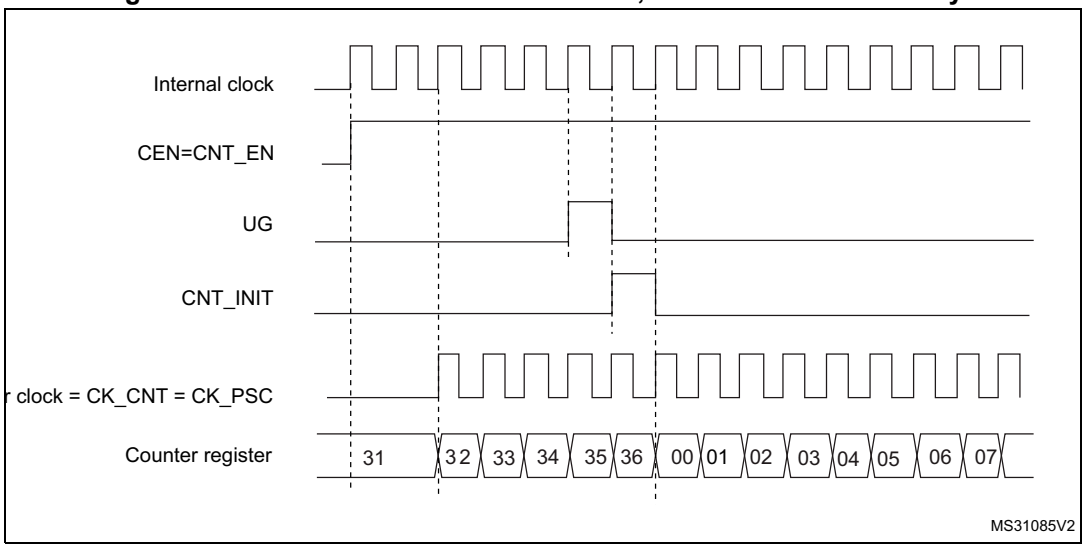
Internal clock source (CK_INT)

The internal clock source is the default clock source for TIM13/TIM14.

For TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS='000'). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 462 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

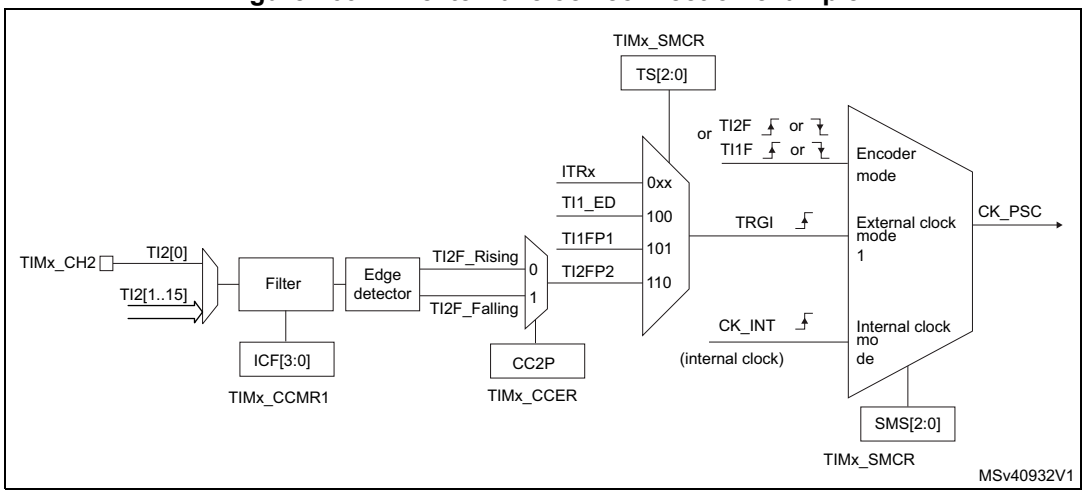
Figure 462. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1 (TIM12)

This mode is selected when SMS='111' in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 463. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

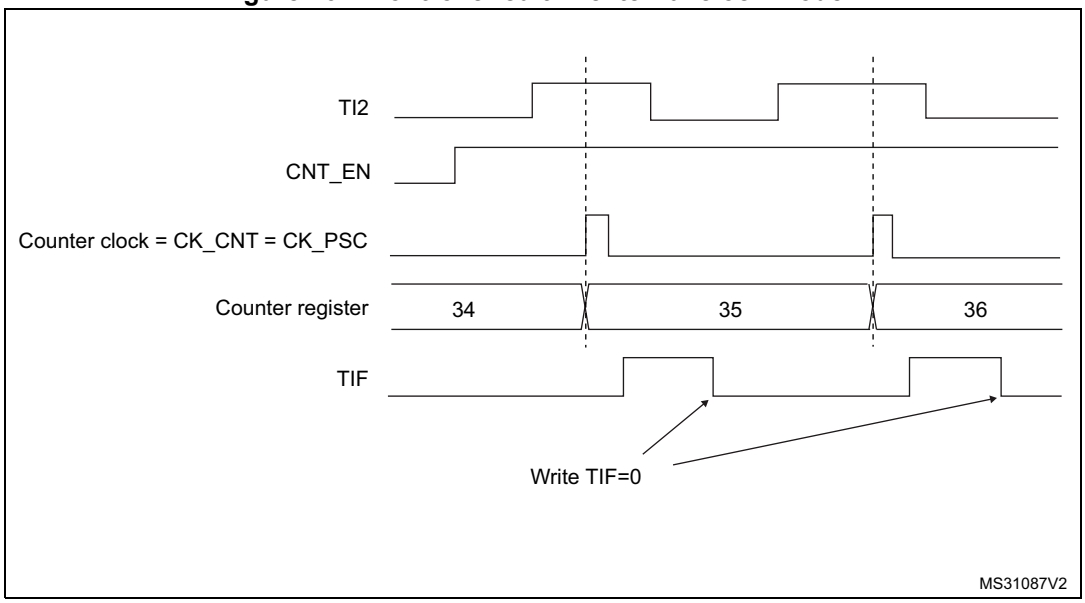
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=’0000’).
4. Select the rising edge polarity by writing CC2P=’0’ and CC2NP=’0’ in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=’111’ in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=’110’ in the TIMx_SMCR register.
7. Enable the counter by writing CEN=’1’ in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don’t need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 464. Control circuit in external clock mode 1



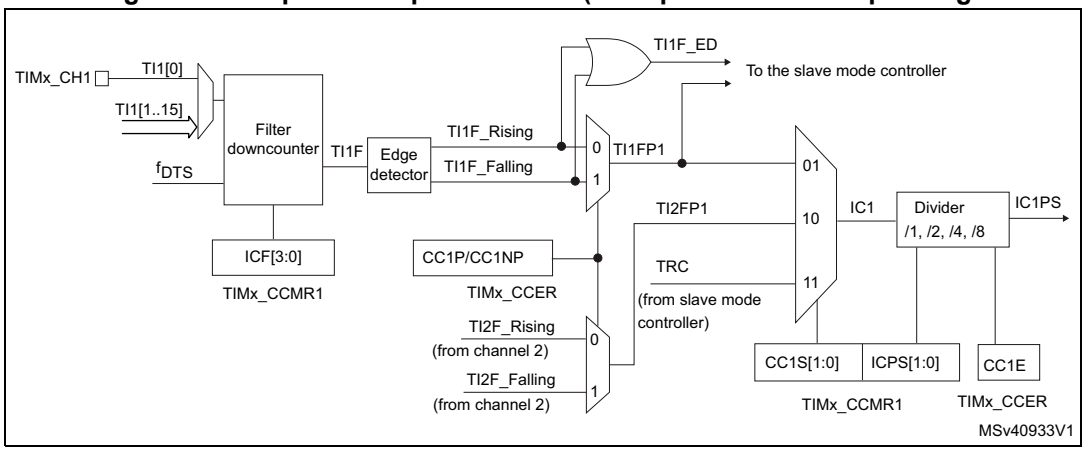
39.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 465 to *Figure 467* give an overview of one capture/compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 465. Capture/compare channel (example: channel 1 input stage



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 466. Capture/compare channel 1 main circuit

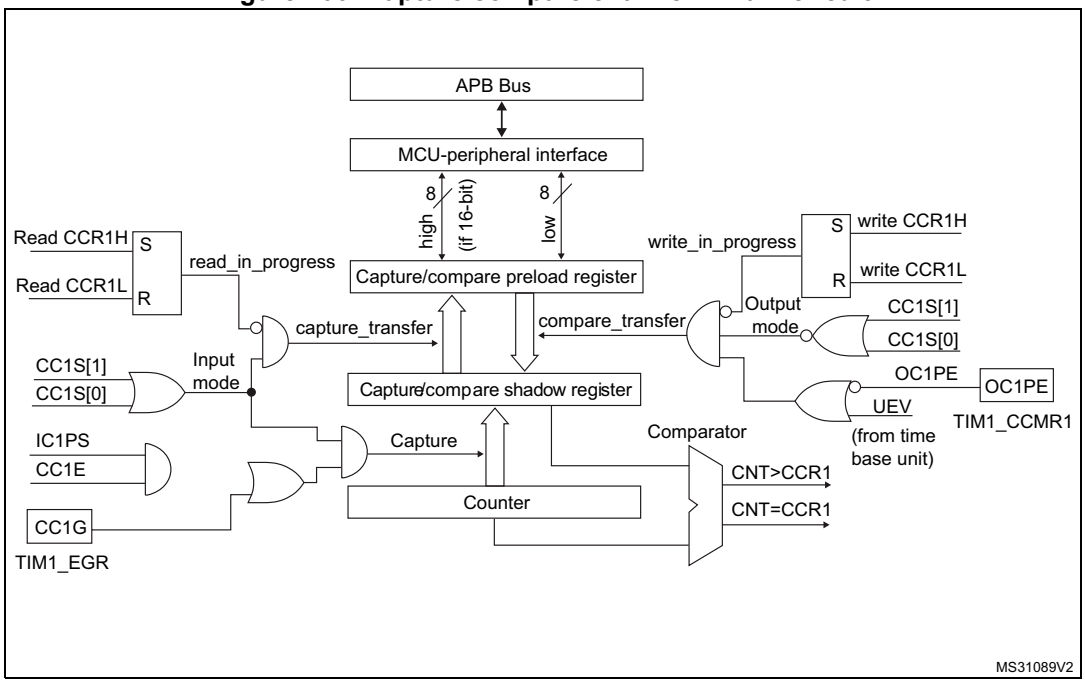
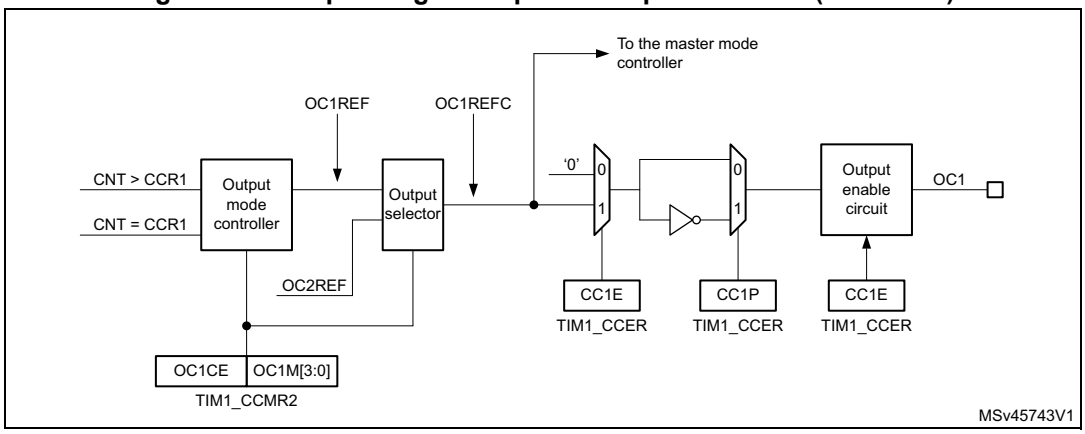


Figure 467. Output stage of capture/compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

39.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to '01' in the TIMx_CCMR1 register. As soon as CC1S becomes different from '00', the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
3. Program the input filter duration you need with respect to the signal you connect to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the

new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to '0011' in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to '00' in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

39.3.6 PWM input mode (only for TIM12)

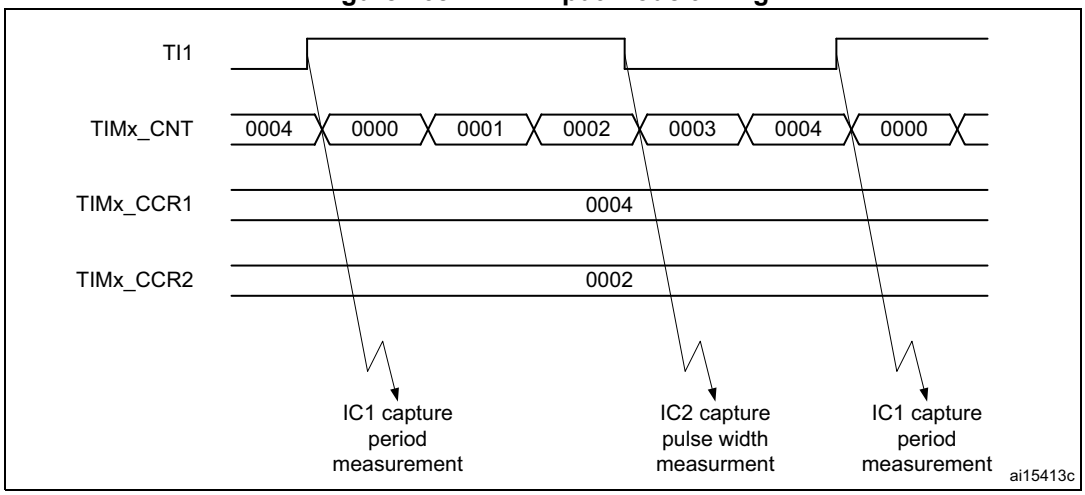
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to ‘01’ in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to ‘00’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to ‘10’ in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to ‘11’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to ‘00101’ in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to ‘100’ in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 468. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

39.3.7 Forced output mode

In output mode (CCxS bits = ‘00’ in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write ‘0101’ in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=’0’ (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to ‘0100’ in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

39.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM='0000'), be set active (OCxM='0001'), be set inactive (OCxM='0010') or can toggle (OCxM='0011') on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

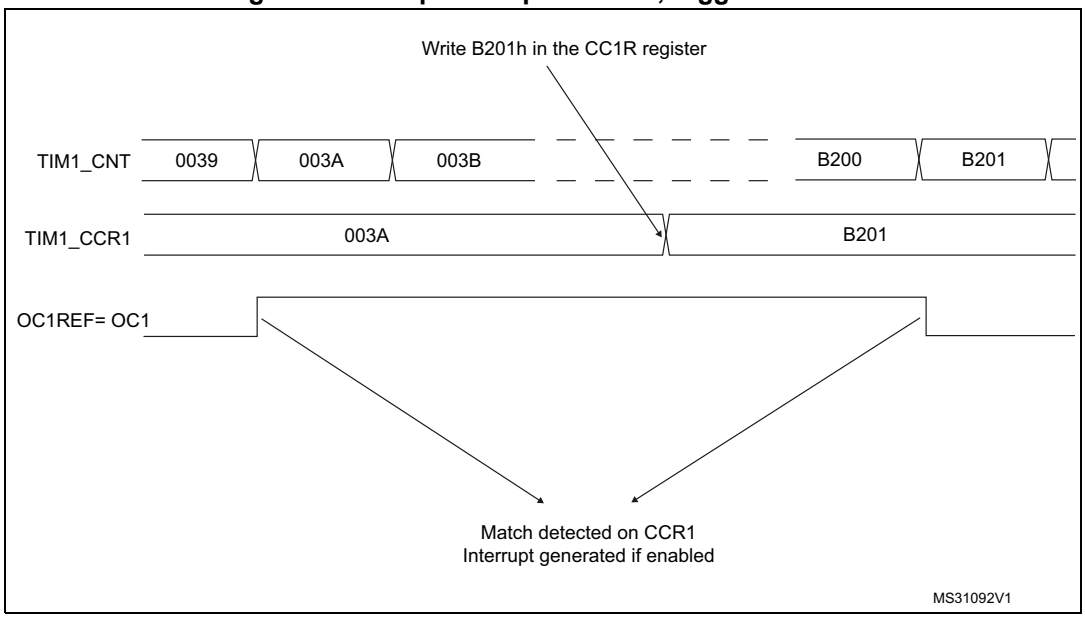
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = '0011' to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = '0' to disable preload register
   - Write CCxP = '0' to select active high polarity
   - Write CCxE = '1' to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 469*.

Figure 469. Output compare mode, toggle on OC1.



39.3.9 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

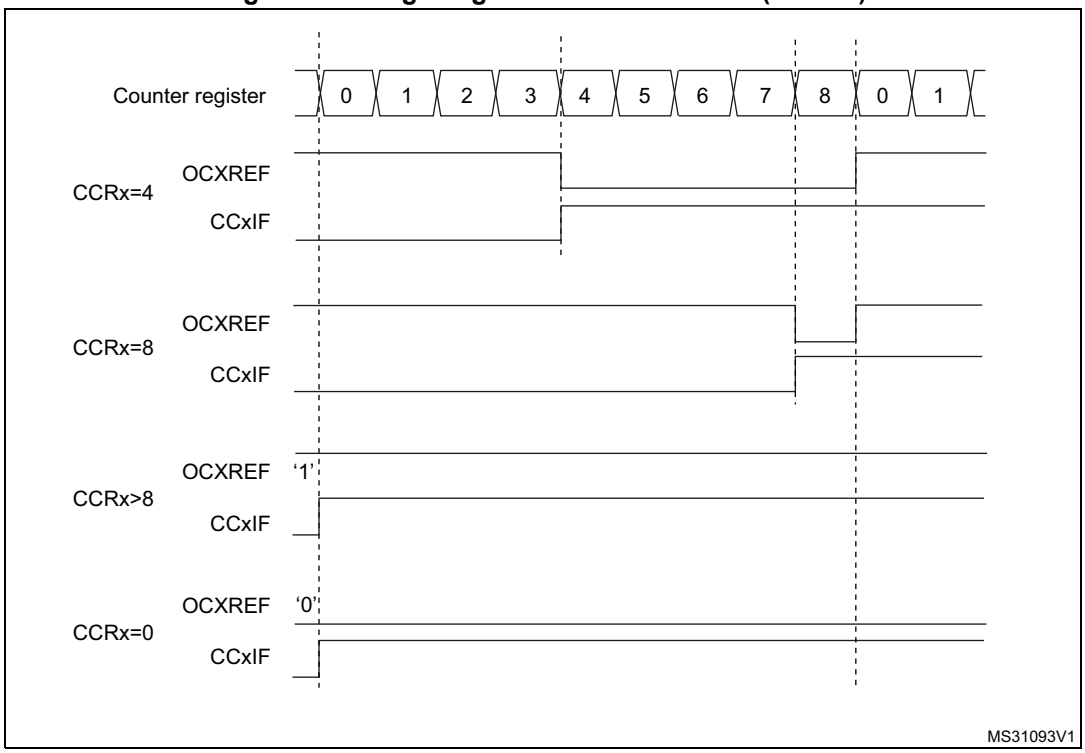
The OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 470* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 470. Edge-aligned PWM waveforms (ARR=8)



39.3.10 Combined PWM mode (TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

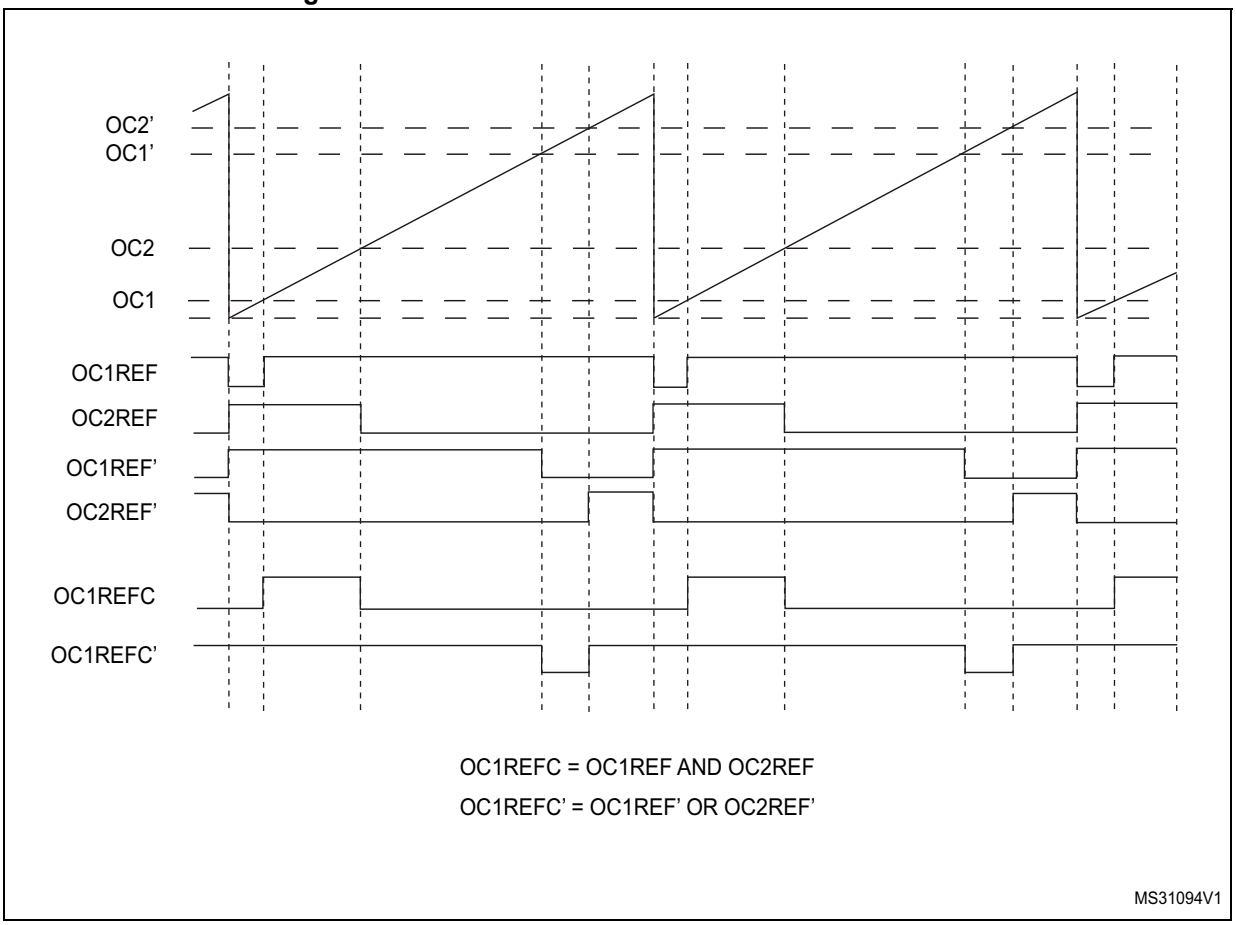
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 471 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 471. Combined PWM mode on channel 1 and 2



39.3.11 One-pulse mode

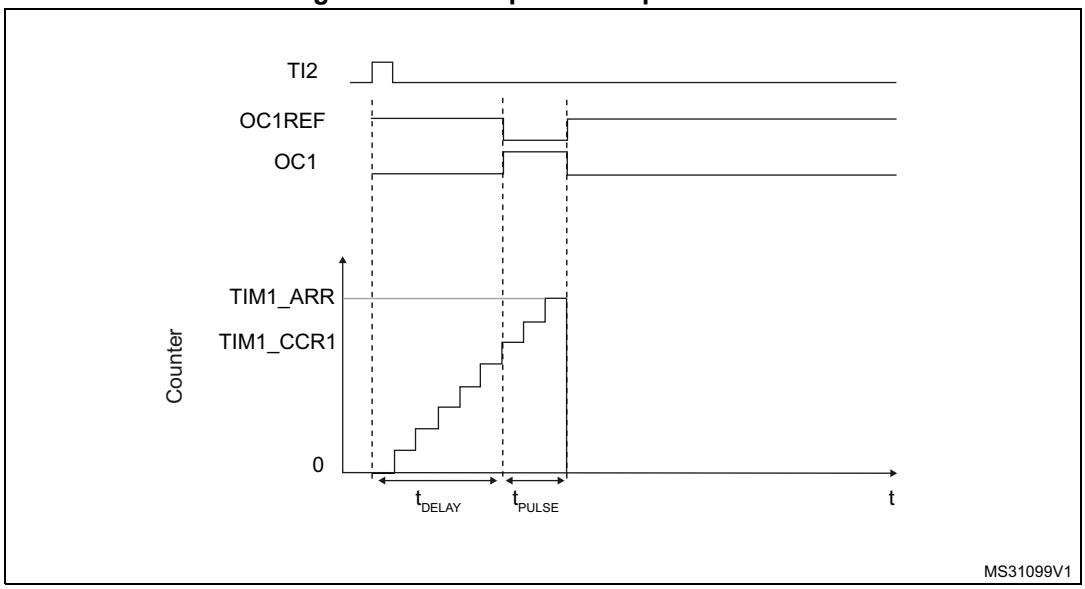
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 472. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S=’01’ in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=’0’ and CC2NP=’0’ in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=’00110’ in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say you want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=’0111’ in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

39.3.12 Retriggerable one pulse mode (OPM) (TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 39.3.11: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

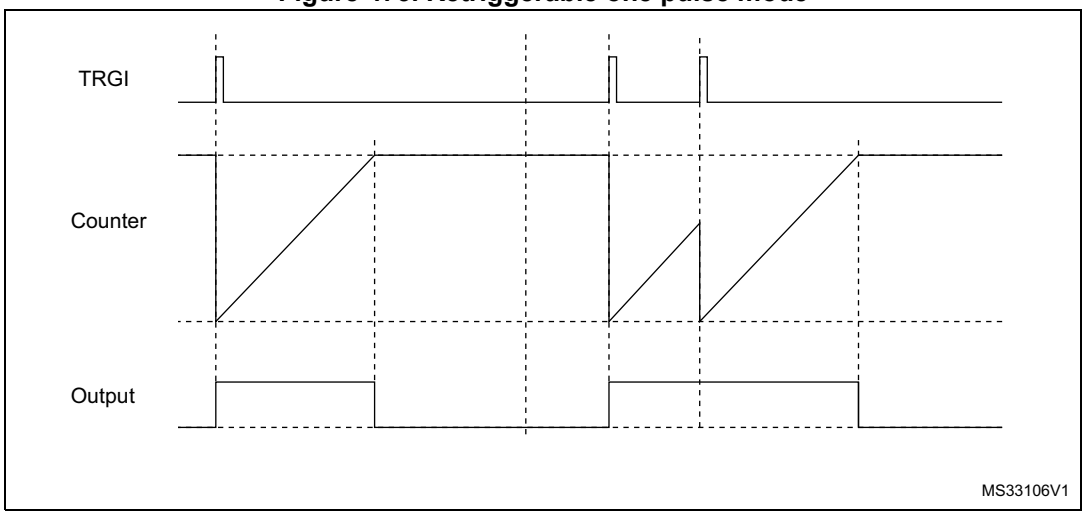
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retrigerrable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 473. Retriggerable one pulse mode



39.3.13 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

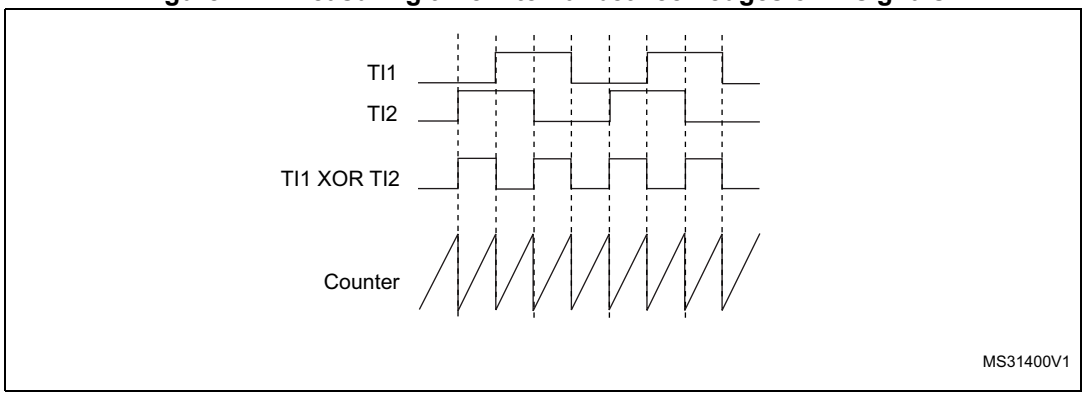
There is no latency between the assertions of the UIF and UIFCPY flags.

39.3.14 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 474*.

Figure 474. Measuring time interval between edges on 2 signals



39.3.15 TIM12 external trigger synchronization

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = '01' in the TIMx_CCMR1 register.

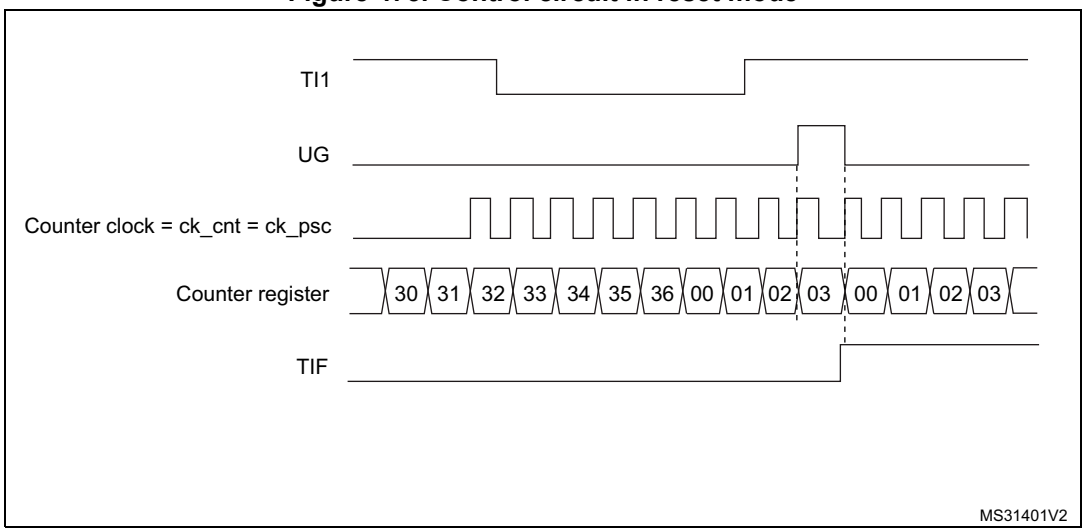
Program CC1P and CC1NP to ‘00’ in TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS=’100’ in TIMx_SMCR register. Select TI1 as the input source by writing TS=’00101’ in TIMx_SMCR register.
3. Start the counter by writing CEN=’1’ in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 475. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

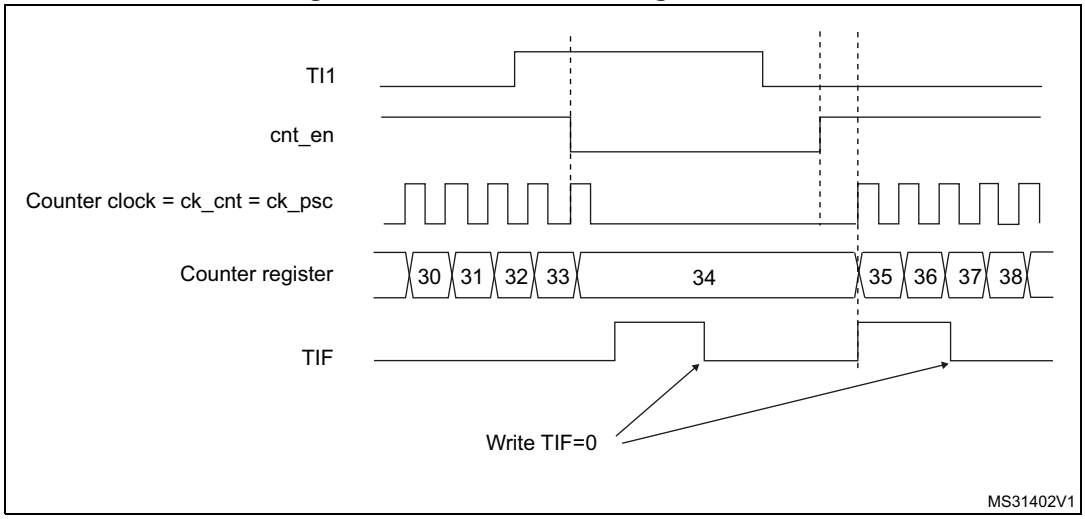
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don’t need any filter, so we keep IC1F=’0000’). The capture prescaler is not used for triggering, so you don’t need to configure it. The CC1S bits select the input capture source only, CC1S=’01’ in TIMx_CCMR1 register. Program CC1P=’1’ and CC1NP=’0’ in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=’101’ in TIMx_SMCR register. Select TI1 as the input source by writing TS=’00101’ in TIMx_SMCR register.
3. Enable the counter by writing CEN=’1’ in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=’0’, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 476. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

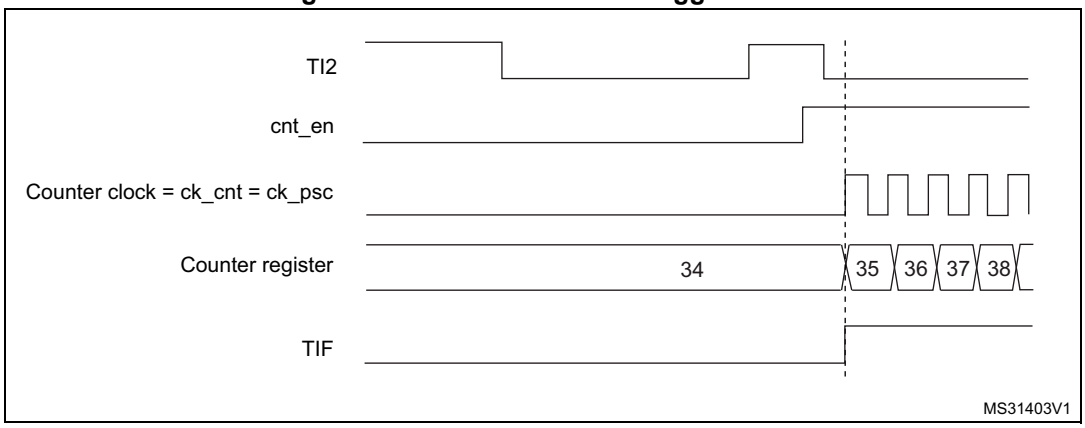
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F='0000'). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S='01' in TIMx_CCMR1 register. Program CC2P='1' and CC2NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS='110' in TIMx_SMCR register. Select TI2 as the input source by writing TS='00110' in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 477. Control circuit in trigger mode



39.3.16 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

39.3.17 Timer synchronization (TIM12)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 38.3.19: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

39.3.18 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 with FPU core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to *Section 59.5.8: Microcontroller debug unit (DBGMCU)*.

39.4 TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

39.4.1 TIM12 control register 1 (TIM12_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),
00: $t_{DTS} = t_{CK_INT}$
01: $t_{DTS} = 2 \times t_{CK_INT}$
10: $t_{DTS} = 4 \times t_{CK_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event

1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt if enabled. These events can be:

– Counter overflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update event (UEV) generation.

0: UEV enabled. An UEV is generated by one of the following events:

– Counter overflow

– Setting the UG bit

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

39.4.2 TIM12 control register 2 (TIM12_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: TI1 selection
0: The TIM12_CH1 pin is connected to TI1 input
1: The TIM12_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: Compare - OC1REF signal is used as trigger output (TRGO).
101: Compare - OC2REF signal is used as trigger output (TRGO).

Bits 3:0 Reserved, must be kept at reset value.

39.4.3 TIM12 slave mode control register (TIM12_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **SMS[3]:** Slave mode selection - bit 3
Refer to SMS description - bits 2:0

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode
0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.
00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
Others: Reserved
See *Table 330: TIMx internal trigger connection on page 1722* for more details on the meaning of ITRx for each timer.

Note: These bits must be changed only when they are not used (e.g. when SMS='000') to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
Other codes: reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='00100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Table 330. TIMx internal trigger connection

Slave TIM	ITR0 (TS = '00000')	ITR1 (TS = '00001')	ITR2 (TS = '00010')	ITR3 (TS = '00011')
TIM12	TIM4	TIM5	TIM13 OC1	TIM14 OC1

39.4.4 TIM12 Interrupt enable register (TIM12_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

39.4.5 TIM12 status register (TIM12_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow and if UDIS='0' in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS='0' and UDIS='0' in the TIMx_CR1 register.

39.4.6 TIM12 event generation register (TIM12_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

39.4.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes. For a given bit, OCxx describes its function when the channel is configured in output mode, ICxx describes its function when the channel is configured in input mode. So you must take care that the same bit can have different meanings for the input stage and the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							Res.								Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, always read as 0

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3
Refer to OC2M description on bits 14:12

Bits 23:17 Reserved, always read as 0

Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3
Refer to OC1M description on bits 6:4

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **OC2M[2:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])
These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas the active level of OC1 depends on the CC1P.
0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
0001: Set channel 1 to active level on match. The OC1REF signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. The OC1REF signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1
0100: Force inactive level - OC1REF is forced low
0101: Force active level - OC1REF is forced high
0110: PWM mode 1 - channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive
0111: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active
1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.
1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.
1010: Reserved,
1011: Reserved,
1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.
1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.
1110: Reserved,
1111: Reserved
Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in the TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bitfield defines the frequency used to sample the TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

39.4.8 TIM12 capture/compare enable register (TIM12_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
CC1 channel configured as output:
0: OC1 active high.
1: OC1 active low.
CC1 channel configured as input:
The CC1P and CC1NP bits select TI1FP1 polarity for capture operations.
00: non-inverted/rising edge
Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
CC1 channel configured as output:
0: Off - OC1 is not active.
1: On - OC1 signal is output on the corresponding output pin.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Table 331. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (OCx='0', OCx_EN='0')
1	OCx=OCxREF + Polarity, OCx_EN='1'

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and on the GPIO registers.

39.4.9 TIM12 counter (TIM12_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

39.4.10 TIM12 prescaler (TIM12_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

39.4.11 TIM12 auto-reload register (TIM12_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to the *Section 39.3.1: Time-base unit on page 1697* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

39.4.12 TIM12 capture/compare register 1 (TIM12_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded into the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit). Else the preload value is copied into the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signaled on the OC1 output.

If channel CC1is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

39.4.13 TIM12 capture/compare register 2 (TIM12_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded into the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit). Else the preload value is copied into the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signalled on the OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

39.4.14 TIM12 timer input selection register (TIM12_TISEL)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw								rw			

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM12_CH2 input

Other: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM12_CH1 input

Other: Reserved

39.4.15 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 332. TIM12 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIM12_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3] Res.		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x24	**TIMx_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 332. TIM12 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved	Reserved																															
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x64	Reserved	Res.																															
0x68	**TIM12_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

39.5 TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

39.5.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),
00: $t_{DTS} = t_{CK_INT}$
01: $t_{DTS} = 2 \times t_{CK_INT}$
10: $t_{DTS} = 4 \times t_{CK_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

- Counter overflow
- Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

39.5.2 TIMx Interrupt enable register (TIMx_DIER)(x = 13, 14)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

39.5.3 TIMx status register (TIMx_SR)(x = 13, 14)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow and if UDIS=’0’ in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=’0’ and UDIS=’0’ in the TIMx_CR1 register.

39.5.4 TIMx event generation register (TIMx_EGR)(x = 13, 14)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

39.5.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13, 14)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															Res.
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:17 Reserved, always read as 0

Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3
Refer to OC1M description on bits 6:4

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])
These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.
0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.
0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.
0100: Force inactive level - OC1REF is forced low.
0101: Force active level - OC1REF is forced high.
0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.
0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active
Others: Reserved
Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Bit 3 **OC1PE**: Output compare 1 preload enable
0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.
Note: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable
This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register).
Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS}1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
0001: $f_{SAMPLING}=f_{CK_INT}$, N=21001: $f_{SAMPLING}=f_{DTS}/8$, N=8
0010: $f_{SAMPLING}=f_{CK_INT}$, N=41010: $f_{SAMPLING}=f_{DTS}/16$, N=5
0011: $f_{SAMPLING}=f_{CK_INT}$, N=81011: $f_{SAMPLING}=f_{DTS}/16$, N=6
0100: $f_{SAMPLING}=f_{DTS}/2$, N=61100: $f_{SAMPLING}=f_{DTS}/16$, N=8
0101: $f_{SAMPLING}=f_{DTS}/2$, N=81101: $f_{SAMPLING}=f_{DTS}/32$, N=5
0110: $f_{SAMPLING}=f_{DTS}/4$, N=61110: $f_{SAMPLING}=f_{DTS}/32$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=81111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

39.5.6 TIMx capture/compare enable register (TIMx_CCER)(x = 13, 14)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.
CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
The CC1P and CC1NP bits select TI1FP1 polarity for capture operations.
00: noninverted/rising edge
Circuit is sensitive to TI1FP1 rising edge (capture mode), TI1FP1 is not inverted.
01: inverted/falling edge
Circuit is sensitive to TI1FP1 falling edge (capture mode), TI1FP1 is inverted.
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TI1FP1 rising and falling edges (capture mode), TI1FP1 is not inverted.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

CC1 channel configured as output:
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 333. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx='0', OCx_EN='0')
1	OCx=OCxREF + Polarity, OCx_EN='1'

Note: The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO registers.

39.5.7 TIMx counter (TIMx_CNT)(x = 13, 14)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

39.5.8 TIMx prescaler (TIMx_PSC)(x = 13, 14)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

39.5.9 TIMx auto-reload register (TIMx_ARR)(x = 13, 14)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to *Section 39.3.1: Time-base unit on page 1697* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

39.5.10 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13, 14)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1).

39.5.11 TIM13 timer input selection register (TIM13_TISEL)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw			

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM13_CH1 input
Other: Reserved

39.5.12 TIM14 timer input selection register (TIM14_TISEL)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw			

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM14_CH1 input
Other: Reserved

39.5.13 TIM13/TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 334. TIM13/TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04 to 0x08	Reserved	Res.																															
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
	Reset value																															0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0								0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																															0	0
0x18	**TIMx_CCMR1** Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
	Reset value																													0		0	0
0x24	**TIMx_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved	Res.																															

Table 334. TIM13/TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x64	Reserved	Res.																															
0x68	**TIM13_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x68	**TIM14_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

40 General-purpose timers (TIM15/TIM16/TIM17)

40.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 40.4.21: Timer synchronization (TIM15)*.

40.2 TIM15 main features

TIM15 includes the following features:

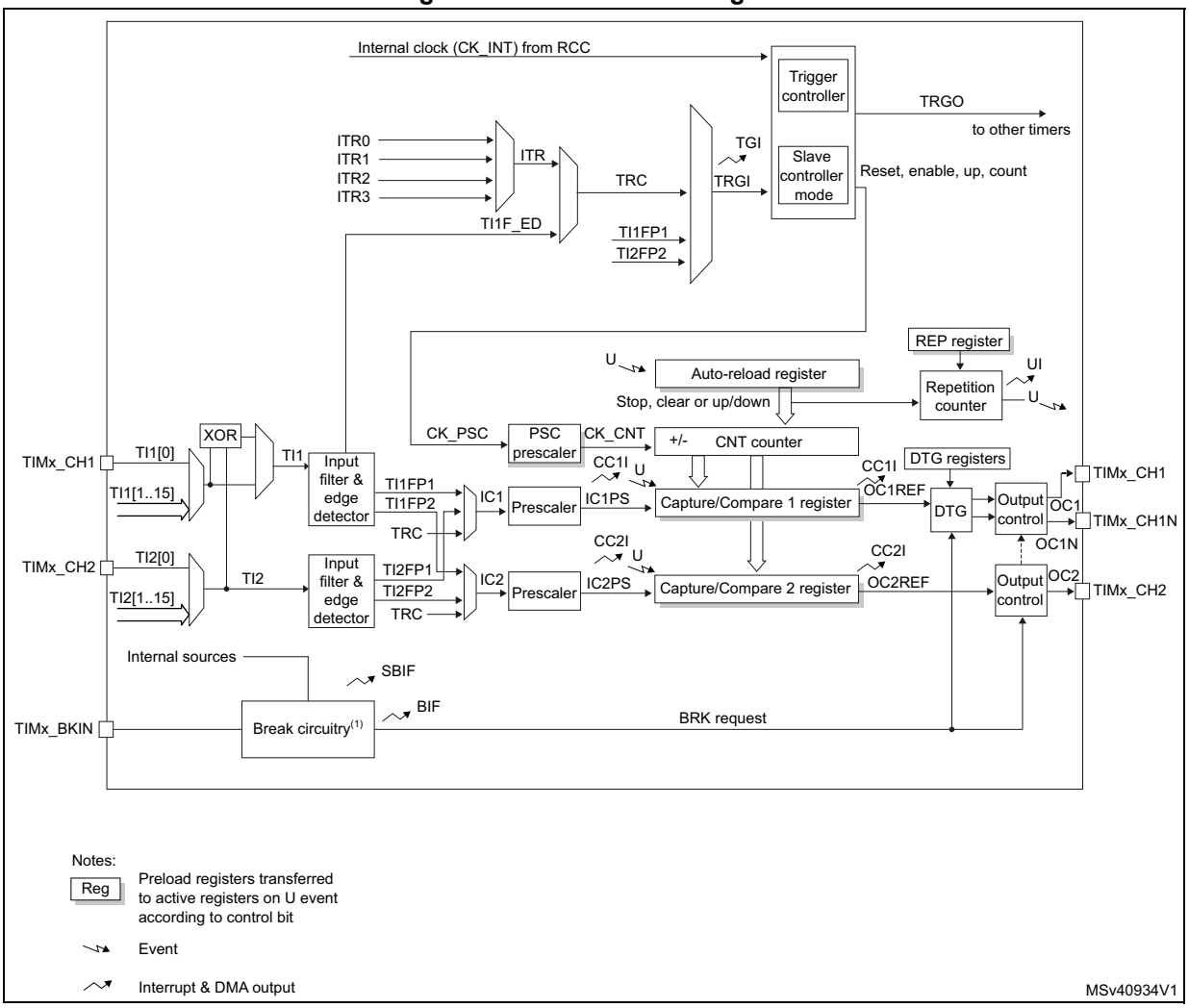
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

40.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

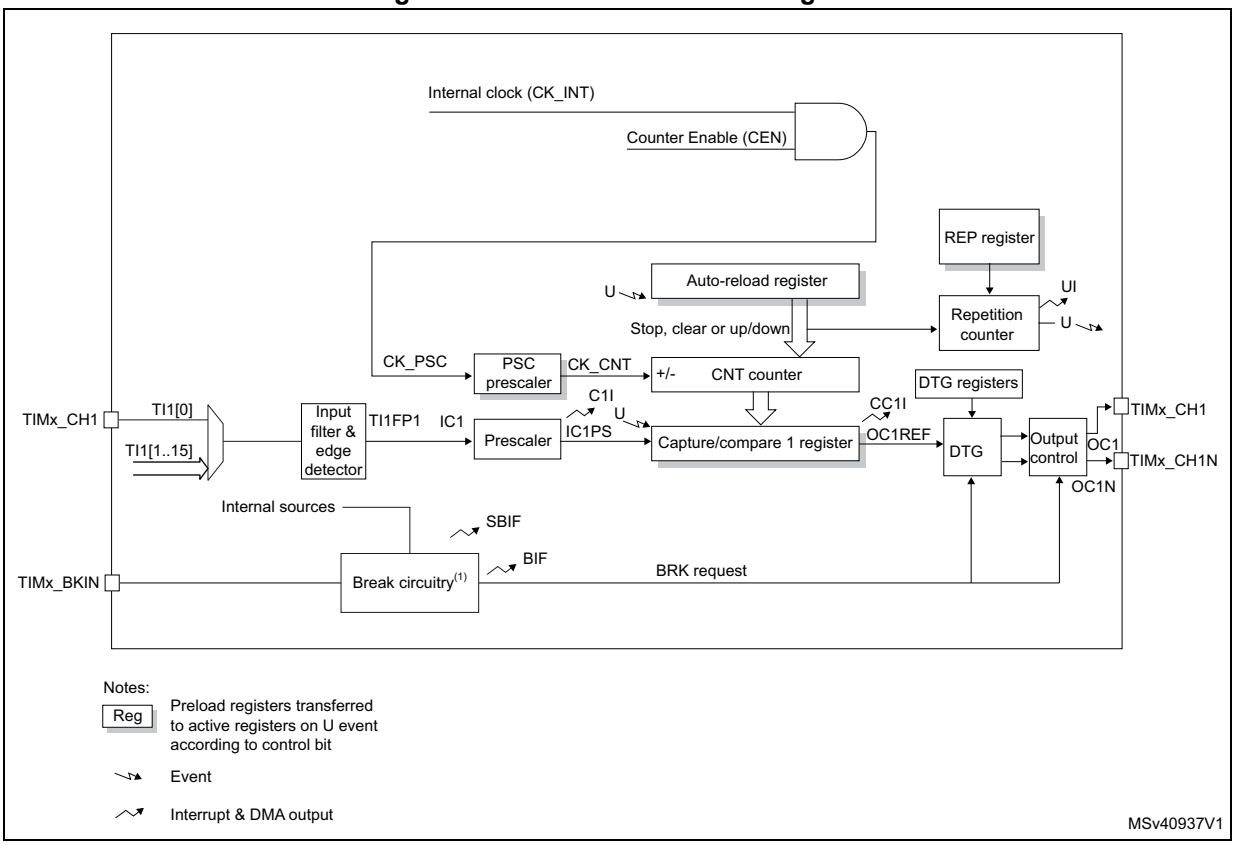
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 478. TIM15 block diagram



1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 7.5.3: Clock Security System (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M7 with FPU LOCKUP (Hardfault) output
   - COMP output

Figure 479. TIM16/TIM17 block diagram



1. The internal break event source can be:
   - A clock failure event generated by CSS. For further information on the CSS, refer to *Section 7.5.3: Clock Security System (CSS)*
   - A PVD output
   - SRAM parity error signal
   - Cortex®-M7 with FPU LOCKUP (Hardfault) output
   - COMP output

40.4 TIM15/TIM16/TIM17 functional description

40.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 480 and *Figure 481* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 480. Counter timing diagram with prescaler division change from 1 to 2

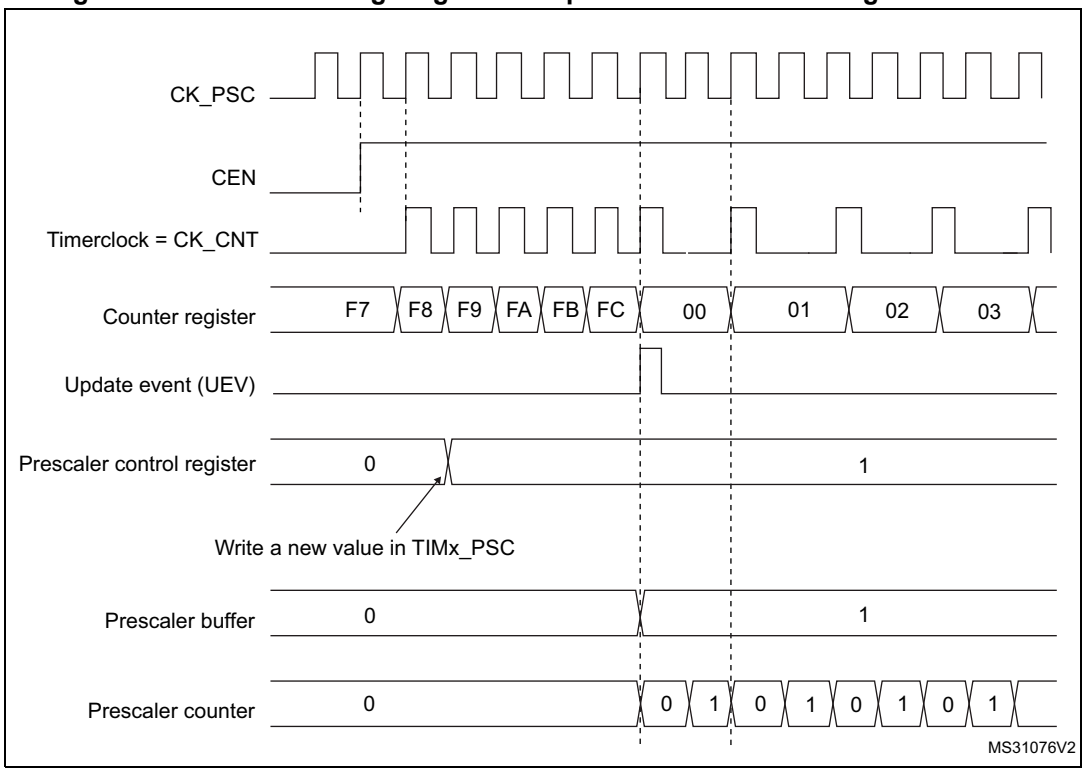
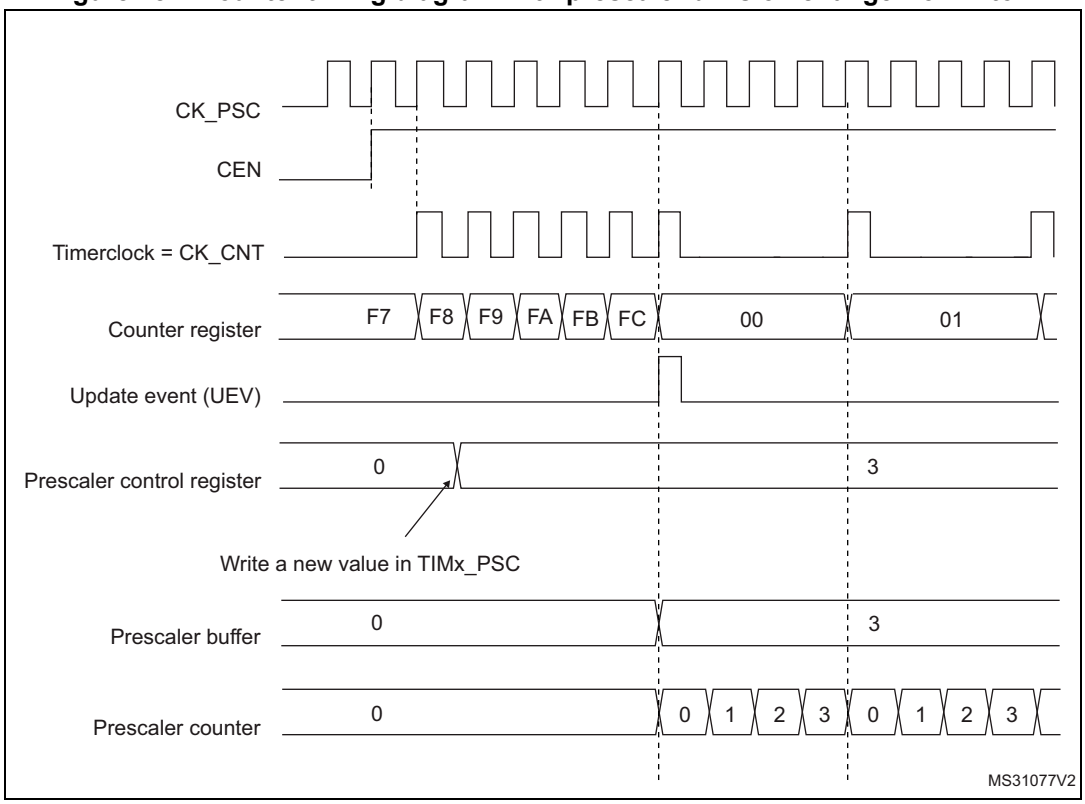


Figure 481. Counter timing diagram with prescaler division change from 1 to 4



40.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 482. Counter timing diagram, internal clock divided by 1

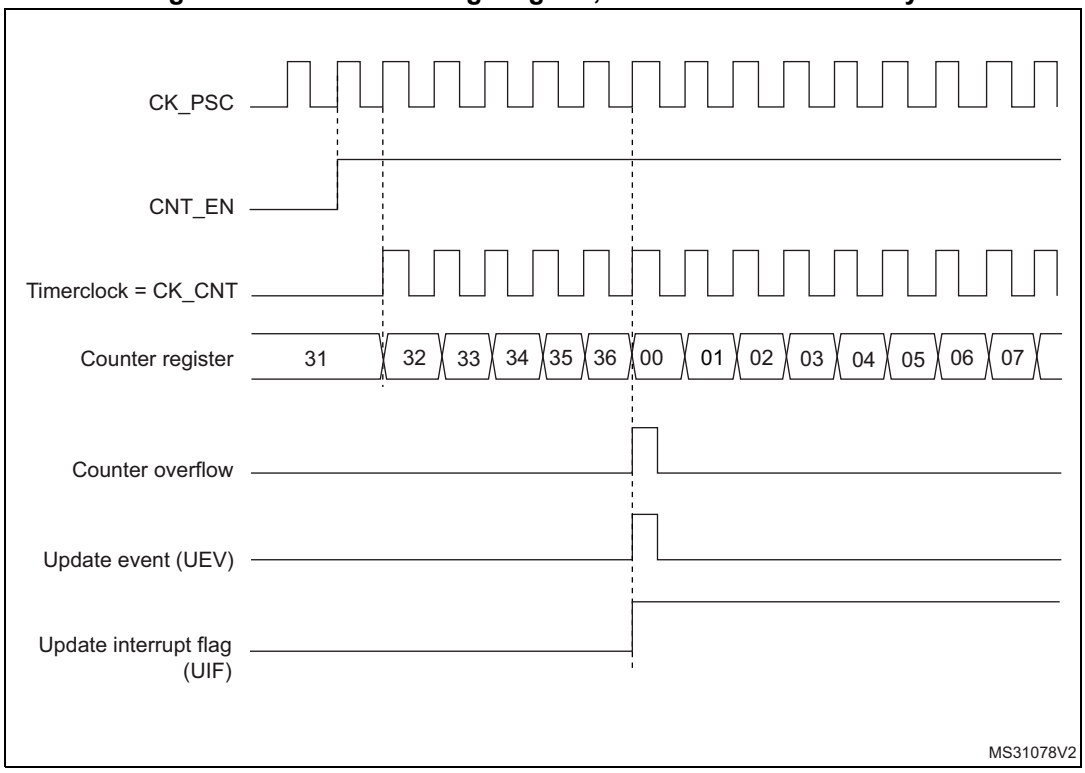


Figure 483. Counter timing diagram, internal clock divided by 2

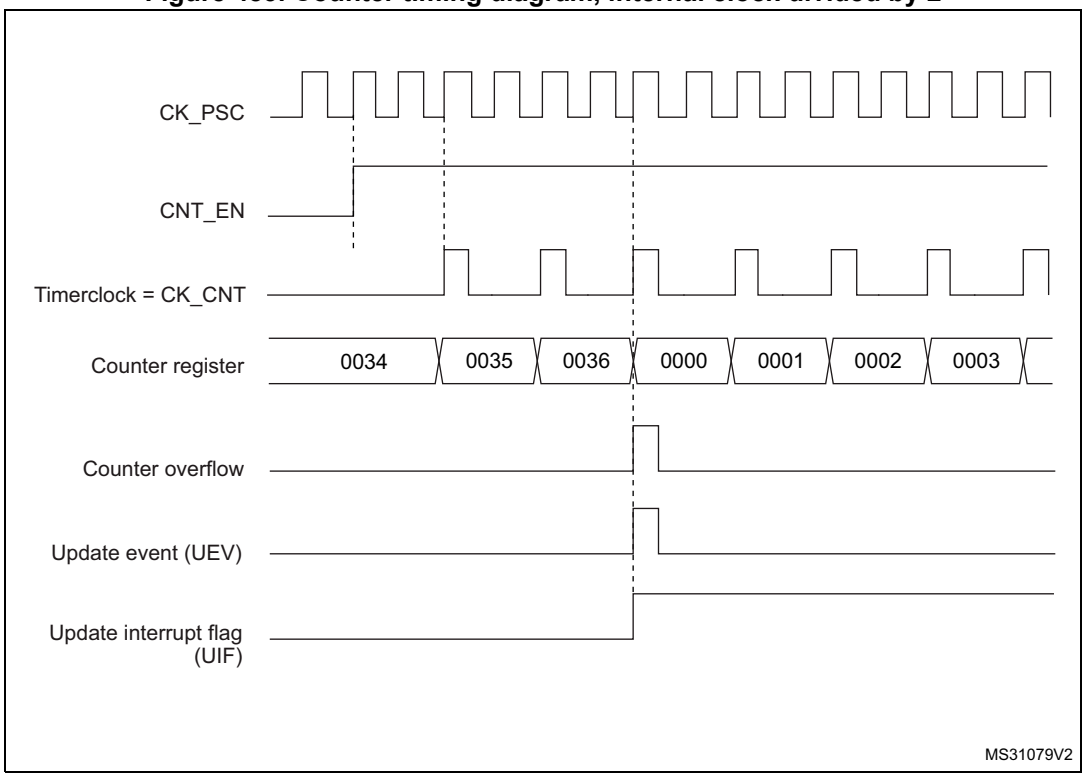


Figure 484. Counter timing diagram, internal clock divided by 4

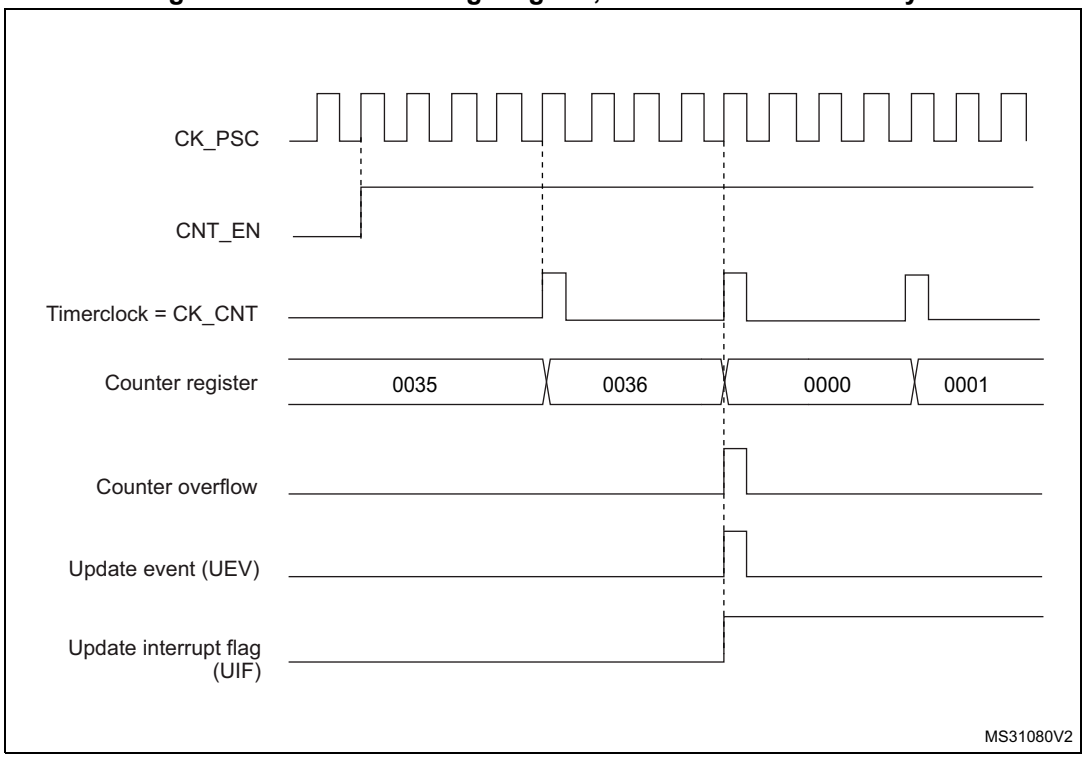


Figure 485. Counter timing diagram, internal clock divided by N

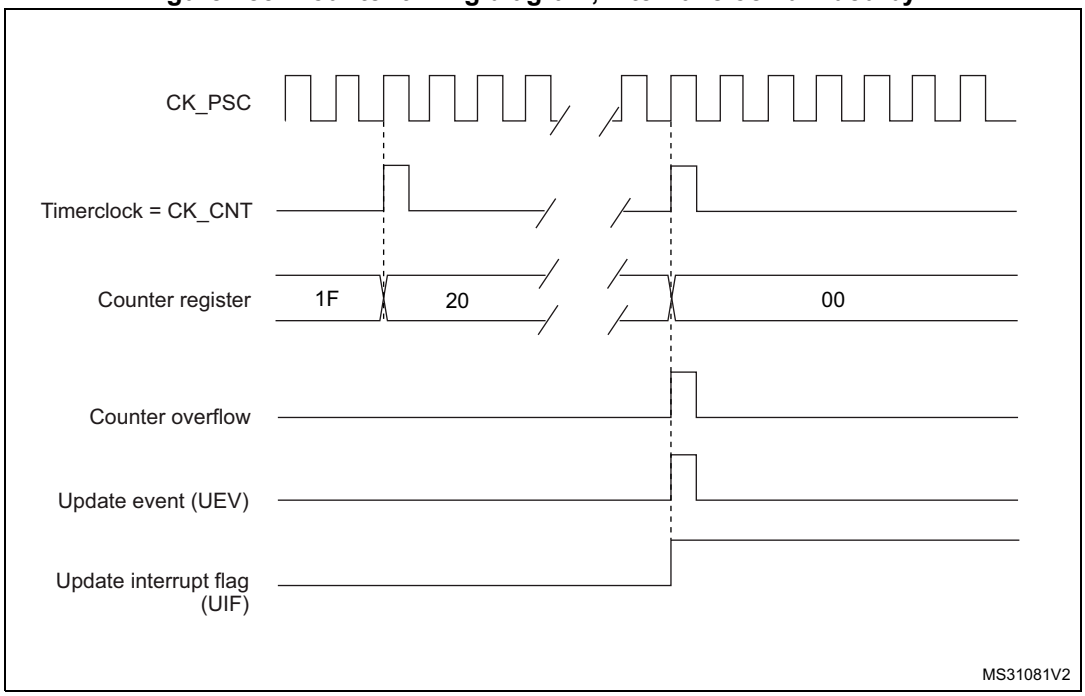


Figure 486. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

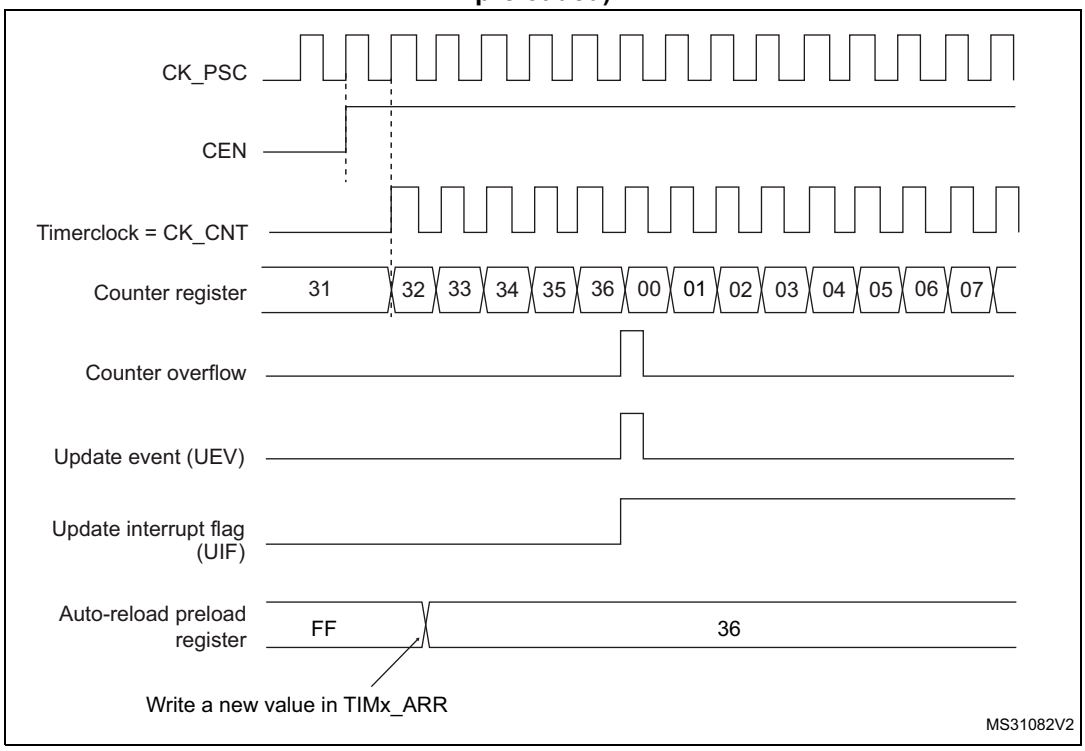
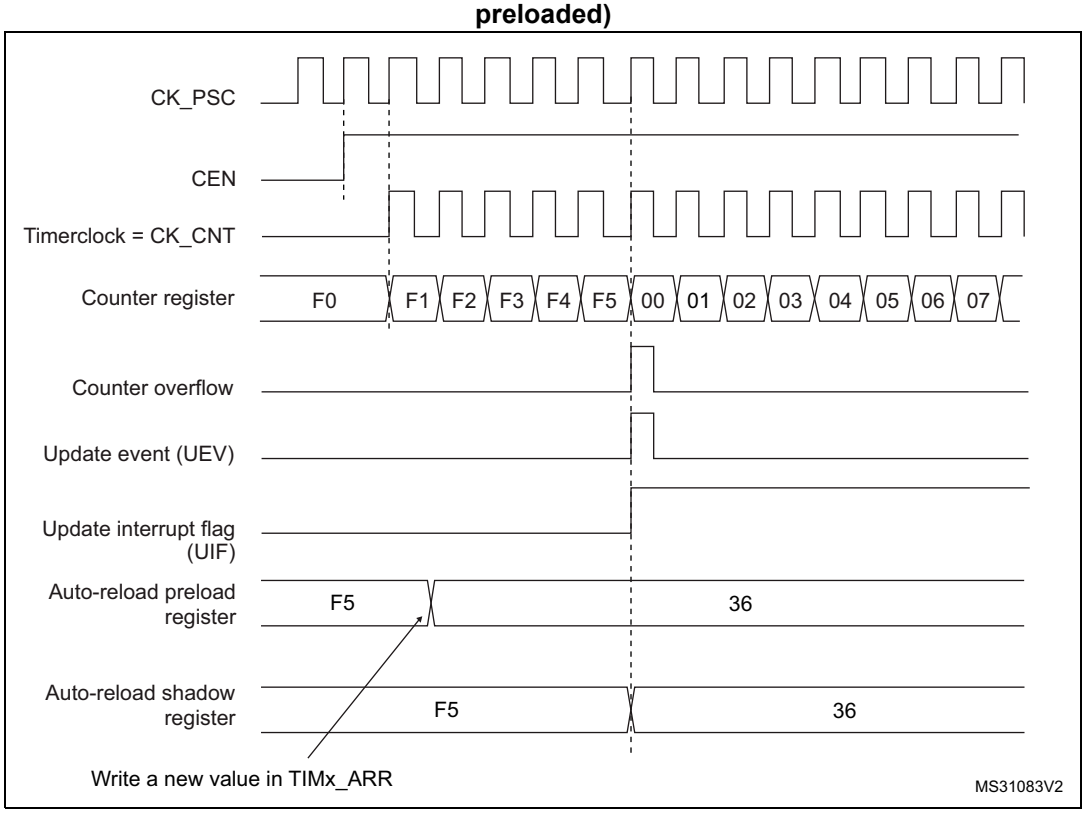


Figure 487. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



40.4.3 Repetition counter

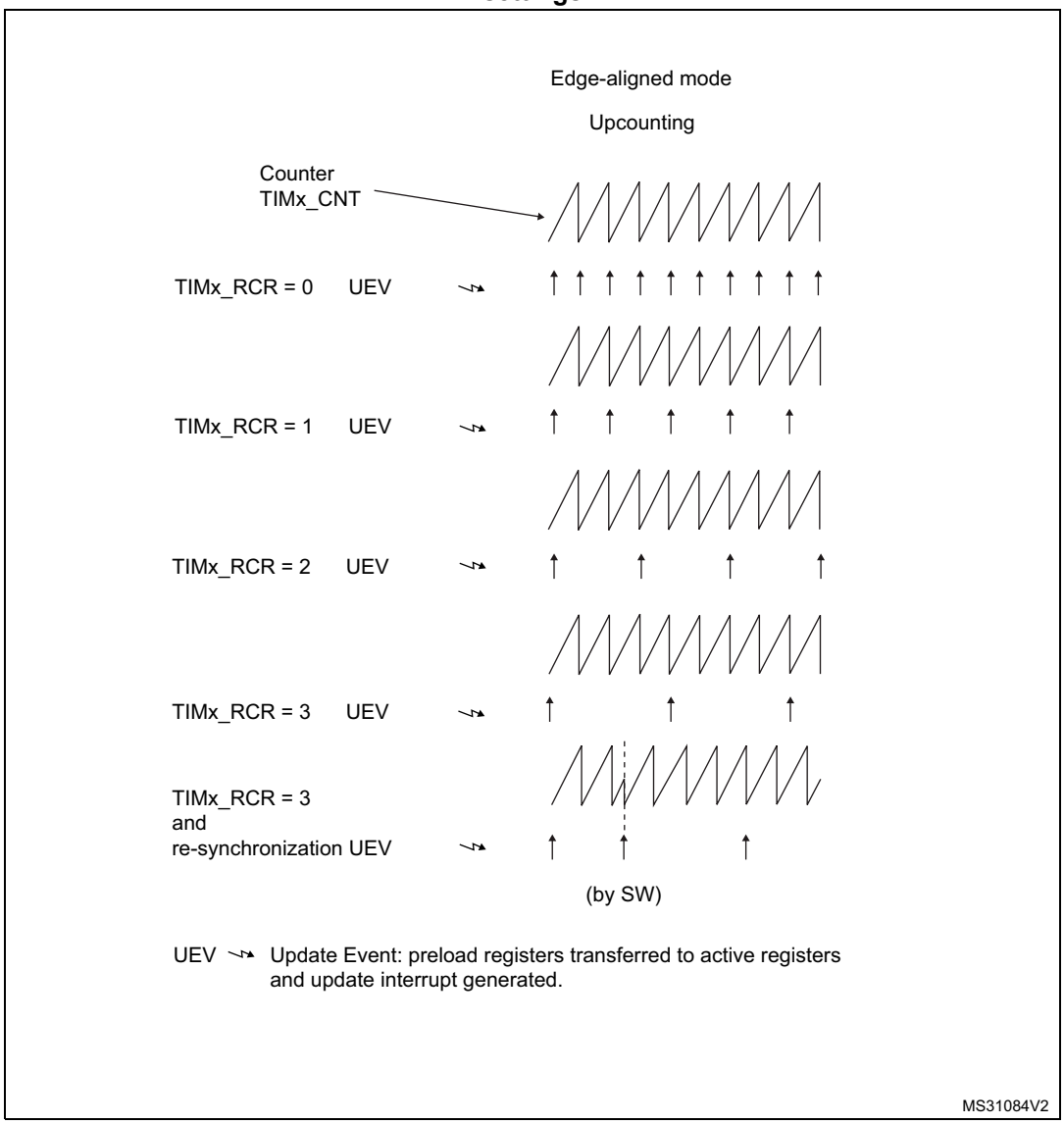
Section 40.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 488*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 488. Update rate examples depending on mode and TIMx_RCR register settings



40.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, you can configure TIM1 to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another timer on page 1658* for more details.

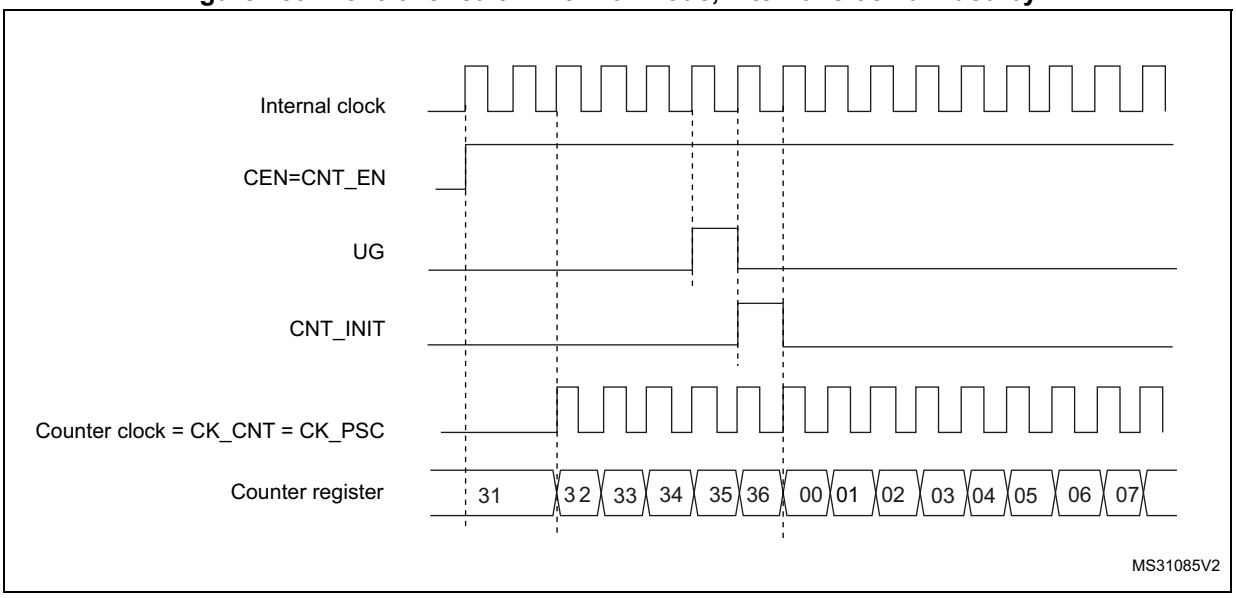
Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 489 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

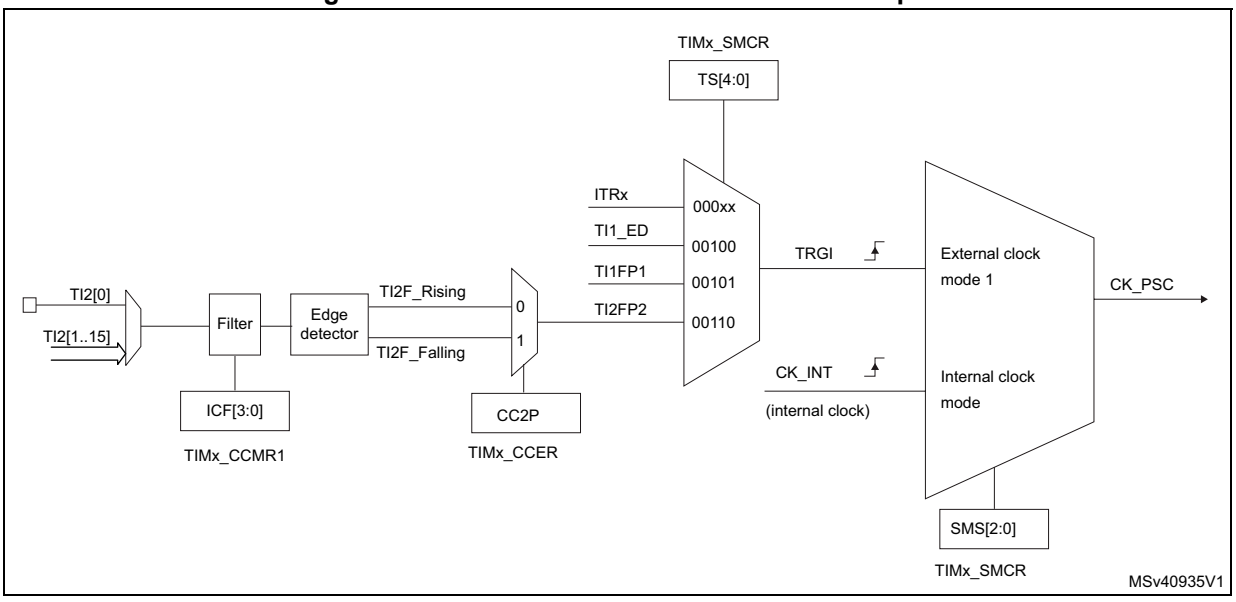
Figure 489. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 490. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

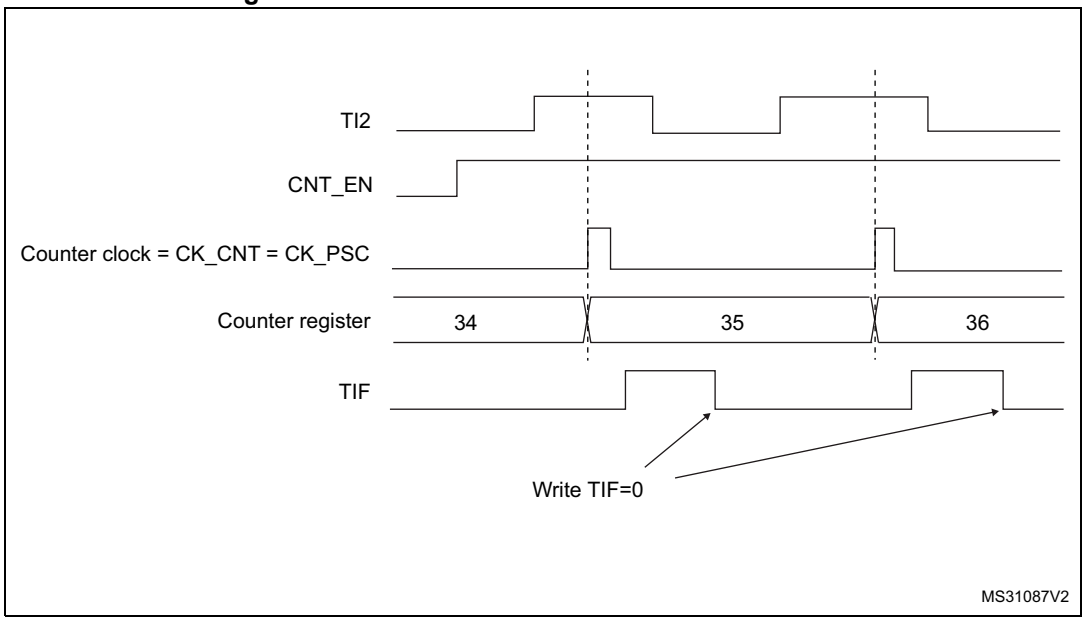
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don’t need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 491. Control circuit in external clock mode 1



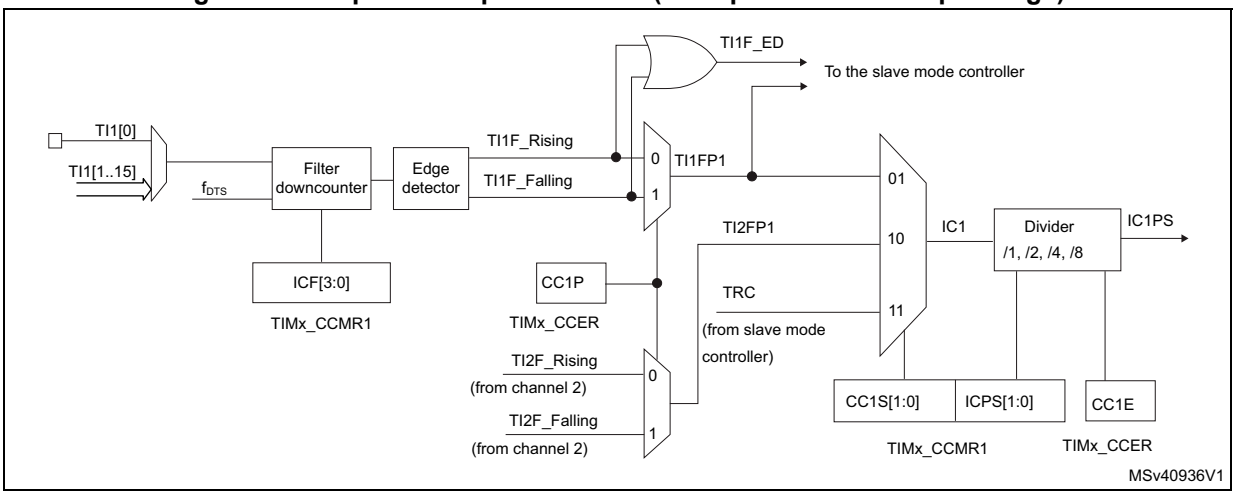
40.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 492 to *Figure 495* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 492. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 493. Capture/compare channel 1 main circuit

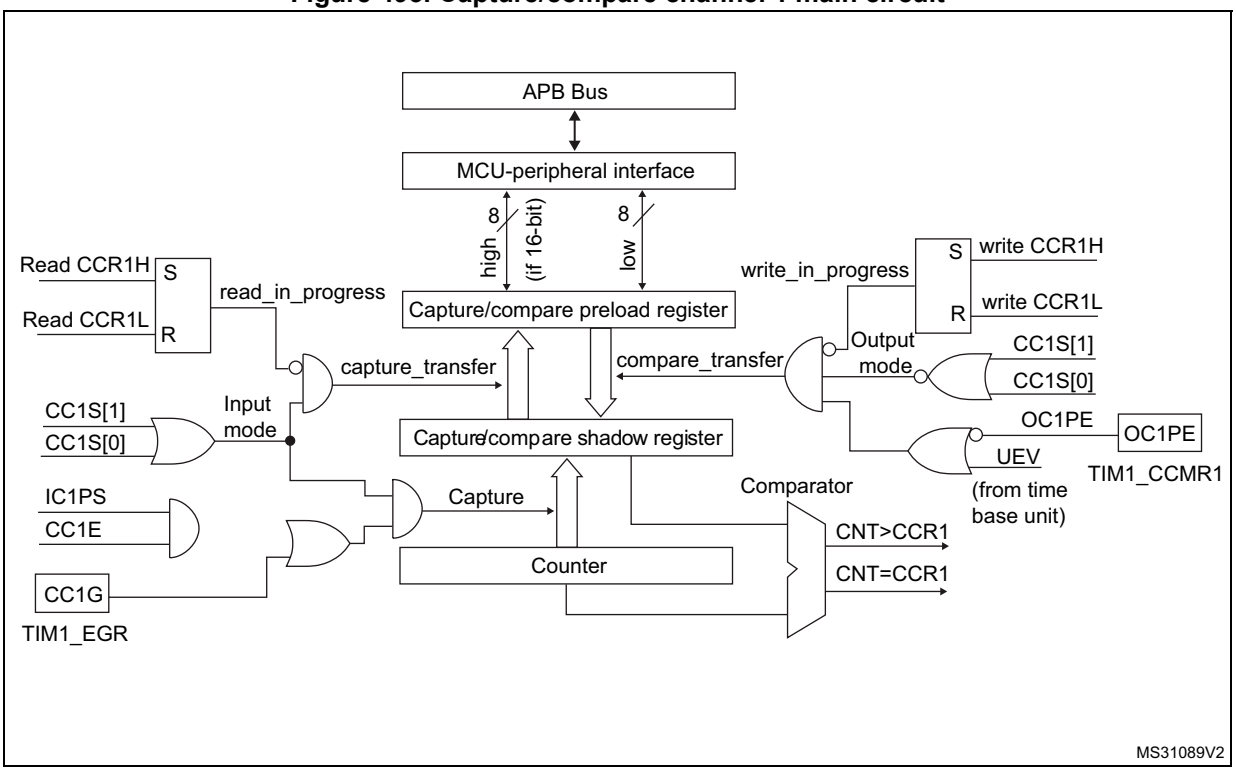


Figure 494. Output stage of capture/compare channel (channel 1)

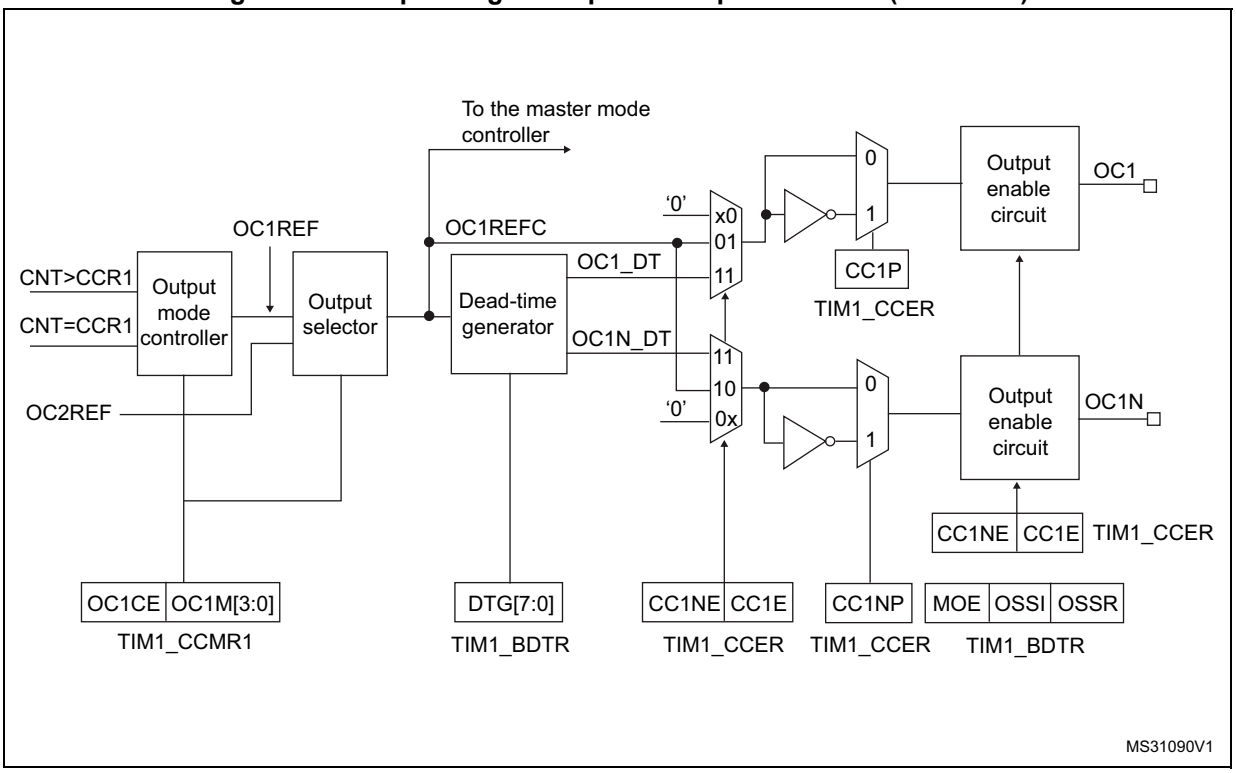
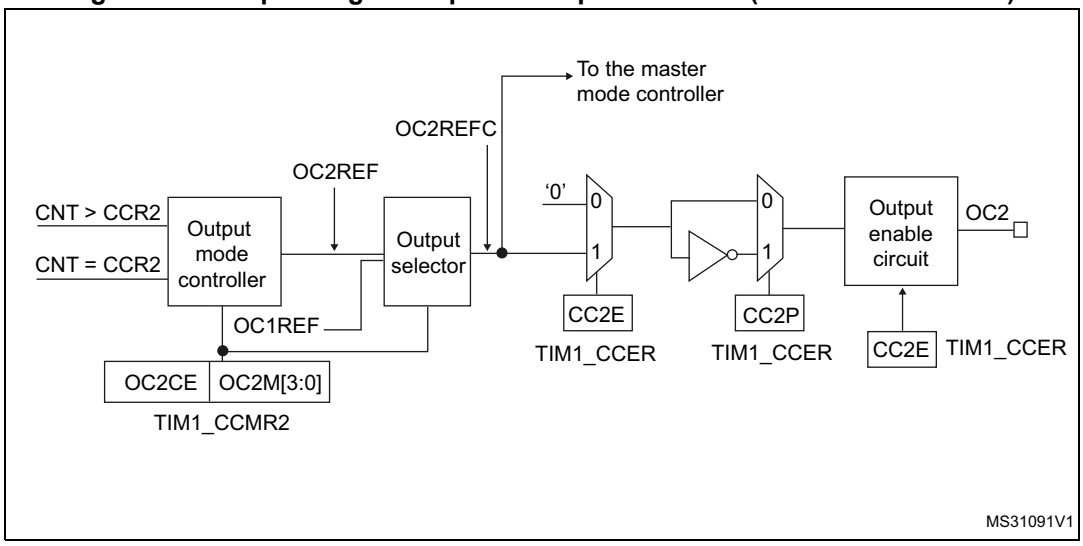


Figure 495. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

40.4.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the input filter duration you need with respect to the signal you connect to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

40.4.7 PWM input mode (only for TIM15)

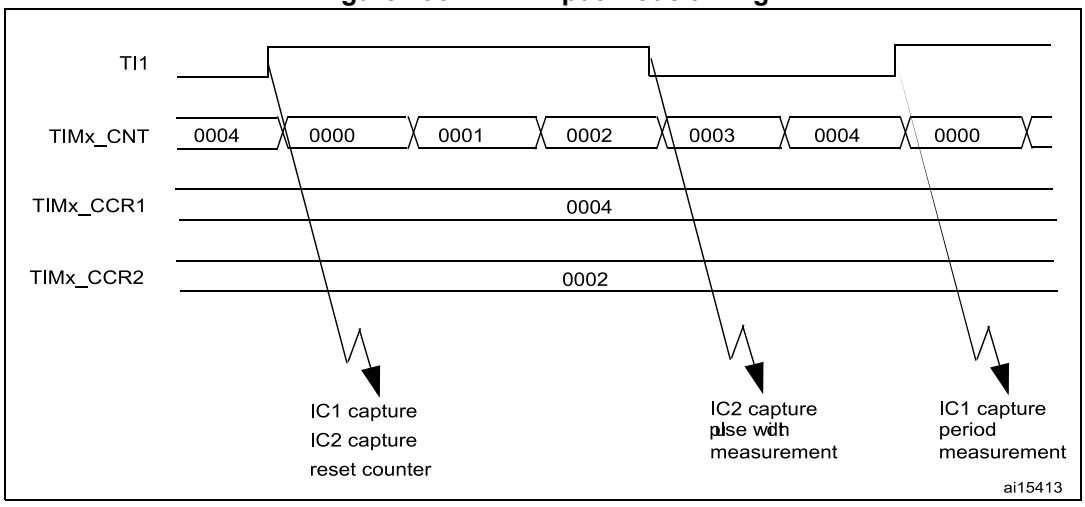
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘1’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 496. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

40.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

40.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

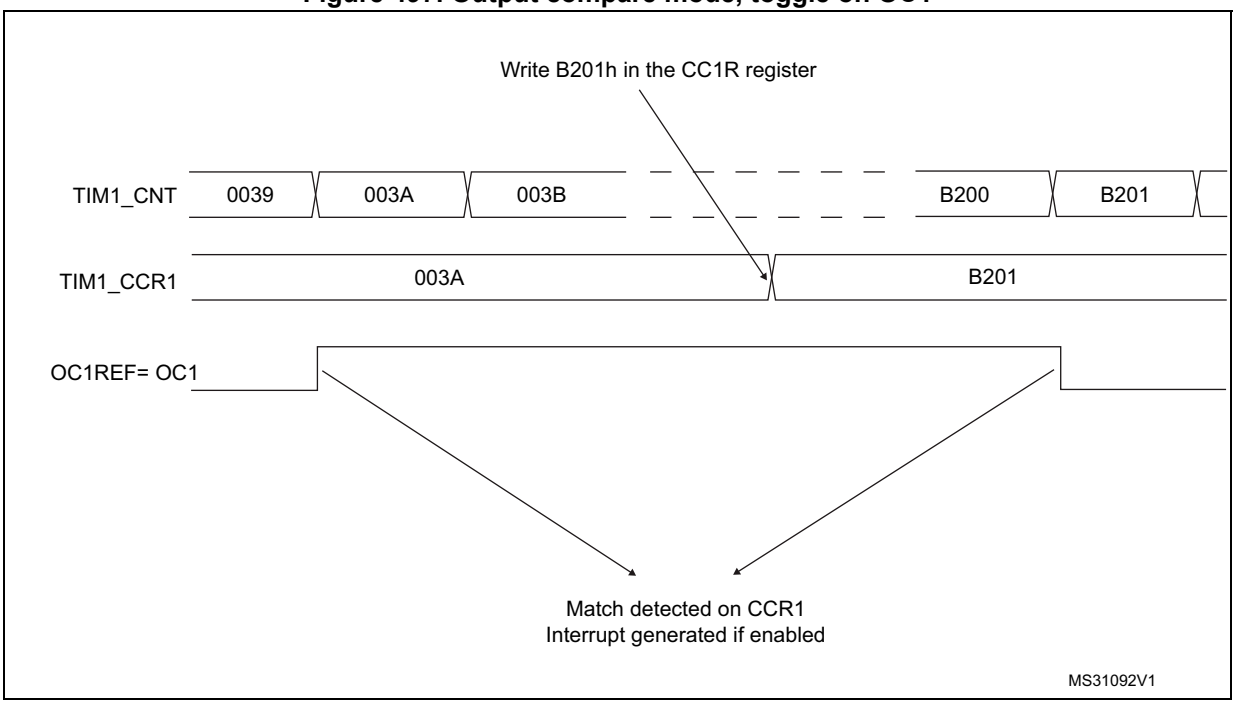
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 497*.

Figure 497. Output compare mode, toggle on OC1



40.4.10 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

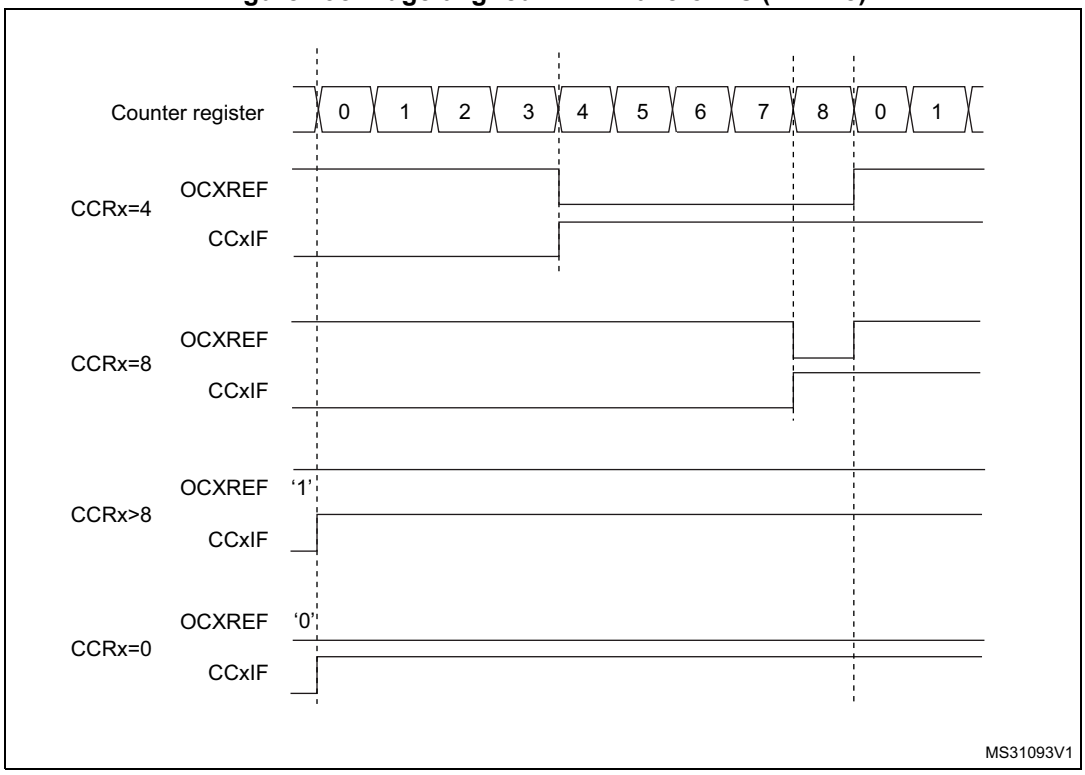
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 1752*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 498* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 498. Edge-aligned PWM waveforms (ARR=8)



40.4.11 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined

by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

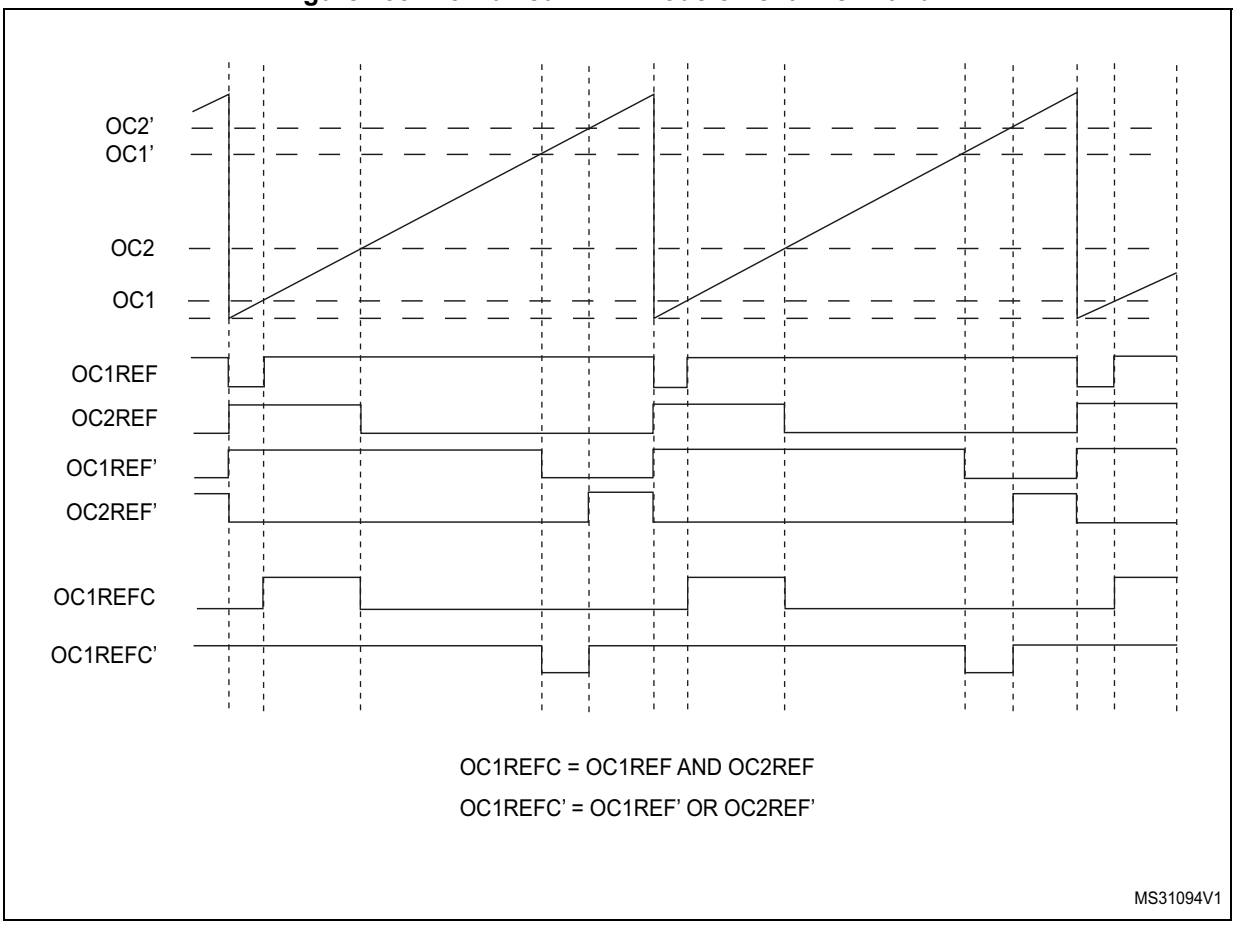
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 499 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 499. Combined PWM mode on channel 1 and 2



40.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 338: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 1817* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 500. Complementary output with dead-time insertion.

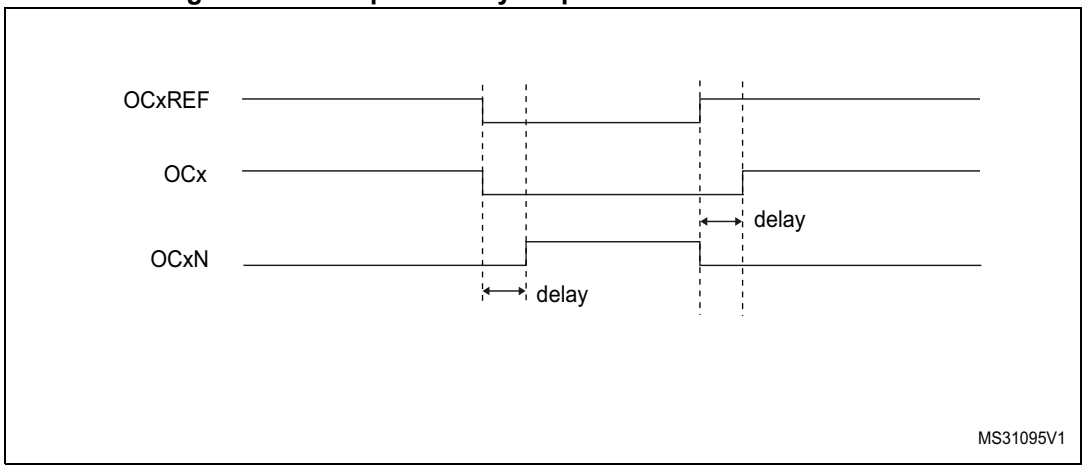


Figure 501. Dead-time waveforms with delay greater than the negative pulse.

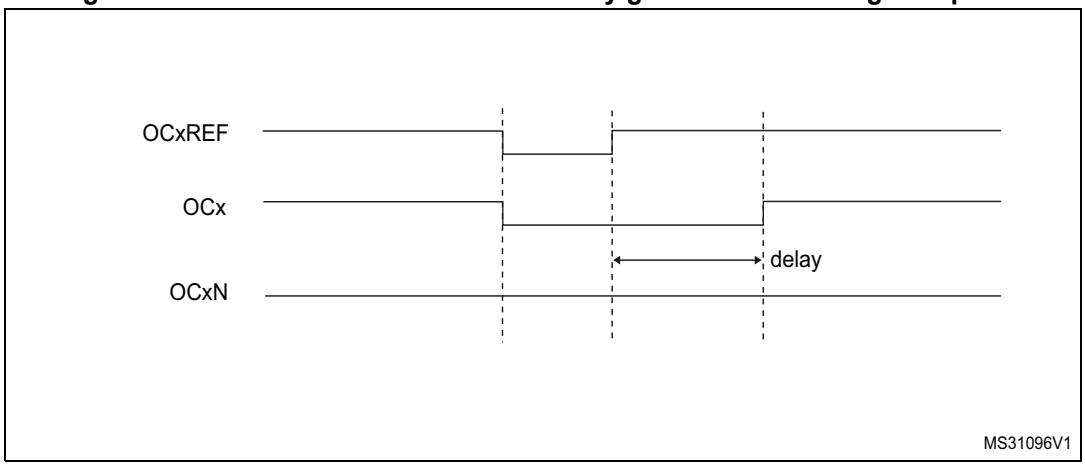
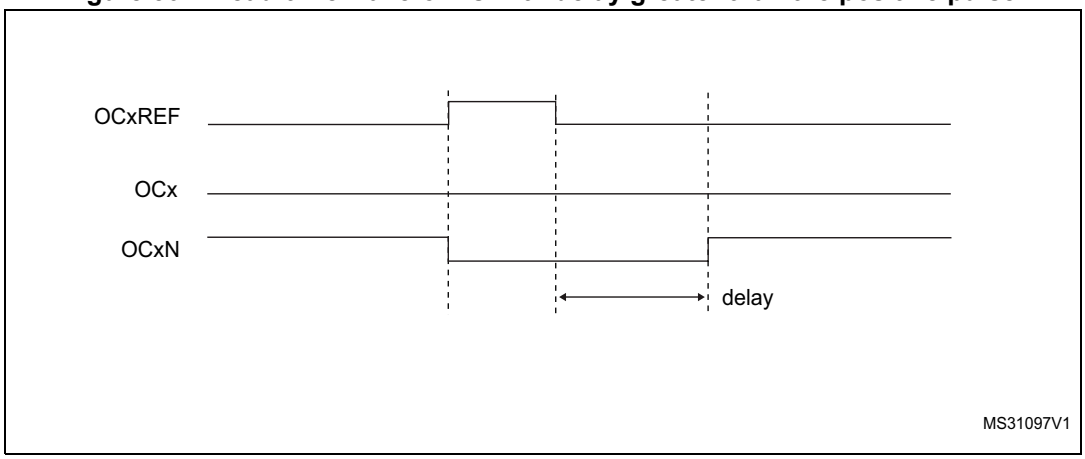


Figure 502. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 40.6.13: TIMx break and dead-time register (TIMx_BDTR)(x = 16, 17) on page 1820* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

40.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 336: Output control bits for complementary OCx and OCxN channels with break feature (TIM15) on page 1797* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if you write MOE to 1 whereas it was low, you must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read the synchronous signal.

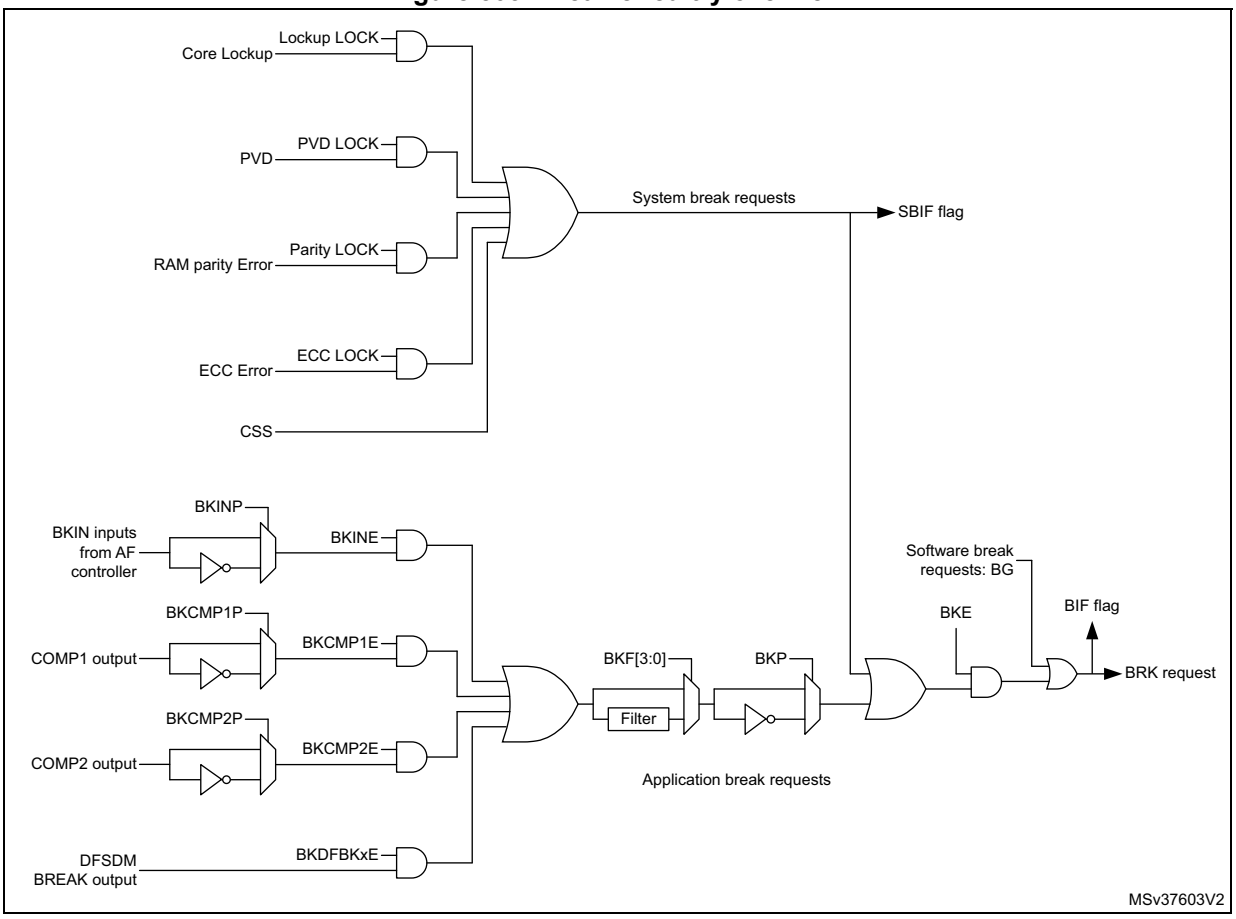
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_OR2 register.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the AFIO controller), with polarity selection and optional digital filtering
- An internal source:
  - the output from a comparator, with polarity selection and optional digital filtering
  - the analog watchdog output of the DFSDM1 peripheral
  - A system break:
    - the Cortex®-M7 with FPU LOCKUP output
    - the PVD output
    - the SRAM parity error signal
    - a Flash ECC error
    - a clock failure event generated by the CSS detector

Figure 503. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the AFIO controller (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the AFIO controller) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the AFIO controller which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until you write it to '1' again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

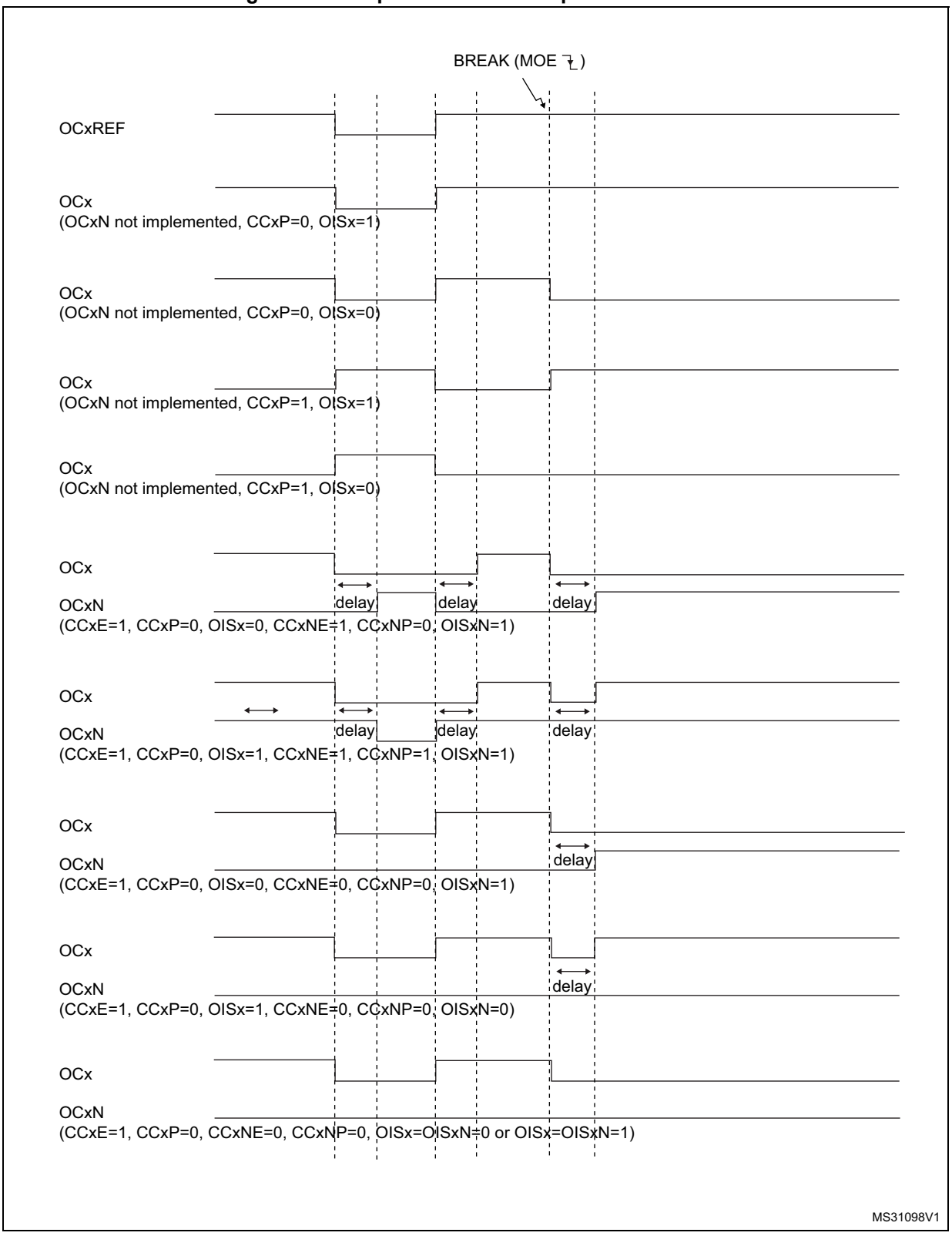
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 40.6.13: TIMx break and dead-time register (TIMx_BDTR)(x = 16, 17) on page 1820*. The LOCK bits can be written only once after an MCU reset.

The *Figure 504* shows an example of behavior of the outputs in response to a break.

Figure 504. Output behavior in response to a break



40.4.14 One-pulse mode

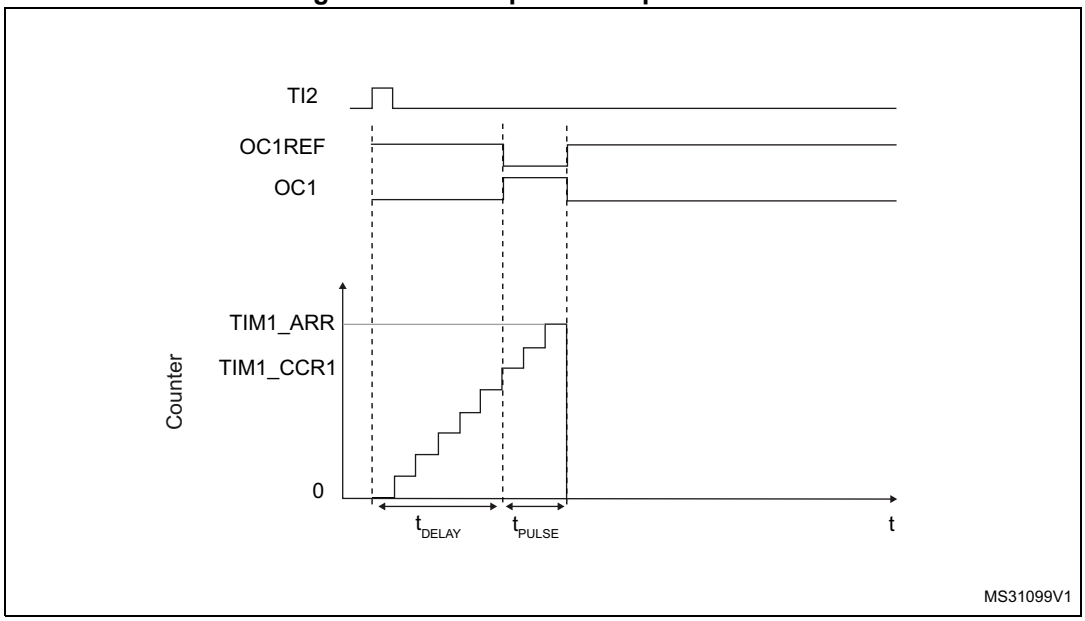
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 505. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

You only want 1 pulse, so you write '1' in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

40.4.15 Retriggerable one pulse mode (OPM) (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 40.4.14*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

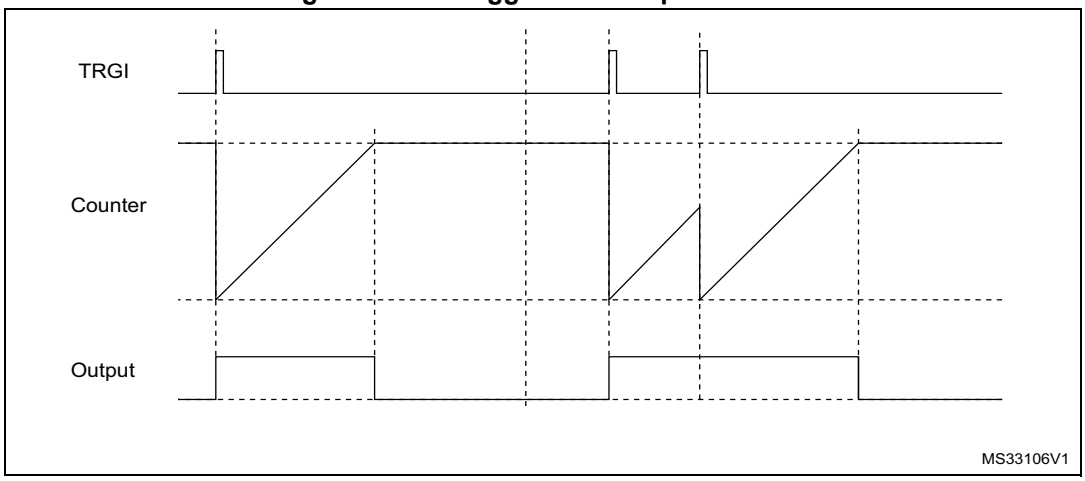
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 506. Retriggerable one pulse mode



40.4.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

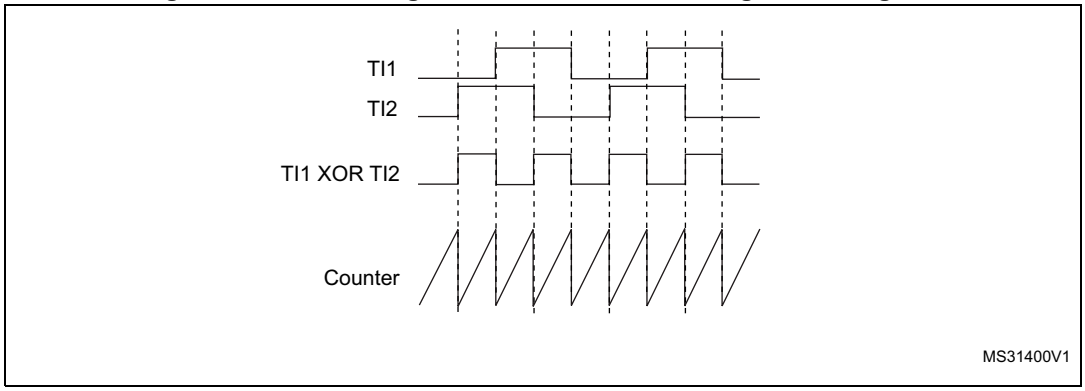
There is no latency between the assertions of the UIF and UIFCPY flags.

40.4.17 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 507*.

Figure 507. Measuring time interval between edges on 2 signals



40.4.18 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

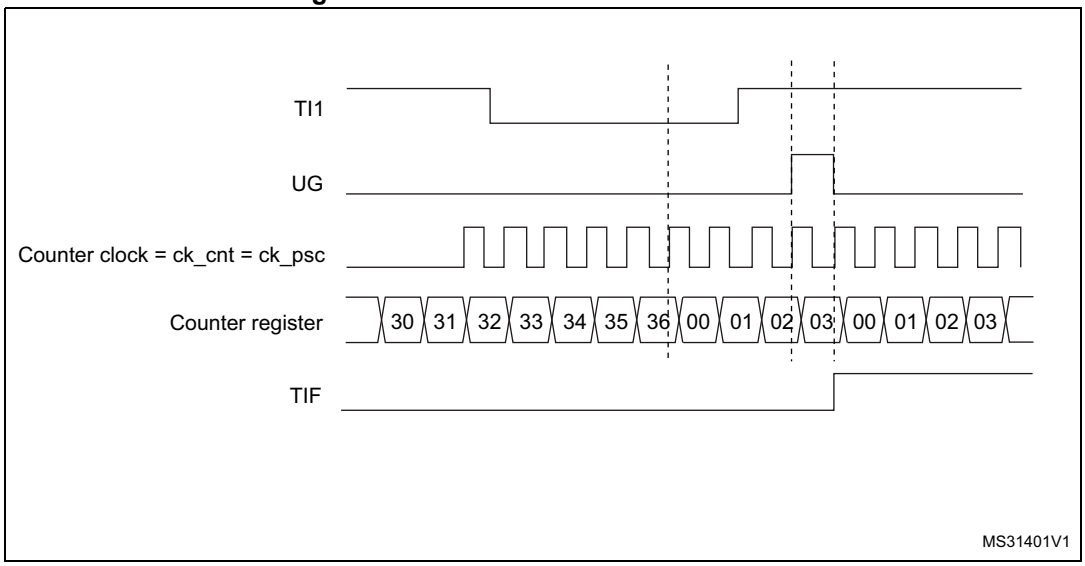
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 508. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

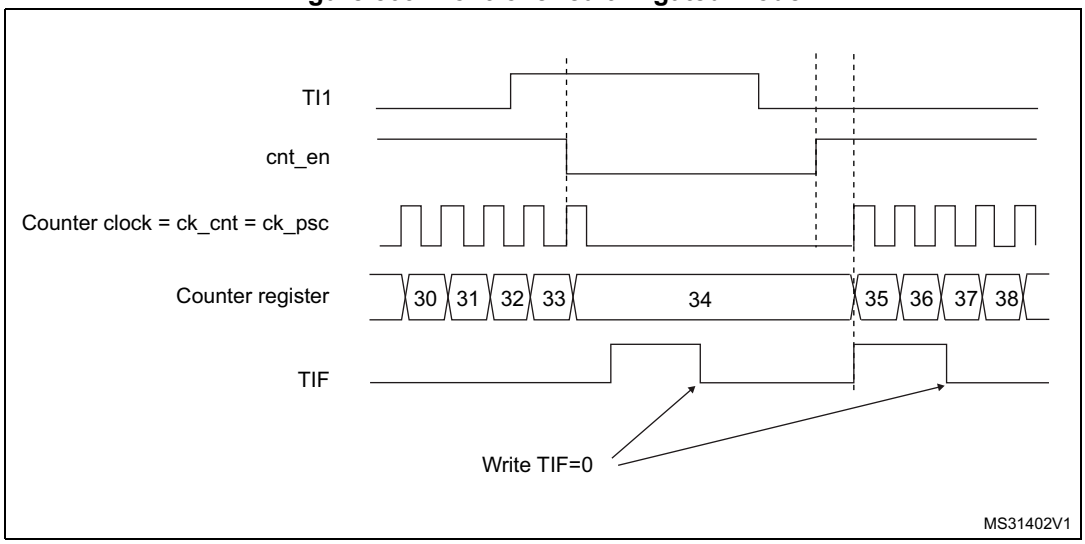
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = '0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 509. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

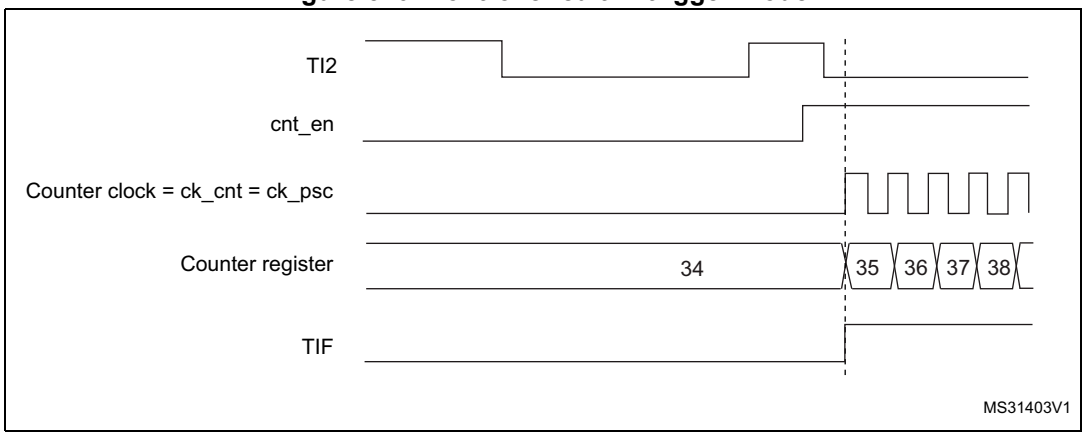
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 510. Control circuit in trigger mode



40.4.19 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

40.4.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

40.4.21 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 38.3.19: Timer synchronization* for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

40.4.22 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 with FPU core halted), the TIMx counter either continues to work normally or stops, depending on TIMx bit in DBGMCU module. For more details, refer to *Section 59: Debug infrastructure*.

For safety purposes, when the counter is stopped (TIMx = 1 in DBGMCU_APB2FZ1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

40.5 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

40.5.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (TIx)
00: $t_{DTS} = t_{CK_INT}$
01: $t_{DTS} = 2*t_{CK_INT}$
10: $t_{DTS} = 4*t_{CK_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

40.5.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (OC2 output)
0: OC2=0 when MOE=0
1: OC2=1 when MOE=0
Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIMx_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REF signal is used as trigger output (TRGO).
101: **Compare** - OC2REF signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

40.5.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **SMS[3]:** Slave mode selection - bit 3
Refer to SMS description - bits 2:0.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode
0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection
This bit field selects the trigger input to be used to synchronize the counter.
00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
Other: Reserved
See *Table 335: TIMx Internal trigger connection on page 1787* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Other codes: reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=’00100’). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 335. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM15	TIM1	TIM3	TIM16 OC1	TIM17 OC1

40.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

40.5.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
If channel CC1 is configured as output: This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input: This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 40.5.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

40.5.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

40.5.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							Res.								Res.
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, always read as 0

Bit 24 **OC2M[3]**: Output Compare 2 mode - bit 3

Bits 23:17 Reserved, always read as 0

Bit 16 **OC1M[3]**: Output Compare 1 mode - bit 3
refer to OC1M description on bits 6:4

Bit 15 Reserved, always read as 0

Bits 14:12 **OC2M[2:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 Reserved, always read as 0

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. 0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved,

1111: Reserved,

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

***3**: On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:16 Reserved, always read as 0

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

40.5.8 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: OC1N active high
1: OC1N active low
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: 1.This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input: The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
00: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
01: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).
10: reserved, do not use this configuration.
11: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

Note: 1. This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

2. On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input: This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled

1: Capture enabled

Table 336. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>OCx output state</th><th>OCxN output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
OCx=0
OCxN=0, OCxN_EN=0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCx=0</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF XOR CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCxN=0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
OCx=CCxP</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF xor CCxP,
OCx_EN=1</td><td>Off-State (output enabled with inactive state)
OCxN=CCxNP, OCxN_EN=1</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: OCx=CCxP, OCxN=CCxNP
Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

40.5.9 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

40.5.10 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

40.5.11 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 40.4.1: Time-base unit on page 1750* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

40.5.12 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

40.5.13 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

40.5.14 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2).

40.5.15 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample the BRK input signal and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)

See OC/OCN enable description for more details (*Section 40.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 1795*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low
1: Break input BRK is active high

*Note: **1:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

__2:__ Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled
1; Break inputs (BRK and CCS clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 40.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 1795*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (*Section 40.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 1795*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

40.5.16 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

40.5.17 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

40.5.18 TIM15 alternate register 1 (TIM15_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK0E**: BRK dfsdm1_break[0] enable

This bit enables the dfsdm1_break[0] for the timer's BRK input. dfsdm1_break[0] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[0]input disabled

1: dfsdm1_break[0]input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

40.5.19 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM15_CH2 input

0001: TIM2_CH2 input

0010: TIM3_CH2 input

0011: TIM4_CH2 input

Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM15_CH1 input

0001: TIM2_CH1 input

0010: TIM3_CH1 input

0011: TIM4_CH1 input

0100: LSE

0101: CSI

0110: MCO2

Other: Reserved

40.5.20 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 337. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM15_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIM15_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0
0x08	**TIM15_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x0C	**TIM15_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0			0	0	0	0	0	0			0	0	0
0x10	**TIM15_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	**TIM15_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM15_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIM15_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM15_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0

Table 337. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**TIM15_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM15_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM15_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM15_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM15_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM15_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM15_BDTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM15_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM15_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM15_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0	0						0	0	1
0x68	**TIM15_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

40.6 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

40.6.1 TIMx control register 1 (TIMx_CR1)(x = 16, 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (TIx),
00: $t_{DTS}=t_{CK_INT}$
01: $t_{DTS}=2*t_{CK_INT}$
10: $t_{DTS}=4*t_{CK_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

40.6.2 TIMx control register 2 (TIMx_CR2)(x = 16, 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

40.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16, 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
		rw				rw	rw	rw		rw				rw	rw

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

40.6.4 TIMx status register (TIMx_SR)(x = 16, 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

40.6.5 TIMx event generation register (TIMx_EGR)(x = 16, 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

40.6.6 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16, 17)

Address offset: 0x18

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															Res.
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, always read as 0

Bit 16 **OC1M[3]:** Output Compare 1 mode (bit 3)

Bits 15:7 Reserved

Bits 6:4 **OC1M[2:0]**: Output Compare 1 mode (bits 2 to 0)

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.
0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.
0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
0100: Force inactive level - OC1REF is forced low.
0101: Force active level - OC1REF is forced high.
0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.
0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.
All other values: Reserved

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

40.6.7 TIMx capture/compare enable register (TIMx_CCER)(x = 16, 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high
1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high
1: OC1 active low

CC1 channel configured as input:

The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

00: Non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

01: Inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode.

10: Reserved, do not use this configuration.

1: Non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

*Note: **1.** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

***2.** On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:

0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.

0: Capture disabled
1: Capture enabled

Table 338. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer anymore). The output state is defined by the GPIO controller and can be High, Low or Hi-Z.	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and AFIO registers.

40.6.8 TIMx counter (TIMx_CNT)(x = 16, 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

40.6.9 TIMx prescaler (TIMx_PSC)(x = 16, 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

40.6.10 TIMx auto-reload register (TIMx_ARR)(x = 16, 17)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 40.4.1: Time-base unit on page 1750* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

40.6.11 TIMx repetition counter register (TIMx_RCR)(x = 16, 17)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

40.6.12 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16, 17)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

40.6.13 TIMx break and dead-time register (TIMx_BDTR)(x = 16, 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, BRK acts asynchronously
0001: $f_{SAMPLING}=f_{CK_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8
This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (*Section 40.6.7: TIMx capture/compare enable register (TIMx_CCER)(x = 16, 17) on page 1815*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: **1.** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

__2.__ Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1; Break inputs (BRK and CCS clock failure event) enabled

*Note: **1.** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

__2.__ Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 40.6.7: TIMx capture/compare enable register (TIMx_CCER)(x = 16, 17) on page 1815*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the AFIO logic, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.
See OC/OCN enable description for more details (*Section 40.6.7: TIMx capture/compare enable register (TIMx_CCER)(x = 16, 17) on page 1815*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

40.6.14 TIMx DMA control register (TIMx_DCR)(x = 16, 17)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

40.6.15 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16, 17)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

40.6.16 TIM16 alternate function register 1 (TIM16_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK1E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDFBK1E**: BRK dfsdm1_break[1] enable

This bit enables the dfsdm1_break[1] for the timer's BRK input. dfsdm1_break[1] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[1] input disabled

1: dfsdm1_break[1] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

40.6.17 TIM16 input selection register (TIM16_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM16_CH1 input

0001: LSI

0010: LSE

0011: WKUP_IT

Other: Reserved

40.6.18 TIM17 alternate function register 1 (TIM17_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BKCMP2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable

This bit enables the dfsdm1_break[2] for the timer's BRK input. dfsdm1_break[2] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[2] input disabled

1: dfsdm1_break[2] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

40.6.19 TIM17 input selection register (TIM17_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM17_CH1 input

0001: SPDIFRX_FRAME_SYNC is connected to TIM17_CH1 to measure the clock drift of the received SPDIF frames

0010: HSE_1MHz

0011: MCO1

Other: Reserved

40.6.20 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 339. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																			0				0	0	0		0				0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 339. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	**TIMx_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIMx_BDTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**TIM16_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK1E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0	0						0	0	1
0x60	**TIM17_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP2P	BKCMP1P	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	BKCMP2E	BKCMP1E	BKINE
	Reset value																					0	0	0	0						0	0	1
0x68	**TIM16_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TISEL[3:0]			
	Reset value																													0	0	0	0
0x68	**TIM17_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TISEL[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

41 Basic timers (TIM6/TIM7)

41.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

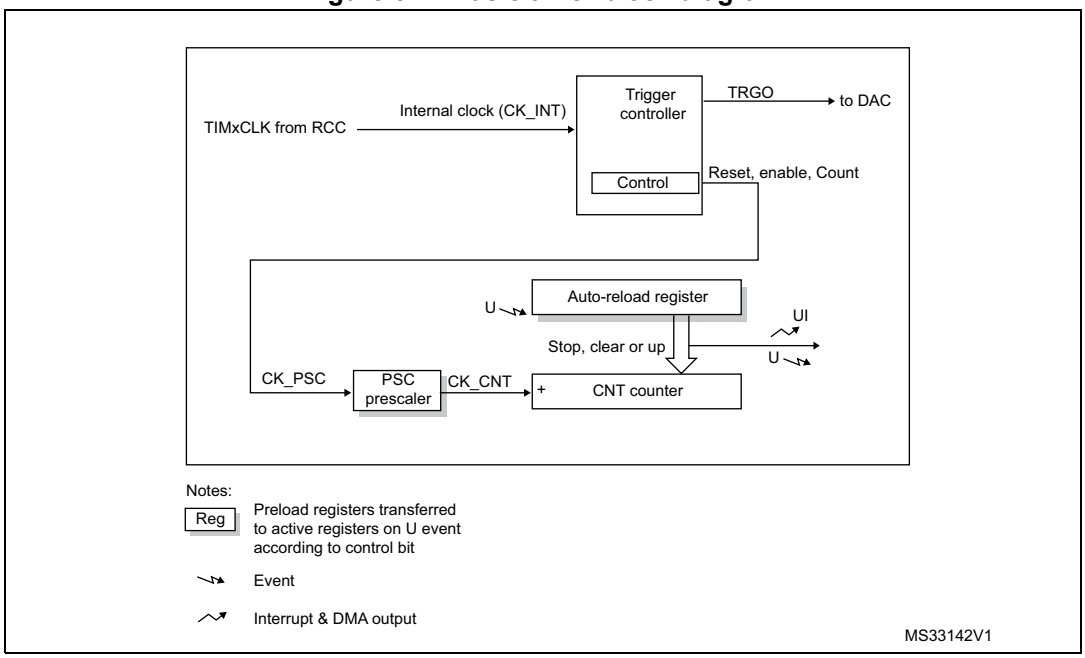
The timers are completely independent, and do not share any resources.

41.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 511. Basic timer block diagram



41.3 TIM6/TIM7 functional description

41.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 512 and *Figure 513* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 512. Counter timing diagram with prescaler division change from 1 to 2

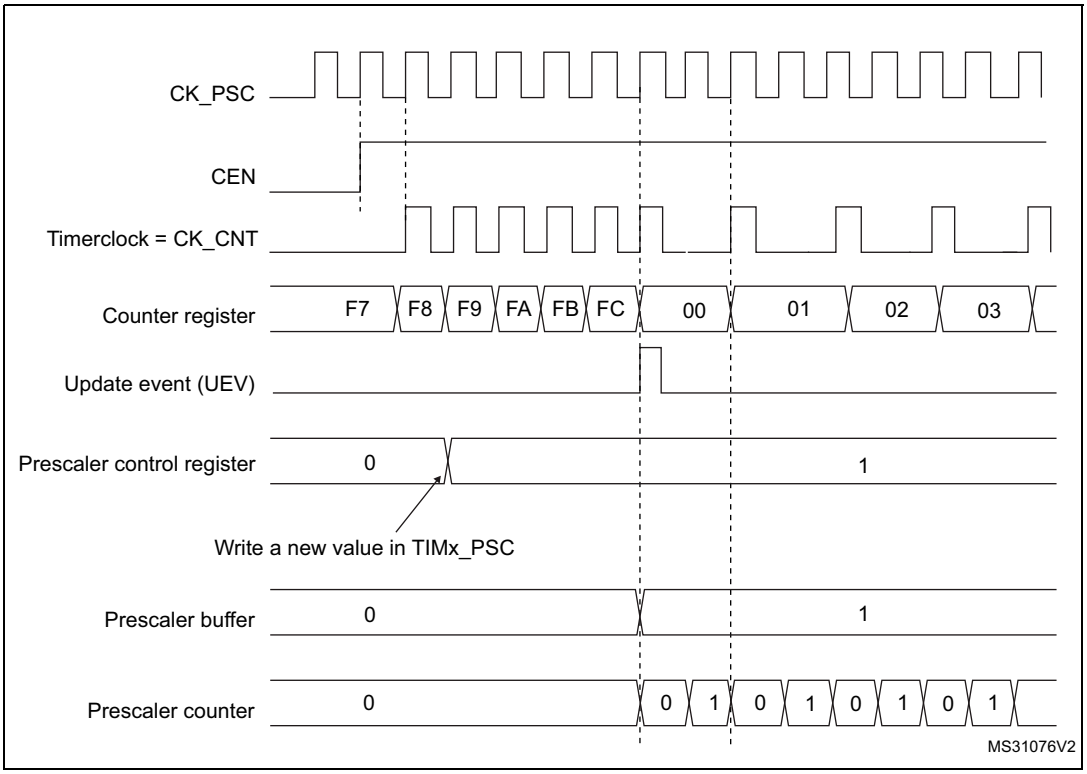
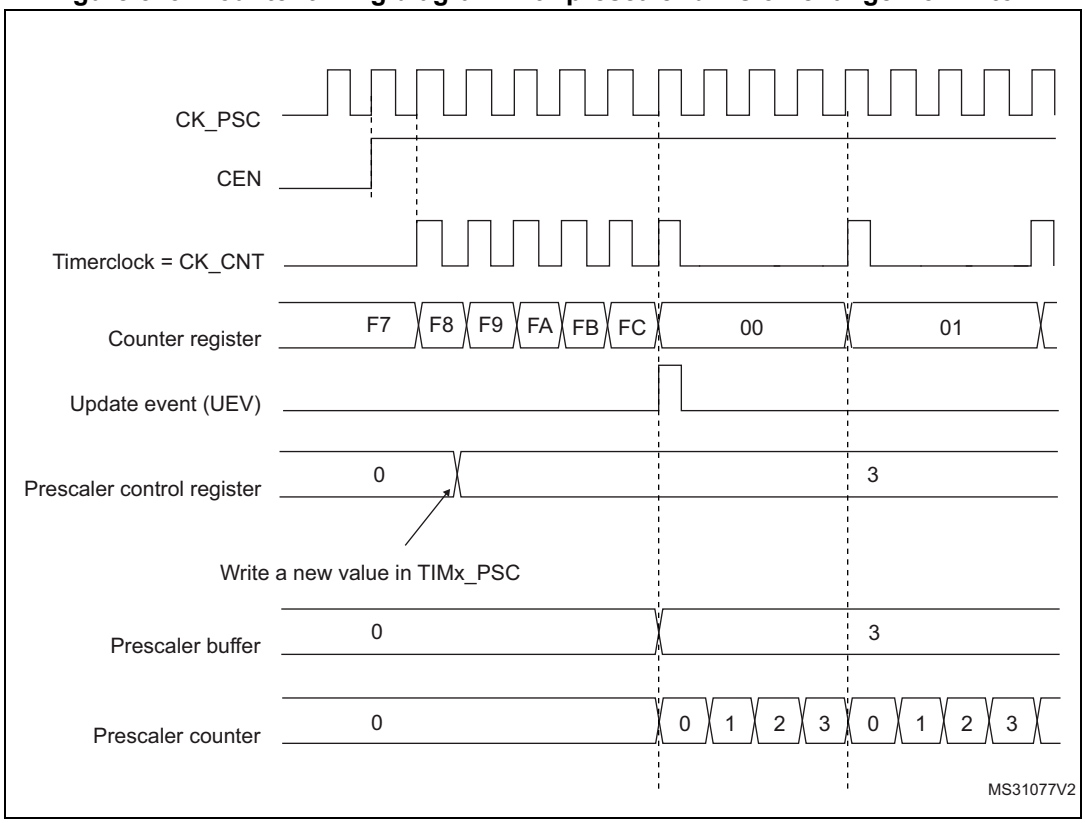


Figure 513. Counter timing diagram with prescaler division change from 1 to 4



41.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 514. Counter timing diagram, internal clock divided by 1

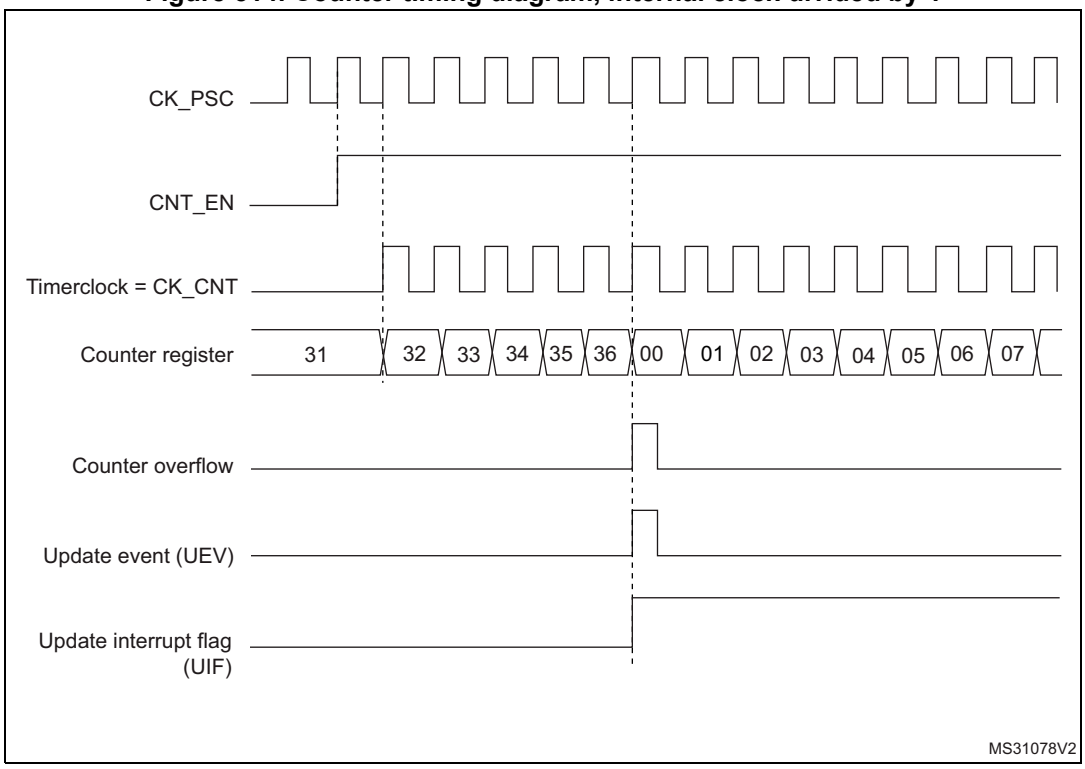


Figure 515. Counter timing diagram, internal clock divided by 2

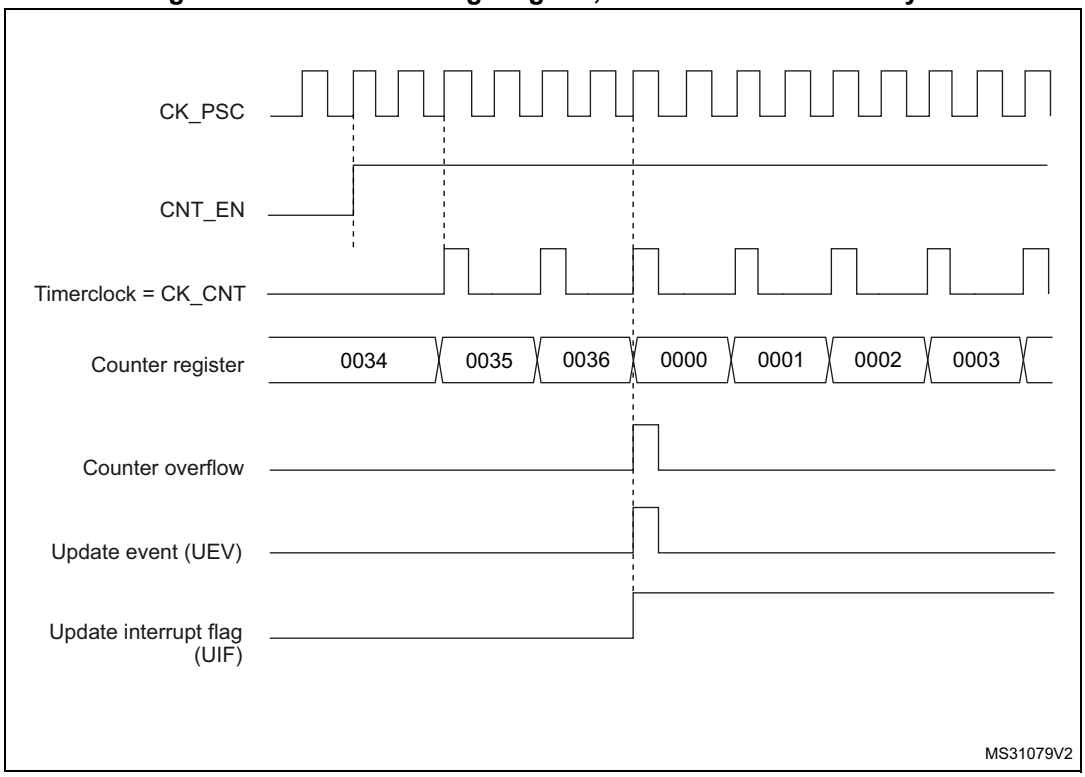


Figure 516. Counter timing diagram, internal clock divided by 4

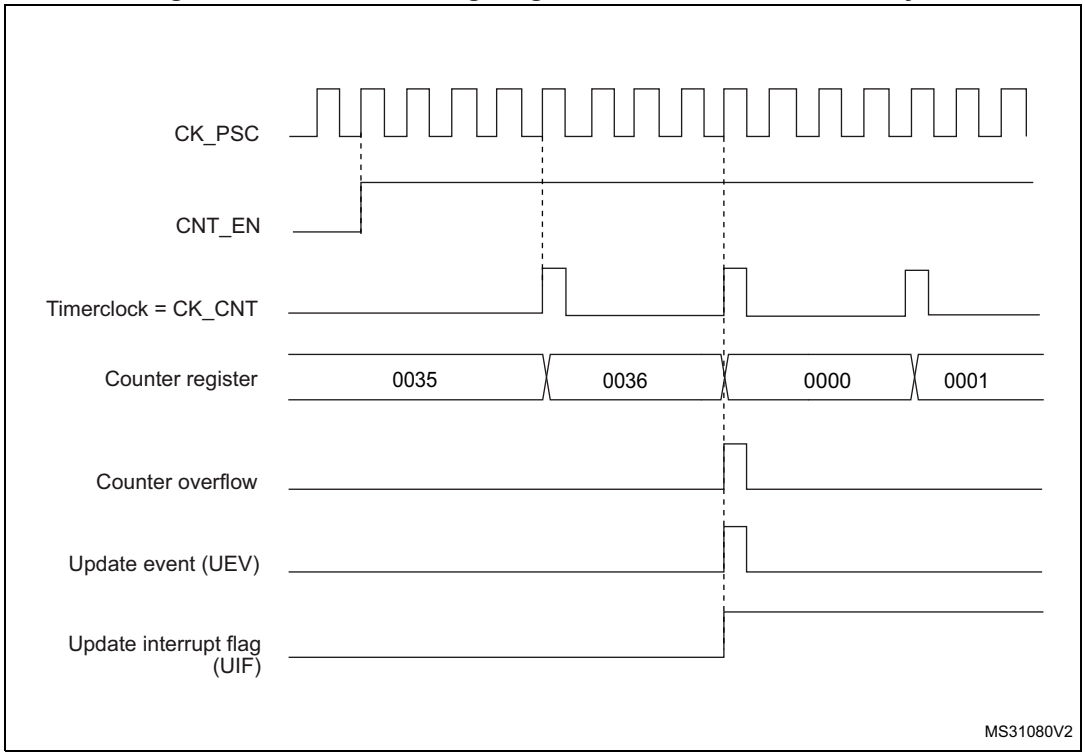


Figure 517. Counter timing diagram, internal clock divided by N

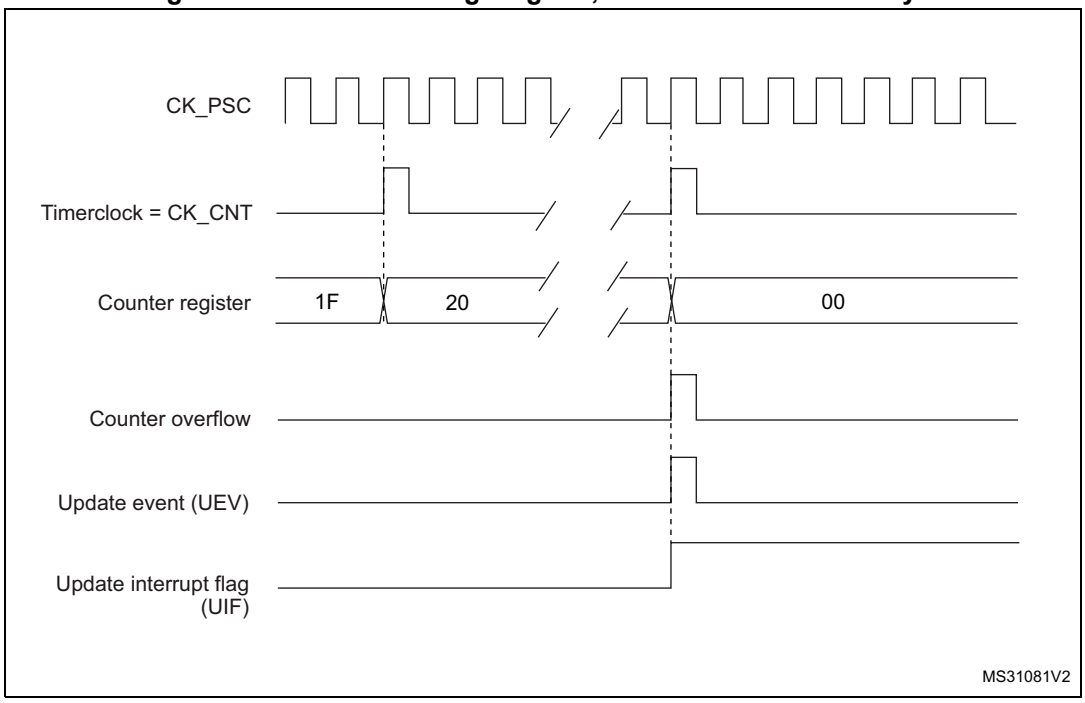


Figure 518. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

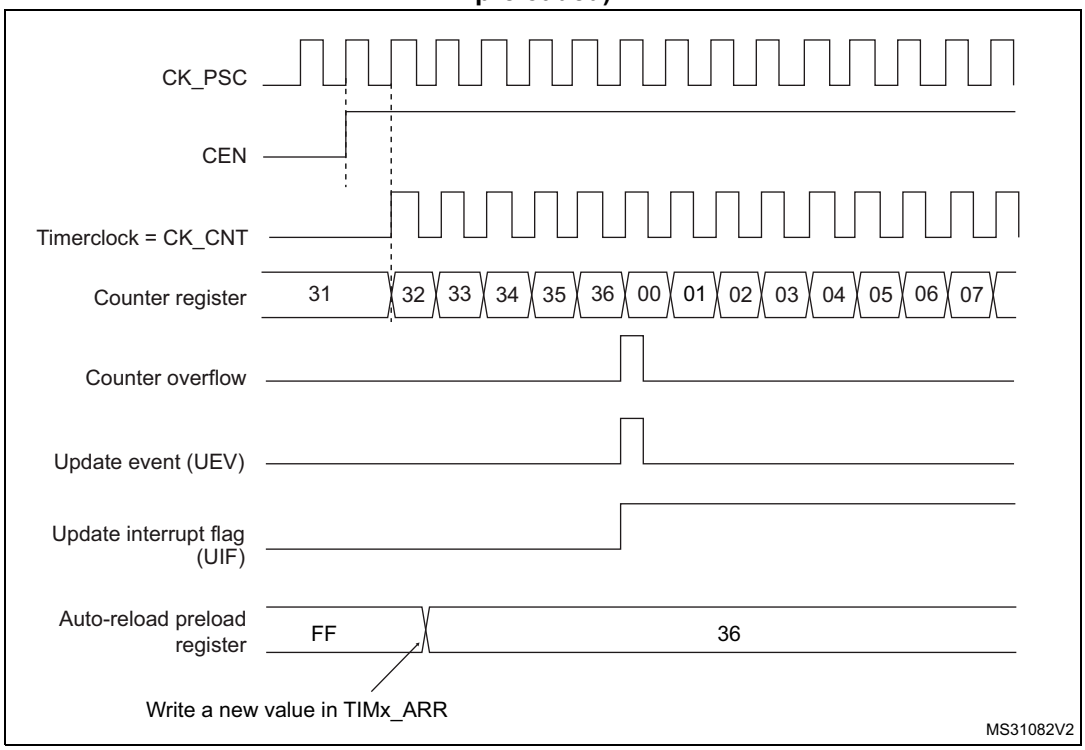
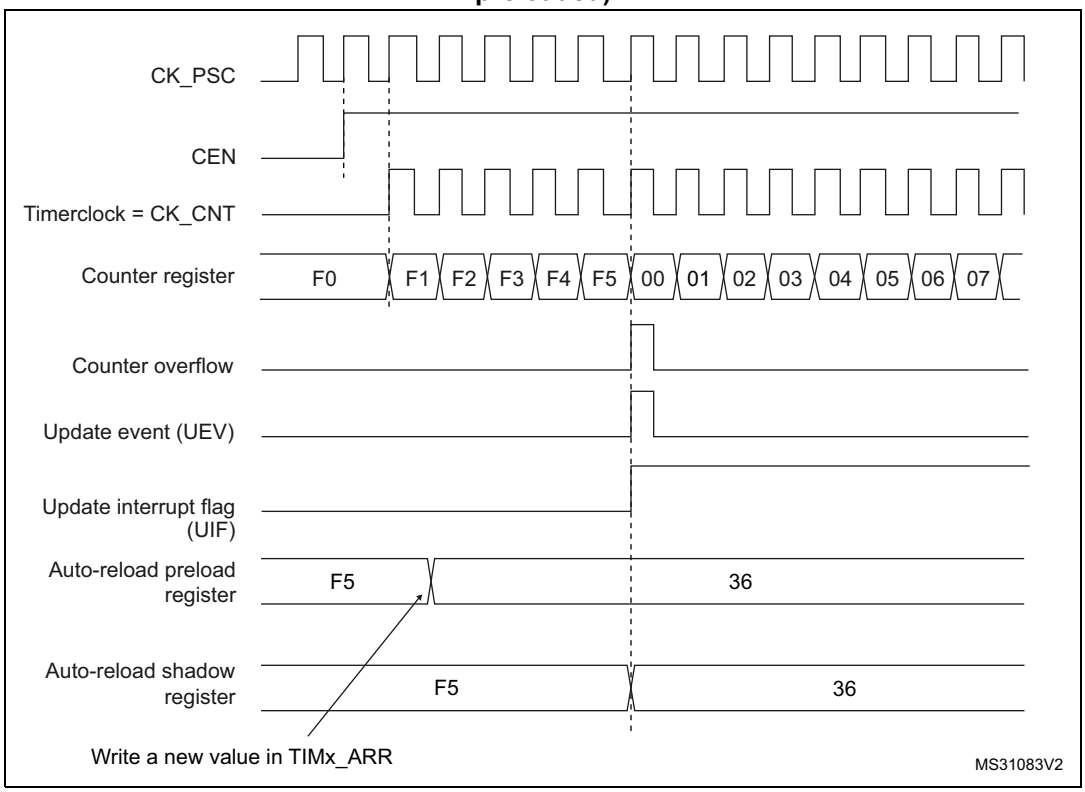


Figure 519. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



41.3.3 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

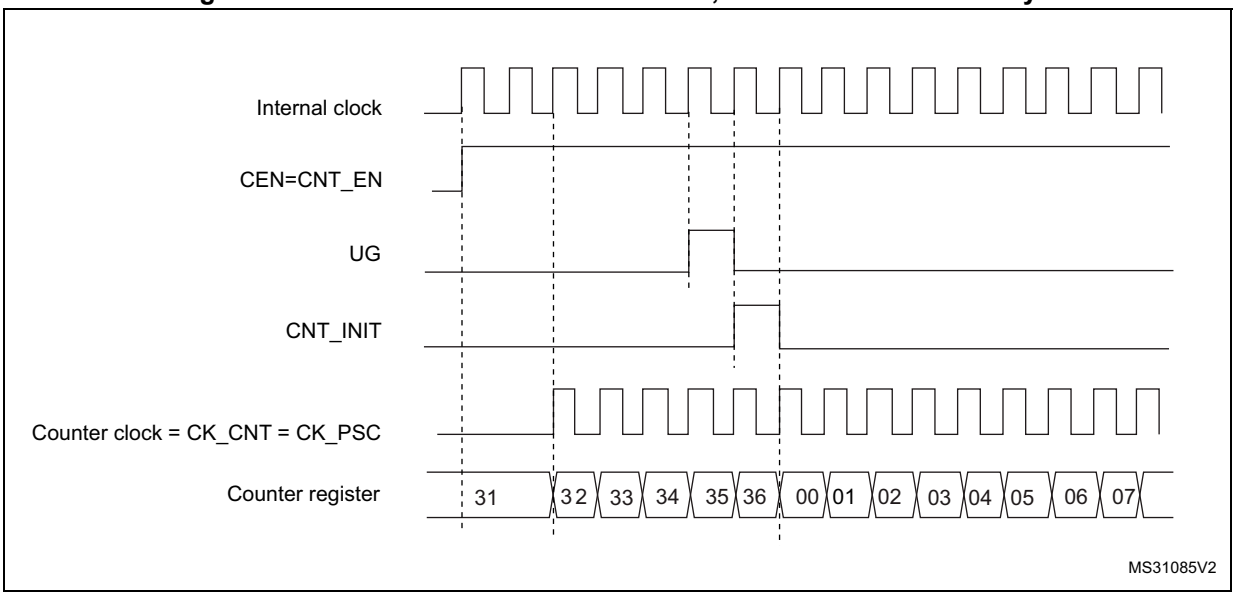
41.3.4 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 520 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 520. Control circuit in normal mode, internal clock divided by 1



41.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex®-M7 with FPU core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBGMCU module. For more details, refer to *Section 59.5.8: Microcontroller debug unit (DBGMCU)*.

41.4 TIM6/TIM7 registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

41.4.1 TIM6/TIM7 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw				rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

41.4.2 TIM6/TIM7 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

41.4.3 TIM6/TIM7 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

41.4.4 TIM6/TIM7 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

41.4.5 TIM6/TIM7 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

41.4.6 TIM6/TIM7 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

41.4.7 TIM6/TIM7 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

41.4.8 TIM6/TIM7 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value
ARR is the value to be loaded into the actual auto-reload register.
Refer to *Section 41.3.1: Time-base unit on page 1831* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

41.4.9 TIM6/TIM7 register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 340. TIM6/TIM7 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0				0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

42 Low-power timer (LPTIM)

42.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

42.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: LSE, LSI, HSI or APB clock
  - External clock source over LPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

42.3 LPTIM implementation

Table 341 describes LPTIM implementation on STM32H743/753 and STM32H750 devices.

Table 341. STM32H743/753 and STM32H750 LPTIM features

LPTIM modes/features(1)	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5
Encoder mode	X	X	-	-	-

1. X = supported.

42.4 LPTIM functional description

42.4.1 LPTIM block diagram

Figure 521. Low-power timer block diagram (LPTIM1 and LPTIM2)

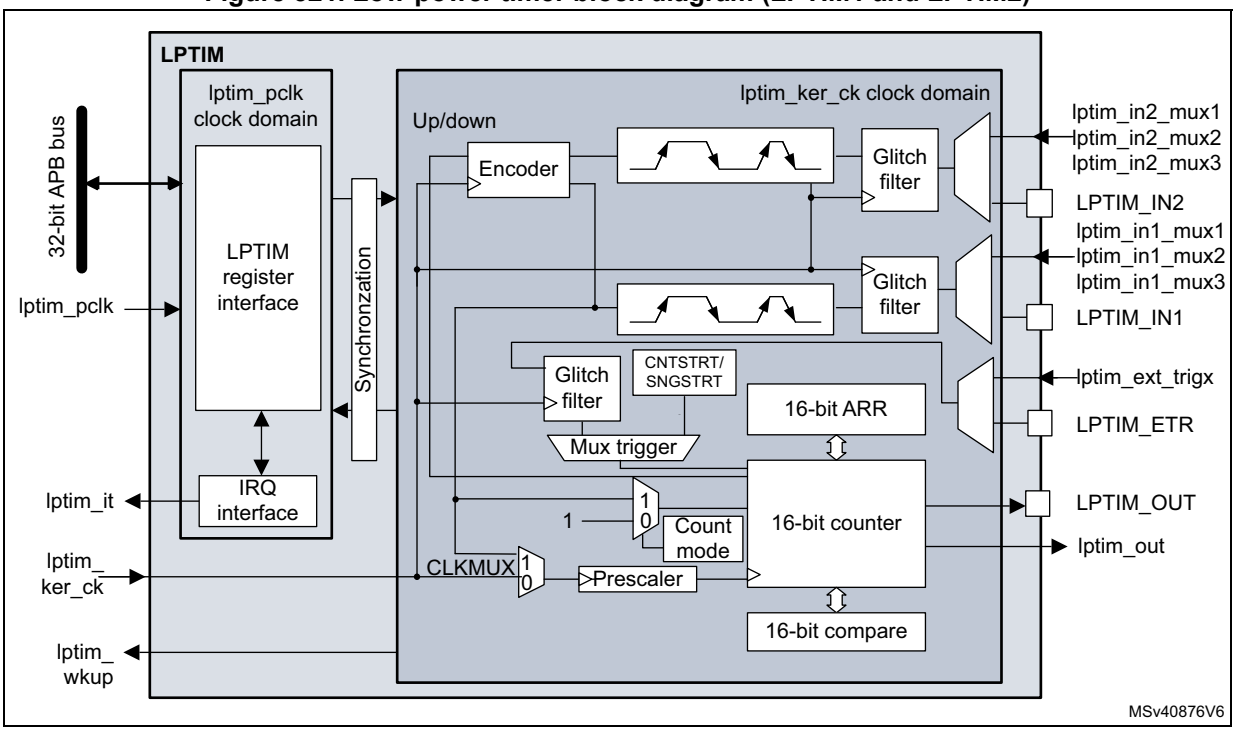


Figure 522. Low-power timer block diagram (LPTIM3)

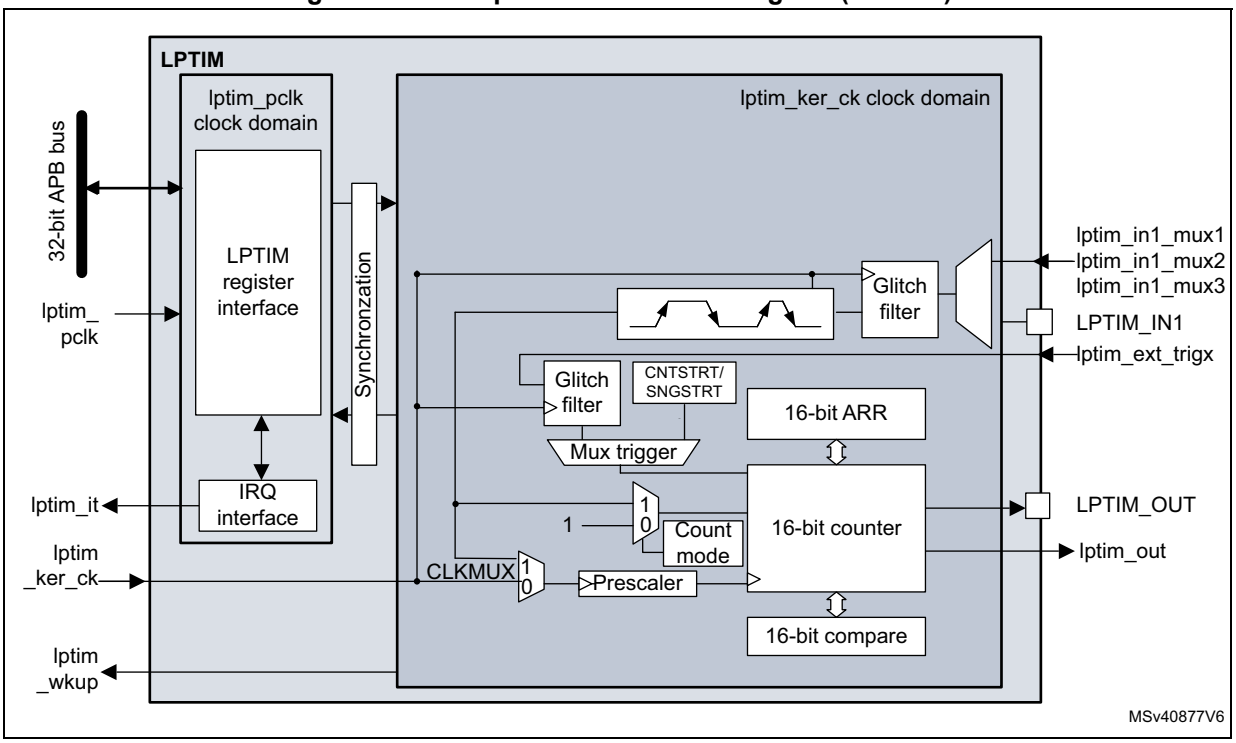
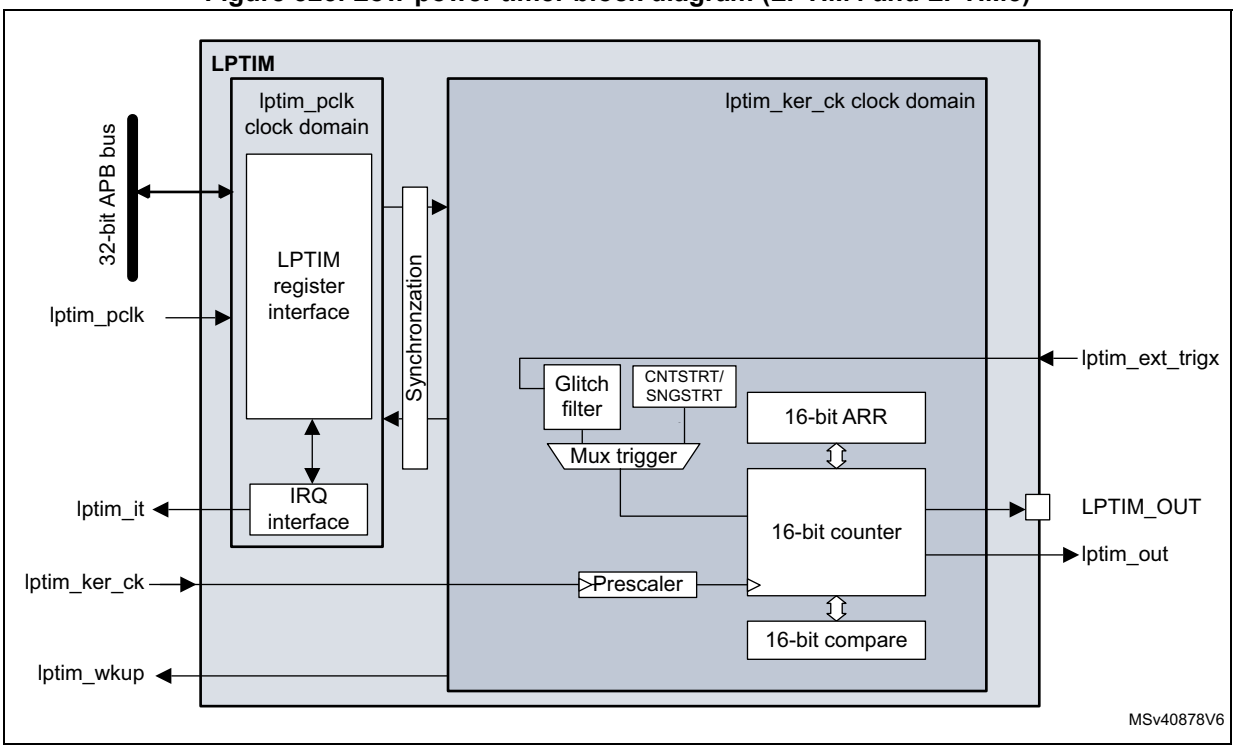


Figure 523. Low-power timer block diagram (LPTIM4 and LPTIM5)



42.4.2 LPTIM pins and internal signals

Table 343 and *Table 342* gives the list of LPTIM internal signals and pins, respectively.

Table 342. LPTIM input/output pins

Names	Signal type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM Output GPIO pin

Table 343. LPTIM internal signals

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wakeup event

42.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

Table 344. LPTIM1 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM1_ETR alternate function
lptim_ext_trig1	RTC_ALARMA
lptim_ext_trig2	RTC_ALARMB
lptim_ext_trig3	RTC_TAMP1_OUT
lptim_ext_trig4	RTC_TAMP2_OUT
lptim_ext_trig5	RTC_TAMP3_OUT

Table 344. LPTIM1 external trigger connection (continued)

TRIGSEL	External trigger
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

Table 345. LPTIM2 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM2_ETR alternate function
lptim_ext_trig1	RTC_ALARMA
lptim_ext_trig2	RTC_ALARMB
lptim_ext_trig3	RTC_TAMP1_OUT
lptim_ext_trig4	RTC_TAMP2_OUT
lptim_ext_trig5	RTC_TAMP3_OUT
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

Table 346. LPTIM3 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	LPTIM2_OUT
lptim_ext_trig1	Not connected
lptim_ext_trig2	LPTIM4_OUT
lptim_ext_trig3	LPTIM5_OUT
lptim_ext_trig4	SAI1_FS_A
lptim_ext_trig5	SAI1_FS_B
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 347. LPTIM4 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	LPTIM2_OUT
lptim_ext_trig1	LPTIM3_OUT
lptim_ext_trig2	Not connected
lptim_ext_trig3	LPTIM5_OUT
lptim_ext_trig4	SAI2_FS_A
lptim_ext_trig5	SAI2_FS_B

Table 347. LPTIM4 external trigger connection (continued)

TRIGSEL	External trigger
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 348. LPTIM5 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	LPTIM2_OUT
lptim_ext_trig1	LPTIM3_OUT
lptim_ext_trig2	LPTIM4_OUT
lptim_ext_trig3	SAI4_FS_A
lptim_ext_trig4	SAI4_FS_B
lptim_ext_trig5	Not connected
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 349. LPTIM1 input 1 connection

lptim_in1_mux	LPTIM1 input 1 connected to
lptim_in1_mux0	GPIO pin as LPTIM1_IN1 alternate function
lptim_in1_mux1	COMP1_OUT
lptim_in1_mux2	Not connected
lptim_in1_mux3	Not connected

Table 350. LPTIM1 input 2 connection

lptim_in2_mux	LPTIM1 input 2 connected to
lptim_int2_mux0	GPIO pin as LPTIM1_IN2 alternate function
lptim_in2_mux1	COMP2_OUT
lptim_in2_mux2	Not connected
lptim_in2_mux3	Not connected

Table 351. LPTIM2 input 1 connection

lptim_in1_mux	LPTIM2 input 1 connected to
lptim_in1_mux0	GPIO pin as LPTIM2_IN1 alternate function
lptim_in1_mux1	COMP1_OUT
lptim_in1_mux2	COMP2_OUT
lptim_in1_mux3	COMP1_OUT OR COMP2_OUT

Table 352. LPTIM2 input 2 connection

lptim_in2_mux	LPTIM2 input 2 connected to
lptim_int2_mux0	GPIO pin as LPTIM2_IN2 alternate function
lptim_in2_mux1	COMP2_OUT
lptim_in2_mux2	Not connected
lptim_in2_mux3	Not connected

Table 353. LPTIM3 input 1 connection

lptim_in1_mux	LPTIM3 Input 1 connected to
lptim_in1_mux0	Not connected
lptim_in1_mux1	SAI4_FS_A
lptim_in1_mux2	SAI4_FS_B
lptim_in1_mux3	Not connected

42.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be chosen among APB, LSI, LSE or HSI sources through the Reset and Clock controller (RCC). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM either from APB or any other embedded oscillator including LSE, LSI and HSI.
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM will use an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.

42.4.5 Glitch filter

The LPTIM inputs, either external (mapped to microcontroller GPIOs) or internal (mapped on the chip-level to other embedded peripherals, such as embedded comparators), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

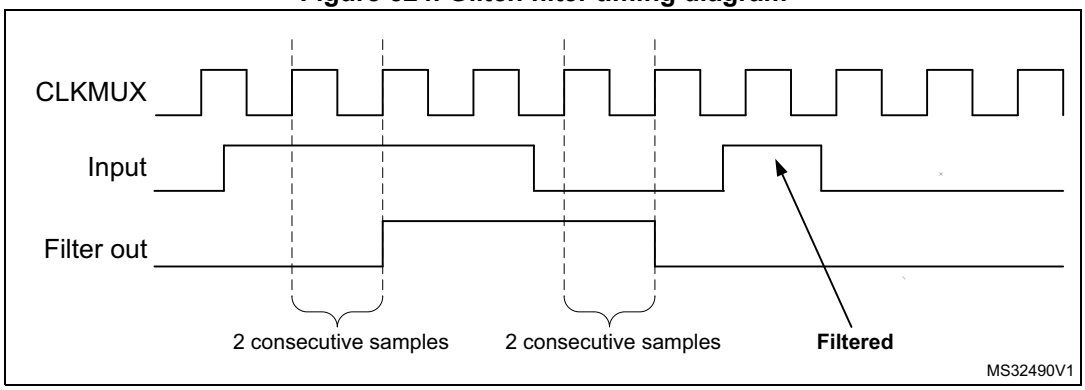
The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 524* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 524. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

42.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

Table 354. Prescaler division ratios

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

42.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it will be ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled will be discarded by hardware.

42.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when reaching the ARR value.

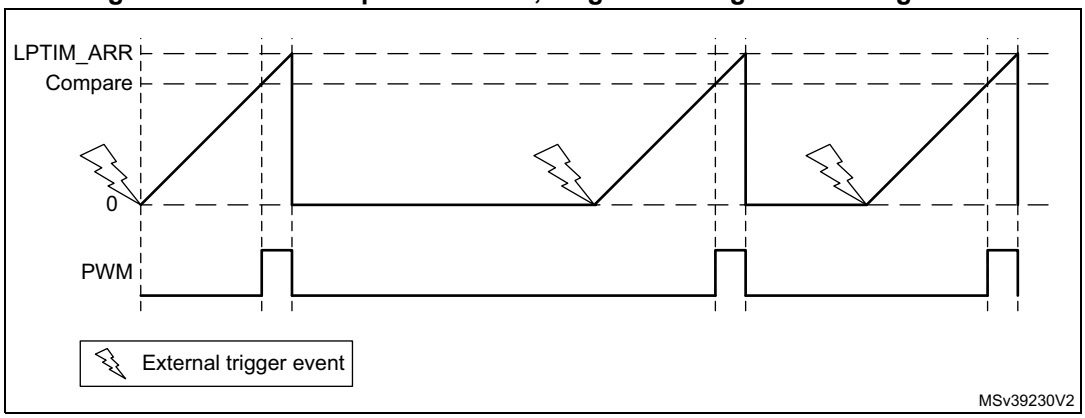
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in *Figure 525*.

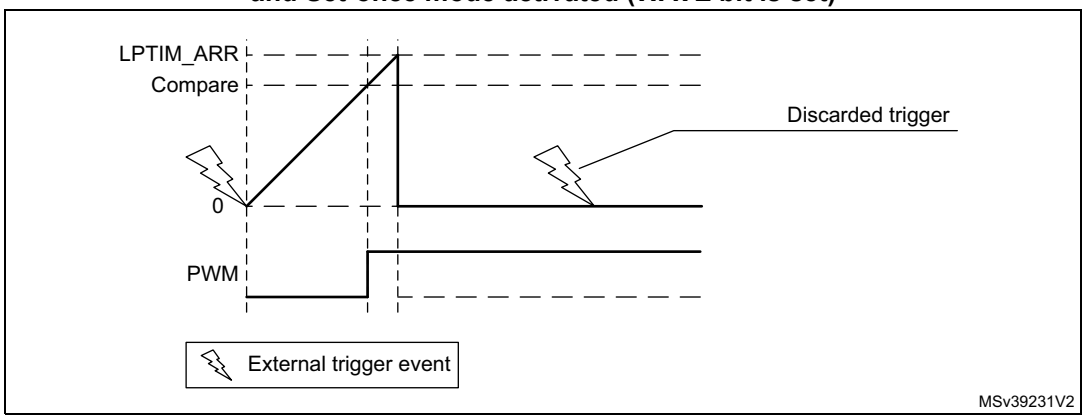
Figure 525. LPTIM output waveform, single counting mode configuration



- Set-once mode activated:

It should be noted that when the WAVE bit-field in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 526*.

Figure 526. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting will start the counter for one-shot counting.

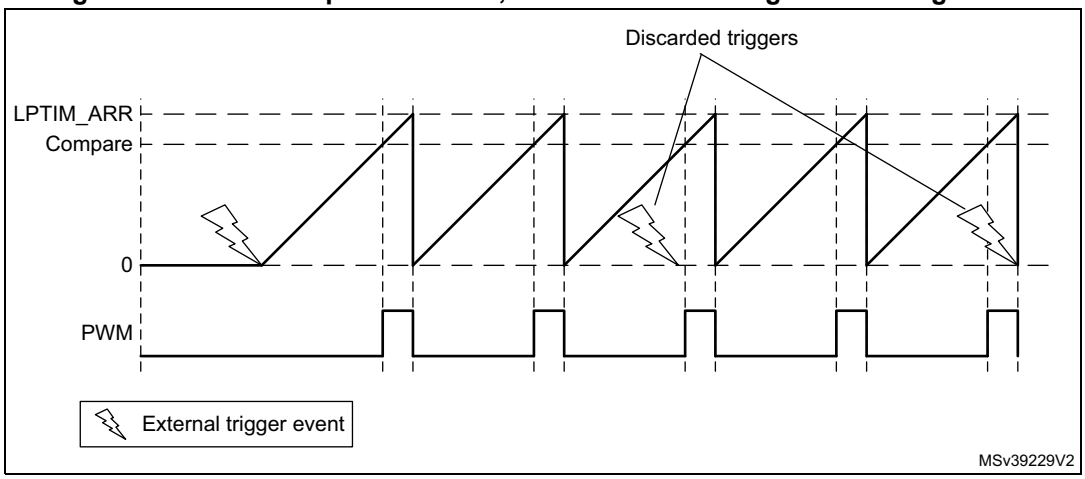
Continous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in *Figure 527*.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT will start the counter for continuous counting.

Figure 527. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.

42.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

42.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CMP (compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CMP register value.

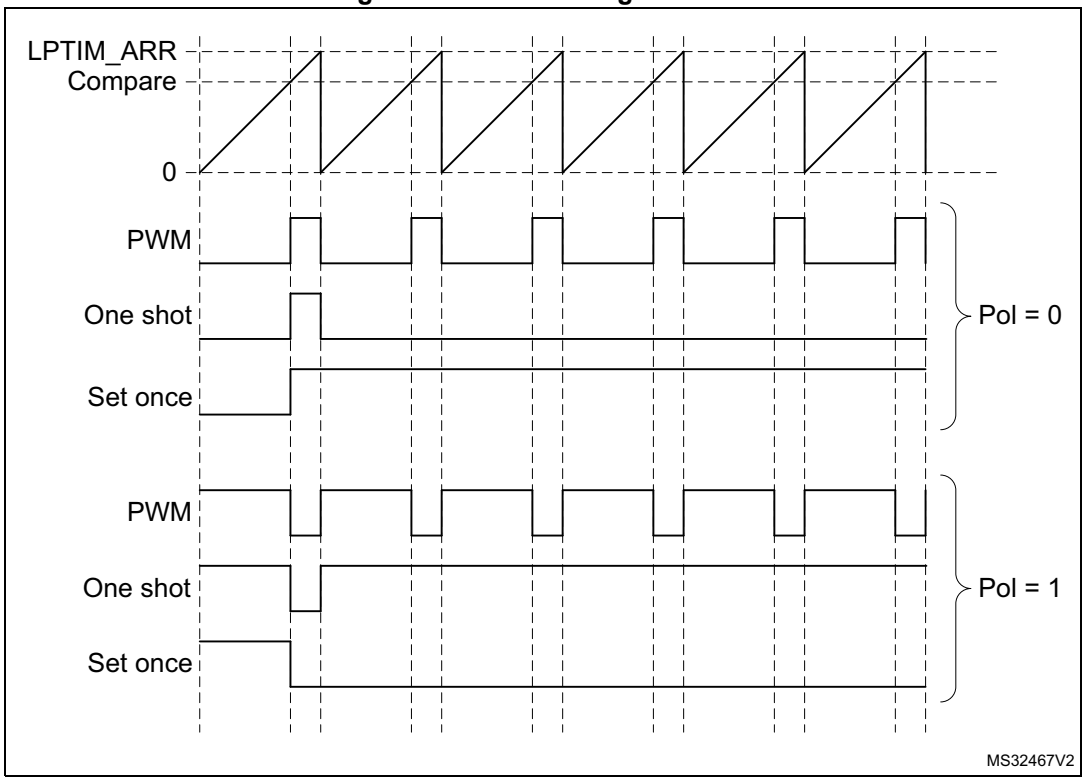
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. *Figure 528* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.

Figure 528. Waveform generation



42.4.11 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation, or at the end of the current period if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR and the LPTIM_CMP registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR and the LPTIM_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR and the LPTIM_CMP registers are updated at the end of the current period, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register and the LPTIM_CMP register.

After a write to the LPTIM_ARR register or the LPTIM_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, will lead to unpredictable results.

42.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source will be used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

42.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFGR and LPTIM_IER registers must be modified only when the LPTIM is disabled.

42.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bit-field to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic will elapse before the reset is taken into account. This will make the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.
- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register will reset its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle will occur on the LPTIM Input1.
  It should be noted that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.

42.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore you must configure LPTIM_ARR before starting. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

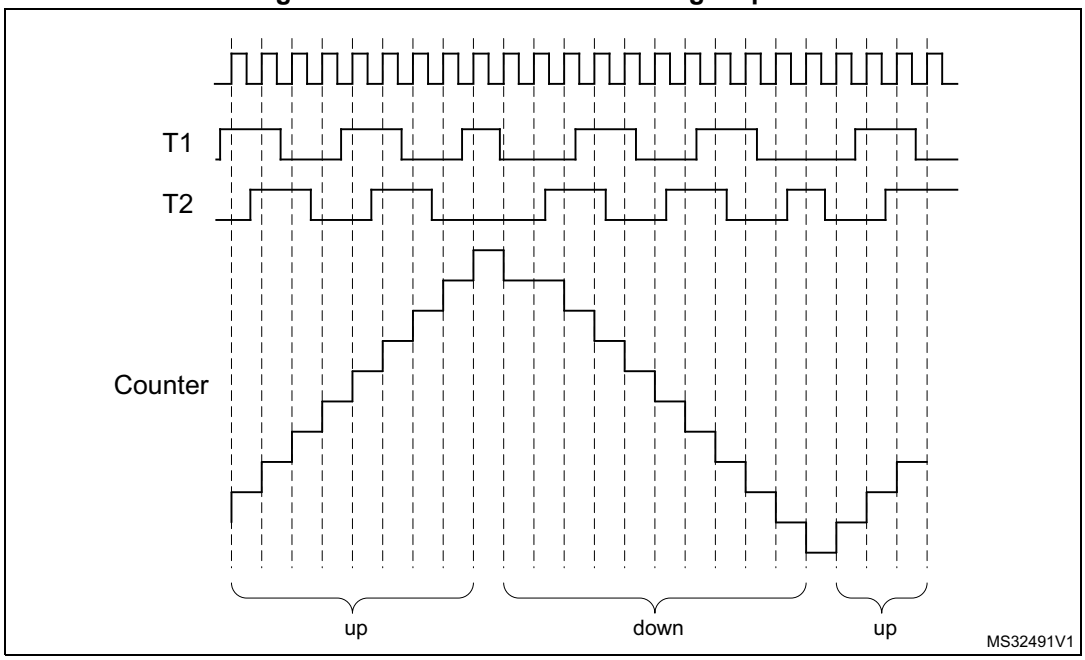
Table 355. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).

Figure 529. Encoder mode counting sequence



42.4.16 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

42.5 LPTIM low-power modes

Table 356. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	The LPTIM peripheral is active when it is clocked by LSE or LSI. LPTIM interrupts cause the device to exit Stop mode
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

42.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).

Note: If any bit in the LPTIM_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

Table 357. Interrupt events

Interrupt event	Description
Compare match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).
Auto-reload match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).
External trigger event	Interrupt flag is raised when an external trigger event is detected
Auto-reload register update OK	Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.
Compare register update OK	Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.
Direction change	Used in Encoder mode. Two interrupt flags are embedded to signal direction change: – UP flag signals up-counting direction change – DOWN flag signals down-counting direction change.

42.7 LPTIM registers

42.7.1 LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN	UP	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
									r	r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMPOK**: Compare register update OK

CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_CMP register has been successfully completed.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CMPM**: Compare match

The CMPM bit is set by hardware to inform application that LPTIM_CNT register value reached the LPTIM_CMP register's value.

42.7.2 LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
									w	w	w	w	w	w	w

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMPOKCF**: Compare register update OK clear flag
Writing 1 to this bit clears the CMPOK flag in the LPTIM_ISR register

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CMPMCF**: Compare match clear flag
Writing 1 to this bit clears the CMP flag in the LPTIM_ISR register

42.7.3 LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 Reserved, must be kept at reset value.

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable

0: DOWN interrupt disabled

1: DOWN interrupt enabled

Bit 5 **UPIE**: Direction change to UP Interrupt Enable

0: UP interrupt disabled

1: UP interrupt enabled

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
- 0: ARROK interrupt disabled
- 1: ARROK interrupt enabled

Bit 3 **CMPOKIE**: Compare register update OK Interrupt Enable
- 0: CMPOK interrupt disabled
- 1: CMPOK interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
- 0: EXTTRIG interrupt disabled
- 1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
- 0: ARRM interrupt disabled
- 1: ARRM interrupt enabled

Bit 0 **CMPMIE**: Compare match Interrupt Enable
- 0: CMPM interrupt disabled
- 1: CMPM interrupt enabled

Caution: The LPTIM_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’)

42.7.4 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode
- 0: Encoder mode disabled
- 1: Encoder mode enabled

Bit 23 **COUNTMODE**: counter mode enabled

The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
- 0: the counter is incremented following each internal clock pulse
- 1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode

The PRELOAD bit controls the LPTIM_ARR and the LPTIM_CMP registers update modality

0: Registers are updated after each APB bus write access

1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVEPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CMP registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CMP registers

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

0: Deactivate Set-once mode, PWM / One Pulse waveform (depending on OPMODE bit)

1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started will be ignored

1: A trigger event arriving when the timer is already started will reset and restart the counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: software trigger (counting start is initiated by software)

01: rising edge is the active edge

10: falling edge is the active edge

11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0

001: lptim_ext_trig1

010: lptim_ext_trig2

011: lptim_ext_trig3

100: lptim_ext_trig4

101: lptim_ext_trig5

110: lptim_ext_trig6

111: lptim_ext_trig7

See *Section 42.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger
01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition

01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock Polarity

If LPTIM is clocked by an external clock source:

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: the rising edge is the active edge used for counting

01: the falling edge is the active edge used for counting

10: both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four time the external clock frequency.

11: not allowed

If the LPTIM is configured in Encoder mode (ENC bit is set):

00: the encoder sub-mode 1 is active

01: the encoder sub-mode 2 is active

10: the encoder sub-mode 3 is active

Refer to *Section 42.4.15: Encoder mode* for more details about Encoder mode sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM will use:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

42.7.5 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											w	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register will asynchronously reset LPTIM_CNT register content.

Caution: This bitfield is write-only. This means that the bit cannot be read back to verify the value which has been written. As an example, if this bit is set to 1, attempting to read it back will return 0 even if the "Reset after read" function is enabled (due to the fact that this bitfield has previously been written to 1). To turn off the "Reset after read" or to make sure that it has already been turned off, this bit should be reset (by programming it to 0) even if it already contains 0.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit will trigger a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in Continuous mode.
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.
If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.
This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in single pulse mode.
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.
If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM_ARR and LPTIM_CNT registers.
This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.
0:LPTIM is disabled
1:LPTIM is enabled

42.7.6 LPTIM compare register (LPTIM_CMP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP[15:0]:** Compare value
CMP is the compare value used by the LPTIM.

Caution: The LPTIM_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

42.7.7 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CMP[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

42.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value

When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

It should be noted that for a reliable LPTIM_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.

42.7.9 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]:** LPTIM input 2 selection

The IN2SEL bits control the LPTIM Input 2 multiplexer, which connect LPTIM Input 2 to one of the available inputs.

00: lptim_in2_mux0

01: lptim_in2_mux1

10: lptim_in2_mux2

11: lptim_in2_mux3

For connection details refer to *Section 42.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]:** LPTIM input 1 selection
The IN1SEL bits control the LPTIM Input 1 multiplexer, which connects LPTIM Input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 42.4.3: LPTIM input and trigger mapping*.

Caution: The LPTIM_CFGR2 register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

42.7.10 LPTIM3 configuration register 2 (LPTIM3_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN1SEL[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM3 Input1 selection
The IN1SEL bits control the LPTIM3 Input 1 multiplexer, which connects LPTIM3 Input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Table 353: LPTIM3 input 1 connection*.

Caution: The LPTIM3_CFGR2 registers must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

42.7.11 LPTIM register map

The following table summarizes the LPTIM registers.

Table 358. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN	UP	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
	Reset value																										0	0	0	0	0	0	0
0x004	LPTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
	Reset value																										0	0	0	0	0	0	0
0x008	LPTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
	Reset value																										0	0	0	0	0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNTMODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0		0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																											0	0	0	0	0
0x014	LPTIM_CMP	CMP[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x018	LPTIM_ARR	ARR[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1																
0x01C	LPTIM_CNT	CNT[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x024	LPTIM_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value																											0	0			0	0

Table 358. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	LPTIM3_ CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN1SEL	
	Reset value																															0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

43 System window watchdog (WWDG)

43.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. A reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

43.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see *Figure 531*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

43.3 WWDG functional description

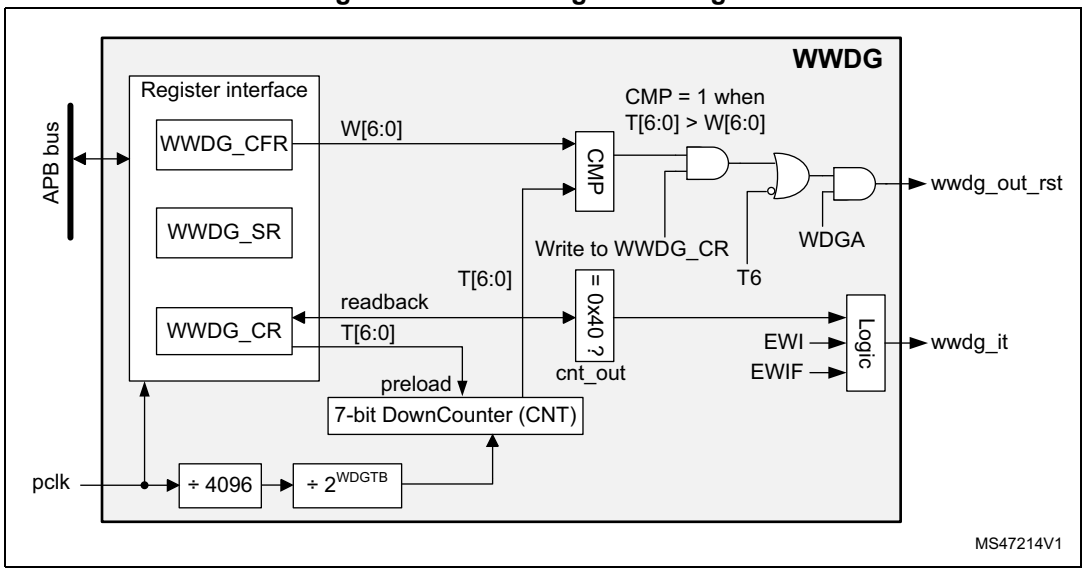
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation must occur only when the counter value is lower than the window register value and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 530* and to *Section 43.3.2: WWDG internal signals* for the WWDG block diagram.

43.3.1 WWDG block diagram

Figure 530. Watchdog block diagram



43.3.2 WWDG internal signals

Table 359 gives the list of WWDG internal signals.

Table 359. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

43.3.3 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

43.3.4 Controlling the downcounter

This downcounter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 531*). The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. *Figure 531* describes the window watchdog process.

Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

43.3.5 Advanced watchdog interrupt feature

The early wakeup interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

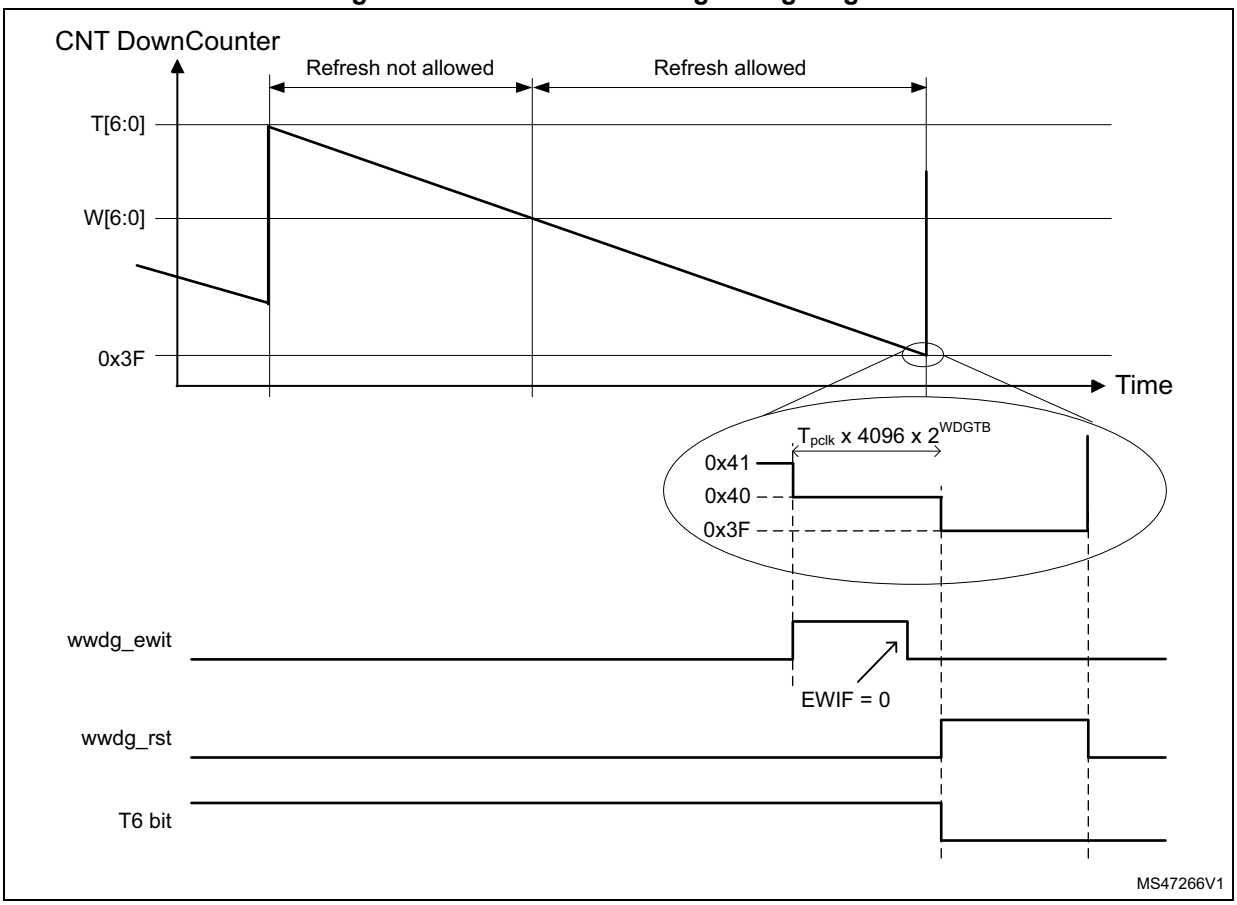
Note: When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset is eventually generated.

43.3.6 How to program the watchdog timeout

Use the formula in *Figure 531* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 531. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0]+1) \quad (ms)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK}: APB clock period measured in ms

4096: value corresponding to internal divider

As an example, lets assume APB frequency is equal to 48 MHz, WDGTB[2:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63+1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of the t_{WWDG}.

43.3.7 Debug mode

When the CPU enters debug mode, WWDG1 counter either continues to work normally or stops, depending on DBGMCU_APB3LFZ1. For more details, refer to *Section 59: Debug infrastructure*.

43.4 WWDG registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

43.4.1 Control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

43.4.2 Configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:
000: CK Counter Clock (PCLK div 4096) div 1
001: CK Counter Clock (PCLK div 4096) div 2
010: CK Counter Clock (PCLK div 4096) div 4
011: CK Counter Clock (PCLK div 4096) div 8
100: CK Counter Clock (PCLK div 4096) div 16
101: CK Counter Clock (PCLK div 4096) div 32
110: CK Counter Clock (PCLK div 4096) div 64
111: CK Counter Clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the downcounter.

43.4.3 Status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0’. Writing ‘1’ has no effect. This bit is also set if the interrupt is not enabled.

43.4.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 360. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**WWDG_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	**WWDG_CFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	**WWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

44 Independent watchdog (IWDG)

44.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to *Section 43 on page 1873*.

44.2 IWDG main features

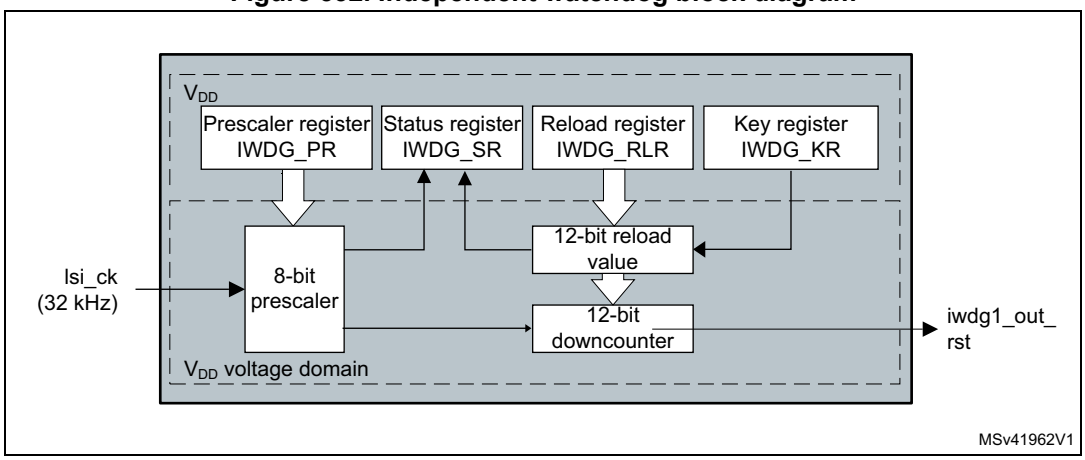
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional Reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

44.3 IWDG functional description

44.3.1 IWDG block diagram

Figure 532 shows the functional blocks of the independent watchdog module.

Figure 532. Independent watchdog block diagram



1. The watchdog function is implemented in the V_{DD} voltage domain that is still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the *Key register (IWDG_KR)*, the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the *Key register (IWDG_KR)*, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

44.3.2 IWDG internal signals

Table 361 gives the list of IWDG internal signals.

Table 361. IWDG internal input/output signals

Signal name	Signal type	Description
lsi_ck	Digital input	LSI clock
iwdg1_out_rst	Digital output	IWDG1 reset signal output

44.3.3 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *Window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *Window register (IWDG_WINR)*, then a reset is provided.

The default value of the *Window register (IWDG_WINR)* is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the *Reload register (IWDG_RLR)* value and ease the cycle number calculation to generate the next reload.

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *Key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *Key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *Prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *Reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Write to the *Window register (IWDG_WINR)*. This automatically refreshes the counter value in the *Reload register (IWDG_RLR)*.

Note: Writing the window value allows to refresh the Counter value by the RLR when Status register (IWDG_SR) is set to 0x0000 0000.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *Key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *Key register (IWDG_KR)*.
3. Write the prescaler by programming the *Prescaler register (IWDG_PR)* from 0 to 7.
4. Write the *Reload register (IWDG_RLR)*.
5. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
6. Refresh the counter value with IWDG_RLR (IWDG_KR = 0x0000 AAAA).

44.3.4 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the *Key register (IWDG_KR)* is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

44.3.5 Low-power freeze

Depending on the IWDG_FZ_STOP and IWDG_FZ_STBY options configuration, the IWDG can continue counting or not during the Stop mode and the Standby mode respectively. If the IWDG is kept running during Stop or Standby modes, it can wake up the device from this mode. Refer to *Section : User and read protection option bytes* for more details.

44.3.6 Behavior in Stop and Standby modes

Once running, the IWDG cannot be stopped.

44.3.7 Register access protection

Write access to *Prescaler register (IWDG_PR)*, *Reload register (IWDG_RLR)* and *Window register (IWDG_WINR)* is protected. To modify them, the user must first write the code 0x0000 5555 in the *Key register (IWDG_KR)*. A write access to this register with a different value will break the sequence and register access will be protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is on going.

44.3.8 Debug mode

When the microcontroller enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module.

44.4 IWDG registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

44.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG_PR, IWDG_RLR and IWDG_WINR registers (see *Section 44.3.7: Register access protection*)

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

44.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 44.3.7: Register access protection*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *Status register (IWDG_SR)* must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the Status register (IWDG_SR) is reset.

44.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Register access protection*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *Key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the *Status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the Status register (IWDG_SR) is reset.

44.4.4 Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five RC 40 kHz cycles).

Window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to five RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to five RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.

44.4.5 Window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 44.3.7*, they contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the *Status register (IWDG_SR)* must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the Status register (IWDG_SR) is reset.

44.4.6 IWDG register map

The following table gives the IWDG register map and reset values.

Table 362. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WVU	RVU	PVU
	Reset value																														0	0	0
0x10	**IWDG_WINR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

45 Real-time clock (RTC)

45.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After Backup domain reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

45.2 RTC main features

The RTC unit main features are the following (see *Figure 534: Detailed RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wakeup interrupt
  - Time-stamp
  - Tamper detection
- 32 backup registers.

45.3 RTC functional description

45.3.1 RTC block diagram

Figure 533. RTC block overview

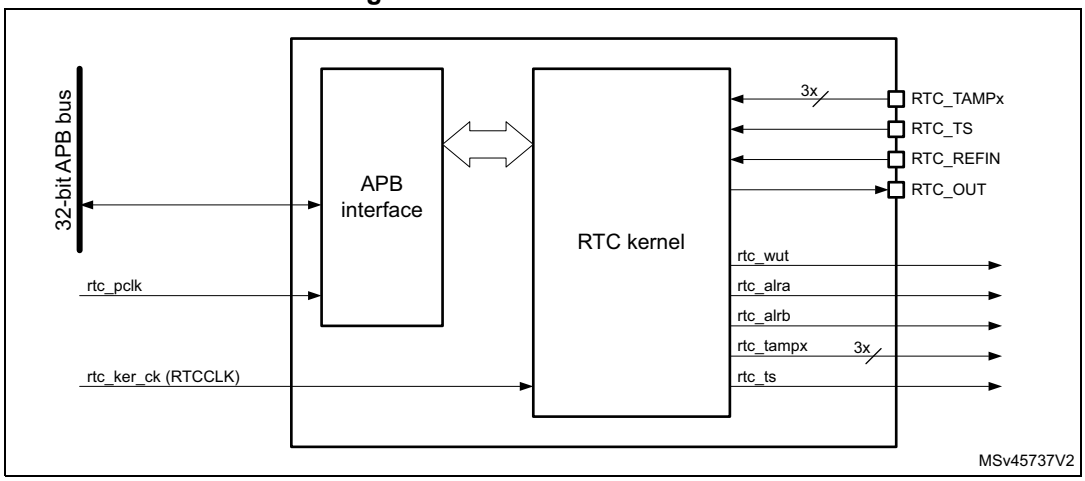


Figure 534. Detailed RTC block diagram

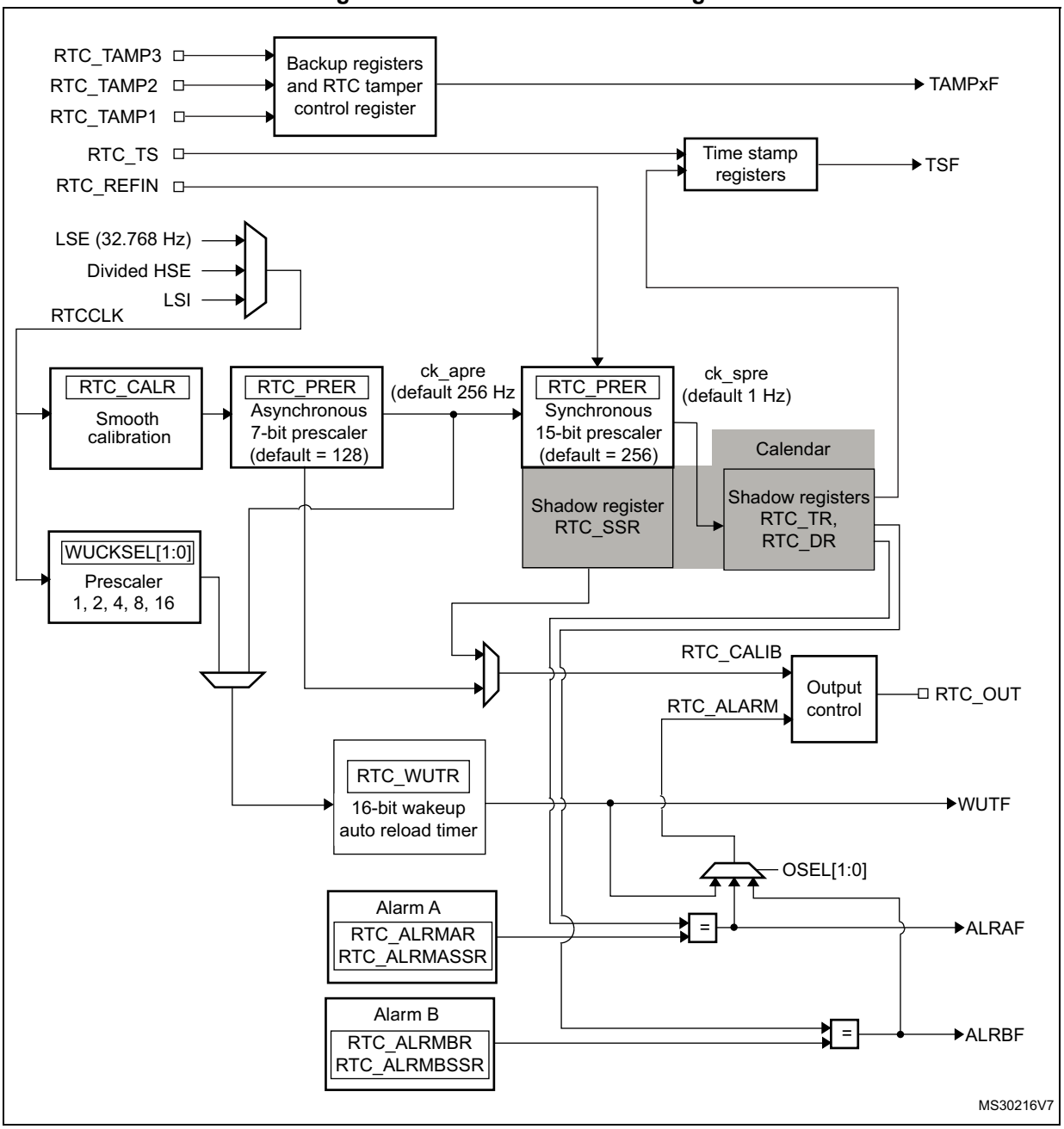
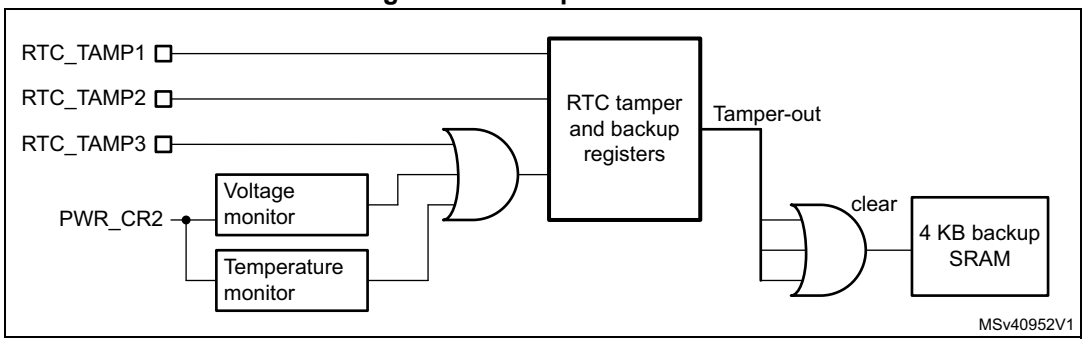


Figure 535. Tamper detection



The RTC includes:

- Two alarms
- Three tamper events from I/Os
  - Tamper detection erases the backup registers and the backup RAM.
  - In addition, the tamper detection forbids software access to the backup SRAM until its erase operation is finished. Refer to *Section 45.3.15: Tamper detection*
  - The tamper3 event detection is generated either by an event on I/O, or by an over or under voltage of the RTC power supply domain, or by an over or under temperature detection. These voltage and temperature monitor detections are enabled in the *PWR control register 2 (PWR_CR2)*.
- One timestamp event from I/O
- Tamper event detection can generate a timestamp event
- Timestamp can be generated when a switch to V_{BAT} occurs
- 32 x 32-bit backup registers
  - The backup registers (RTC_BKPxR) are implemented in the RTC domain that remains powered-on by VBAT when the VDD power is switched off.
- Output functions: RTC_OUT which selects one of the following two outputs:
  - RTC_CALIB: 512 Hz or 1Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
  - RTC_ALARM: This output is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the Alarm A, Alarm B or Wakeup outputs.
- Input functions:
  - RTC_TS: timestamp event
  - RTC_TAMP1: tamper1 event detection
  - RTC_TAMP2: tamper2 event detection
  - RTC_TAMP3: tamper3 event detection
  - RTC_REFIN: 50 or 60 Hz reference clock input

45.3.2 RTC pins and internal signals

Table 363. RTC pins and internal signals

Signal name	Signal type	Description
RTC_TS	Input	Timestamp input
RTC_TAMPx (x = 1,2,3)	Input	Tamper input

Table 363. RTC pins and internal signals (continued)

Signal name	Signal type	Description
RTC_REFIN	Input	Reference clock input
RTC_OUT	Output	RTC output
rtc_ker_ck *(RTCCLK)*	Internal input	RTC clock source (LSE clock, LSI clock and HSE clock)
rtc_pclk	Internal input	RTC APB interface clock
rtc_wut	Internal output	RTC wakeup event output for on chip peripherals
rtc_alra	Internal output	RTC Alarm A event output for on chip peripherals
rtc_alrb	Internal output	RTC Alarm B event output for on chip peripherals
rtc_tampx	Internal output	RTC Tamper[1..3] event output for on chip peripherals
rtc_ts	Internal output	RTC Timestamp event output for on chip peripherals

45.3.3 GPIOs controlled by the RTC

RTC_OUT, RTC_TS and RTC_TAMP1 are mapped on the same pin (PC13). PC13 pin configuration is controlled by the RTC, whatever the PC13 GPIO configuration, except for the RTC_ALARM output open-drain mode. The RTC functions mapped on PC13 are available in all low-power modes and in VBAT mode.

The output mechanism follows the priority order shown in *Table 364*.

Table 364. RTC pin PC13 configuration(1)

PC13 Pin configuration and function	OSEL[1:0] bits (RTC_ALARM output enable)	COE bit (RTC_CALIB output enable)	RTC_OUT_RMP bit	RTC_ALARM_TYPE bit	TAMP1E bit (RTC_TAMP1 input enable)	TSE bit (RTC_TS input enable)
RTC_ALARM output OD	01 or 10 or 11	Don't care	0	0	Don't care	Don't care
			1			
RTC_ALARM output PP	01 or 10 or 11	Don't care	0	1	Don't care	Don't care
			1			
RTC_CALIB output PP	00	1	0	Don't care	Don't care	Don't care
RTC_TAMP1 input floating	00	0	Don't care	Don't care	1	0
	00	1	1			
	01 or 10 or 11	0				
RTC_TS and RTC_TAMP1 input floating	00	0	Don't care	Don't care	1	1
	00	1	1			
	01 or 10 or 11	0				

Table 364. RTC pin PC13 configuration(1) (continued)

PC13 Pin configuration and function	OSEL[1:0] bits (RTC_ALARM output enable)	COE bit (RTC_CALIB output enable)	RTC_OUT_RMP bit	RTC_ALARM_TYPE bit	TAMP1E bit (RTC_TAMP1 input enable)	TSE bit (RTC_TS input enable)
RTC_TS input floating	00	0	Don't care	Don't care	0	1
	00	1	1			
	01 or 10 or 11	0				
Wakeup pin or Standard GPIO	00	0	Don't care	Don't care	0	0
	00	1	1			
	01 or 10 or 11	0				

1. OD: open drain; PP: push-pull.

In addition, it is possible to remap RTC_OUT on PB2 pin thanks to RTC_OUT_RMP bit. In this case it is mandatory to configure PB2 GPIO registers as alternate function with the correct type. The remap functions are shown in *Table 365*.

Table 365. RTC_OUT mapping

OSEL[1:0] bits (RTC_ALARM output enable)	COE bit (RTC_CALIB output enable)	RTC_OUT_RMP bit	RTC_OUT on PC13	RTC_OUT on PB2
00	0	0	-	-
00	1		RTC_CALIB	-
01 or 10 or 11	Don't care		RTC_ALARM	-
00	0	1	-	-
00	1		-	RTC_CALIB
01 or 10 or 11	0		-	RTC_ALARM
01 or 10 or 11	1		RTC_ALARM	RTC_CALIB

The table below summarizes the RTC pins and functions capability in all modes.

Table 366. RTC functions over modes

Pin	RTC functions	Functional in all low-power modes except Standby modes	Functional in Standby mode	Functional in VBAT mode
PC13	RTC_TAMP1 RTC_TS RTC_OUT	YES	YES	YES
PI8	RTC_TAMP2	YES	YES	YES
PC1	RTC_TAMP3	YES	YES	YES

Table 366. RTC functions over modes (continued)

Pin	RTC functions	Functional in all low-power modes except Standby modes	Functional in Standby mode	Functional in VBAT mode
PB2	RTC_OUT	YES	NO	NO
PB15	RTC_REFIN	YES	NO	NO

45.3.4 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to *Section 7: Reset and Clock Control (RCC)*.

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 534: Detailed RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 45.3.7: Periodic auto-wakeup* for details).

45.3.5 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK period, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see *Section 45.6.4: RTC initialization and status register (RTC_ISR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 1 RTCCLK period.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD=0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

45.3.6 Programmable alarms

The RTC unit provides programmable alarm: Alarm A and Alarm B. The description below is given for Alarm A, but can be translated in the same way for Alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register. The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register. The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and Alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the RTC_ALARM output. RTC_ALARM output polarity can be configured through bit POL the RTC_CR register.

45.3.7 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE(32.768kHz), this allows to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value.When the initialization sequence is complete (see *Programming the wakeup timer on page 1898*), the timer starts counting down.When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. RTC_ALARM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

45.3.8 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR_CR1 register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC_TAMPCR, RTC_BKPxR, RTC_OR and RTC_ISR[13:8].

1. Write ‘0xCA’ into the RTC_WPR register.
2. Write ‘0x53’ into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: After a system reset, the application can read the INITS flag in the RTC_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ISR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for Alarm A but can be translated in the same way for Alarm B.

1. Clear ALRAE in RTC_CR to disable Alarm A.
2. Program the Alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable Alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clock cycles after WUTE is cleared, due to clock synchronization.

45.3.9 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK period: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 1898*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 45.3.11: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting

from low-power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.

45.3.10 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the RTC tamper configuration register (RTC_TAMPCR), the RTC backup registers (RTC_BKPxR), the wakeup timer register (RTC_WUTR), the Alarm A and Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR), and the Option register (RTC_OR).

In addition, when it is clocked by the LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to the RTC clock section of the Reset and clock controller for details on the list of RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

45.3.11 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the

SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON=1.

45.3.12 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the synchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

45.3.13 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2]=1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8]=1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to '1' effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Calibration when PREDIV_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other

interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the

calibrated input clock is as follows:

$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8- second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ISR/INITF=0, by using the follow process:

1. Poll the RTC_ISR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

45.3.14 Time-stamp function

Time-stamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a time-stamp event is detected on the RTC_TS pin.

When ITSE is set:

The calendar is saved in the time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal time-stamp event is detected. The internal timestamp event is generated by the switch to the VBAT supply.

When a time-stamp event occurs, due to internal or external event, the time-stamp flag bit (TSF) in RTC_ISR register is set. In case the event is internal, the ITSF flag is also set in RTC_ISR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a time-stamp event occurs.

If a new time-stamp event is detected while the time-stamp flag (TSF) is already set, the time-stamp overflow flag (TSOVF) flag is set and the time-stamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set 2 ck_apre cycles after the time-stamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two time-stamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a time-stamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set.To avoid masking a time-stamp event occurring at the same moment, the application must not write ‘0’ into TSF bit unless it has already read it to ‘1’.

Optionally, a tamper event can cause a time-stamp to be recorded. See the description of the TAMPTS control bit in *Section 45.6.16: RTC tamper configuration register (RTC_TAMPCR)*.

45.3.15 Tamper detection

The RTC_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for the following purposes:

- erase the RTC backup registers and backup SRAM (default configuration)
- generate an interrupt, capable to wakeup from Stop and Standby modes
- generate a hardware trigger for the low-power timers

RTC backup registers

The backup registers (RTC_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs (see *Section 45.6.20: RTC backup registers (RTC_BKPxR)* and *Tamper detection initialization on page 1905*)

except if the TAMPxNOERASE bit is set, or if TAMPxMF is set in the RTC_TAMPCR register.

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC_TAMPCR register.

Each RTC_TAMPx tamper detection input is associated with a flag TAMPxF in the RTC_ISR register.

When TAMPxMF is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (Level detection with filtering)
- 3 ck_apre cycles when TAMPTS=1 (Timestamp on tamper event)
- No latency when TAMPFLT=0x0 (Edge detection) and TAMPTS=0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMF is set:

A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck_rtc additional cycles.

By setting the TAMPIE bit in the RTC_TAMPCR register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPIE is not allowed when one or more TAMPxMF is set.

When TAMPIE is cleared, each tamper pin event interrupt can be individually enabled by setting the corresponding TAMPxIE bit in the RTC_TAMPCR register. Setting TAMPxIE is not allowed when the corresponding TAMPxMF is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMF bit in cleared in RTC_TAMPCR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMF bit is set, the TAMPxF flag is masked, and kept cleared in RTC_ISR register. This configuration allows to trig automatically the low-power timers in Stop mode, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

Timestamp on tamper event

With TAMPTS set to ‘1’, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set at the same time that TSF or TSOVF is set.

Edge detection on tamper inputs

If the TAMPFLT bits are “00”, the RTC_TAMPx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding

TAMPxTRG bit. The internal pull-up resistors on the RTC_TAMPx inputs are deactivated when edge detection is selected.

Caution: When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.
When TAMPFLT="00" and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the RTC_TAMPx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC_BKPxR). This prevents the application from writing to the backup registers while the RTC_TAMPx input value still indicates a tamper detection. This is equivalent to a level detection on the RTC_TAMPx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the RTC_TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on RTC_TAMPx inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The RTC_TAMPx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1,The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the RTC_TAMPx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the datasheets for the electrical characteristics of the pull-up resistors.

45.3.16 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the RTC_CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz. The 1 Hz output is affected when a shift operation is on going and may toggle during the shift operation (SHPF=1).

Note: When the RTC_CALIB or RTC_ALARM output is selected, the RTC_OUT pin is automatically configured as output.

When COSEL bit is cleared, the RTC_CALIB output is the output of the 6th stage of the asynchronous prescaler.

When COSEL bit is set, the RTC_CALIB output is the output of the 8th stage of the synchronous prescaler.

45.3.17 Alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output RTC_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Alarm output

The RTC_ALARM pin can be configured in output open drain or output push-pull using the control bit RTC_ALARM_TYPE in the RTC_OR register.

Note: Once the RTC_ALARM output is enabled, it has priority over RTC_CALIB (COE bit is don't care and must be kept cleared).

When the RTC_CALIB or RTC_ALARM output is selected, the RTC_OUT *pin is automatically configured as output.*

45.4 RTC low-power modes

Table 367. Effect of low-power modes on RTC

Mode	Description
Stop	Peripheral registers content is kept.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode.

45.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to *Section 19: Extended interrupt and event controller (EXTI)*.

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Alarm event in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC_ALARM IRQ channel in the NVIC.
3. Configure the RTC to generate RTC alarms.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Tamper event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC TimeStamp event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC time-stamp event.

To enable the Wakeup timer interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the Wakeup timer even in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC_WKUP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC Wakeup timer event.

Table 368. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop mode	Exit from Standby mode
Alarm A	ALRAF	ALRAIE	yes	yes(1)	yes(1)
Alarm B	ALRBF	ALRBIE	yes	yes(1)	yes(1)
RTC_TS input (timestamp)	TSF	TSIE	yes	yes(1)	yes(1)
RTC_TAMP1 input detection	TAMP1F	TAMPIE	yes	yes(1)	yes(1)
RTC_TAMP2 input detection	TAMP2F	TAMPIE	yes	yes(1)	yes(1)
RTC_TAMP3 input detection	TAMP3F	TAMPIE	yes	yes(1)	yes(1)
Wakeup timer interrupt	WUTF	WUTIE	yes	yes(1)	yes(1)

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.

45.6 RTC registers

Refer to *Section 1.2 on page 97* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

45.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1898* and *Reading the calendar on page 1899*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31-23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

45.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1898* and *Reading the calendar on page 1899*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

45.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
							rw	rw	rw	rw	rw	rw	rw	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ITSE**: timestamp on internal event enable
0: internal event timestamp disabled
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable
This bit enables the RTC_CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to RTC_ALARM output
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wakeup output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of RTC_ALARM output
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection
When COE=1, this bit selects which signal is output on RTC_CALIB.
0: Calibration output is 512 Hz (with default prescaler setting)
1: Calibration output is 1 Hz (with default prescaler setting)
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A=127 and PREDIV_S=255). Refer to *Section 45.3.16: Calibration clock output*

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)

When this bit is set, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change outside initialization mode.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change outside initialization mode.

Bit 15 **TSIE**: Time-stamp interrupt enable

0: Time-stamp Interrupt disable

1: Time-stamp Interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable

0: Wakeup timer interrupt disabled

1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: *Alarm B interrupt enable*

0: Alarm B Interrupt disable

1: Alarm B Interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable

1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable

0: Wakeup timer disabled

1: Wakeup timer enabled

Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.

Bit 9 **ALRBE**: *Alarm B enable*

0: Alarm B disabled

1: Alarm B enabled

Bit 8 **ALRAE:** Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC_REFIN detection disabled

1: RTC_REFIN detection enabled

Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Time-stamp event active edge

0: RTC_TS input rising edge generates a time-stamp event

1: RTC_TS input falling edge generates a time-stamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck_spre (usually 1 Hz) clock is selected

11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value (see note below)

Note: Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).

WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

This register is write protected. The write access procedure is described in RTC register write protection on page 1897.

Caution: TSE must be reset when TSEDGE is changed to avoid spuriously setting of TSF.

45.6.4 RTC initialization and status register (RTC_ISR)

This register is write protected (except for RTC_ISR[13:8] bits). The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to '0'

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	RECALPF
														rc_w0	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value

Bit 17 **ITSF**: Internal tTime-stamp flag

This flag is set by hardware when a time-stamp on the internal event occurs.

This flag is cleared by software by writing 0, and must be cleared together with TSF bit by writing 0 in both bits.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to '1' when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to '0'. Refer to *Re-calibration on-the-fly*.

Bit 15 **TAMP3F**: RTC_TAMP3 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP3 input.

It is cleared by software writing 0

Bit 14 **TAMP2F**: RTC_TAMP2 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP2 input.

It is cleared by software writing 0

Bit 13 **TAMP1F**: RTC_TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP1 input.

It is cleared by software writing 0

Bit 12 **TSOVF**: Time-stamp overflow flag

This flag is set by hardware when a time-stamp event occurs while TSF is already set.

This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF**: Time-stamp flag

This flag is set by hardware when a time-stamp event occurs.

This flag is cleared by software by writing 0. If ITSF flag is set, TSF must be cleared together with ITSF by writing 0 in both bits.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.

This flag is cleared by software by writing 0.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 9 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm B register (RTC_ALRMBR).

This flag is cleared by software by writing 0.

Bit 8 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the Alarm A register (RTC_ALRMAR).

This flag is cleared by software by writing 0.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

0: No shift operation is pending

1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware up to 2 RTCCLK cycles after the WUTE bit has been set to 0 in RTC_CR, and is cleared up to 2 RTCCLK cycles after the WUTE bit has been set to 1. The wakeup timer values can be changed when WUTE bit is cleared and WUTWF is set.

0: Wakeup timer configuration update not allowed

1: Wakeup timer configuration update allowed

Bit 1 **ALRBWF**: Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm B update not allowed

1: Alarm B update allowed

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

Note: The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.

45.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 1898*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor
This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor
This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

45.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ISR.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.

45.6.7 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don't care in Alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format.

Bits 27:24 **DU[3:0]**: Date units or day in BCD format.

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in Alarm A comparison

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in Alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in Alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

45.6.8 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don’t care in Alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don’t care in Alarm B comparison

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don’t care in Alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don’t care in Alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

45.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

45.6.10 RTC sub second register (RTC_SSR)

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

45.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value

Bits 14:0 **SUBFS**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC_ISR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

45.6.12 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value

Bit 22 **PM:** AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

45.6.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value

Bits 15:13 **WDU[1:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

45.6.14 RTC time-stamp sub second register (RTC_TSSSR)

The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

45.6.15 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*.

Address offset: 0x3C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 * CALP) - CALM.

Refer to *Section 45.3.13: RTC smooth digital calibration*.

Bit 14 **CALW8:** Use an 8-second calibration cycle period

When CALW8 is set to '1', the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at "00" when CALW8='1'. Refer to Section 45.3.13: RTC smooth digital calibration.

Bit 13 **CALW16:** Use a 16-second calibration cycle period

When CALW16 is set to '1', the 16-second calibration cycle period is selected.This bit must not be set to '1' if CALW8=1.

Note: CALM[0] is stuck at '0' when CALW16='1'. Refer to Section 45.3.13: RTC smooth digital calibration.

Bits 12:9 Reserved, must be kept at reset value

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 45.3.13: RTC smooth digital calibration on page 1902*.

45.6.16 RTC tamper configuration register (RTC_TAMPCR)

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP3NOERASE	TAMP3IE	TAMP2MF	TAMP2NOERASE	TAMP2IE	TAMP1MF	TAMP1NOERASE	TAMP1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	TAMP3TRG	TAMP3E	TAMP2TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TAMP3MF**: Tamper 3 mask flag

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and the backup SRAM are not erased.

Note: The Tamper 3 interrupt must not be enabled when TAMP3MF is set.

Bit 23 **TAMP3NOERASE**: Tamper 3 no erase

0: Tamper 3 event erases the backup registers and the backup SRAM.

1: Tamper 3 event does not erase the backup registers and the backup SRAM.

Bit 22 **TAMP3IE**: Tamper 3 interrupt enable

0: Tamper 3 interrupt is disabled if TAMPIE = 0.

1: Tamper 3 interrupt enabled.

Bit 21 **TAMP2MF**: Tamper 2 mask flag

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and the backup SRAM are not erased.

Note: The Tamper 2 interrupt must not be enabled when TAMP2MF is set.

Bit 20 **TAMP2NOERASE**: Tamper 2 no erase

0: Tamper 2 event erases the backup registers and the backup SRAM.

1: Tamper 2 event does not erase the backup registers and the backup SRAM.

Bit 19 **TAMP2IE**: Tamper 2 interrupt enable

0: Tamper 2 interrupt is disabled if TAMPIE = 0.

1: Tamper 2 interrupt enabled.

Bit 18 **TAMP1MF**: Tamper 1 mask flag

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware.The backup registers and the backup SRAM are not erased.

Note: The Tamper 1 interrupt must not be enabled when TAMP1MF is set.

Bit 17 **TAMP1NOERASE**: Tamper 1 no erase
0: Tamper 1 event erases the backup registers and the backup SRAM.
1: Tamper 1 event does not erase the backup registers and the backup SRAM.

Bit 16 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt is disabled if TAMPIE = 0.
1: Tamper 1 interrupt enabled.

Bit 15 **TAMPPUDIS**: RTC_TAMPx pull-up disable
This bit determines if each of the RTC_TAMPx pins are precharged before each sample.
0: Precharge RTC_TAMPx pins before sampling (enable internal pull-up)
1: Disable precharge of RTC_TAMPx pins.

Bits 14:13 **TAMPPRCH[1:0]**: RTC_TAMPx precharge duration
These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the RTC_TAMPx inputs.
0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 12:11 **TAMPFLT[1:0]**: RTC_TAMPx filter count
These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a Tamper event. TAMPFLT is valid for each of the RTC_TAMPx inputs.
0x0: Tamper event is activated on edge of RTC_TAMPx input transitions to the active level (no internal pull-up on RTC_TAMPx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency
Determines the frequency at which each of the RTC_TAMPx inputs are sampled.
0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Bit 7 **TAMPTS**: Activate timestamp on tamper detection event
0: Tamper detection event does not cause a timestamp to be saved
1: Save timestamp on tamper detection event
TAMPTS is valid even if TSE=0 in the RTC_CR register.

Bit 6 **TAMP3TRG:** Active level for RTC_TAMP3 input
if TAMPFLT ≠ 00:
0: RTC_TAMP3 input staying low triggers a tamper detection event.
1: RTC_TAMP3 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP3 input rising edge triggers a tamper detection event.
1: RTC_TAMP3 input falling edge triggers a tamper detection event.

Bit 5 **TAMP3E**: RTC_TAMP3 detection enable
0: RTC_TAMP3 input detection disabled
1: RTC_TAMP3 input detection enabled

Bit 4 **TAMP2TRG**: Active level for RTC_TAMP2 input
if TAMPFLT != 00:
0: RTC_TAMP2 input staying low triggers a tamper detection event.
1: RTC_TAMP2 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP2 input rising edge triggers a tamper detection event.
1: RTC_TAMP2 input falling edge triggers a tamper detection event.

Bit 3 **TAMP2E**: RTC_TAMP2 input detection enable
0: RTC_TAMP2 detection disabled
1: RTC_TAMP2 detection enabled

Bit 2 **TAMPIE**: Tamper interrupt enable
0: Tamper interrupt disabled
1: Tamper interrupt enabled.
Note: This bit enables the interrupt for all tamper pins events, whatever TAMPxIE level. If this bit is cleared, each tamper event interrupt can be individually enabled by setting TAMPxIE.

Bit 1 **TAMP1TRG**: Active level for RTC_TAMP1 input
If TAMPFLT != 00
0: RTC_TAMP1 input staying low triggers a tamper detection event.
1: RTC_TAMP1 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP1 input rising edge triggers a tamper detection event.
1: RTC_TAMP1 input falling edge triggers a tamper detection event.

Bit 0 **TAMP1E**: RTC_TAMP1 input detection enable
0: RTC_TAMP1 detection disabled
1: RTC_TAMP1 detection enabled

Caution: When TAMPFLT = 0, TAMPxE must be reset when TAMPxTRG is changed to avoid spuriously setting TAMPxF.

45.6.17 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1897*

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.
2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.
3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.
...
12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.
13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.
14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.
15: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits23:15 Reserved, must be kept at reset value.

Bits 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

45.6.18 RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register is write protected.The write access procedure is described in *Section : RTC register write protection*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
0x1: SS[14:1] are don't care in Alarm B comparison. Only SS[0] is compared.
0x2: SS[14:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.
0x3: SS[14:3] are don't care in Alarm B comparison. Only SS[2:0] are compared.
...
0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared.
0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared.
0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared.
0xF: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

45.6.19 RTC option register (RTC_OR)

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_OUT_RMP	RTC_ALARM_TYPE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RTC_OUT_RMP**: RTC_OUT remap

Setting this bit allows to remap the RTC outputs on PB2 as follows:
RTC_OUT_RMP = '0':
If OSEL/= '00': RTC_ALARM is output on PC13
If OSEL= '00' and COE = '1': RTC_CALIB is output on PC13
RTC_OUT_RMP = '1':
If OSEL /= '00' and COE = '0': RTC_ALARM is output on PB2
If OSEL = '00' and COE = '1': RTC_CALIB is output on PB2
If OSEL /= '00' and COE = '1': RTC_CALIB is output on PB2 and RTC_ALARM is output on PC13.

Bit 0 **RTC_ALARM_TYPE**: RTC_ALARM output type on PC13

This bit is set and cleared by software
0: RTC_ALARM, when mapped on PC13, is open-drain output
1: RTC_ALARM, when mapped on PC13, is push-pull output

45.6.20 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0xCC

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers.
They are powered-on by V_{BAT} when V_{DD} is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.
This register is reset on a tamper detection event, as long as TAMPxF=1.

45.6.21 RTC register map

Table 369. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	**RTC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0		
	Reset value								0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0C	**RTC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	RECALPF	TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUT WF	ALRBWF	ALRAWF
	Reset value															0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x10	**RTC_PRER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	**RTC_ALRMAR**	MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**RTC_ALRMBR**	MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK2	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**RTC_WPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_SSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**RTC_SHIFTR**	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0

Table 369. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	**RTC_TSDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	**RTC_TSSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**RTC_ CALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0
0x40	**RTC_TAMPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP3NOERASE	TAMP3IE	TAMP2MF	TAMP2NOERASE	TAMP2IE	TAMP1MF	TAMP1NOERASE	TAMP1IE	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	TAMP3TRG	TAMP3E	TAMP2TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**RTC_ ALRMASSR**	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RTC_ ALRMBSSR**	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**RTC_ OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_OUT_RMP	RTC_ALARM_TYPE
	Reset value																															0	0
0x50 to 0xCC	**RTC_BKP0R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	to **RTC_BKP31R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

46 Inter-integrated circuit (I2C) interface

46.1 Introduction

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It provides multimaster capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

46.2 I2C main features

- I^2C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see *Section 46.3: I2C implementation*):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the i2c_pclk reprogramming
- Wakeup from Stop mode on address match.

46.3 I2C implementation

This manual describes the full set of features implemented in I2C peripheral. In the STM32H7xxx devices I2C1, I2C2, I2C3 and I2C4 implement the full set of features as shown in the following table.

Table 370. STM32H743/753 and STM32H750 I2C implementation

I2C features(1)	I2C1	I2C2	I2C3	I2C4
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Wakeup from Stop mode	X	X	X	X
SMBus/PMBus	X	X	X	X

1. X = supported.

46.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2C bus.

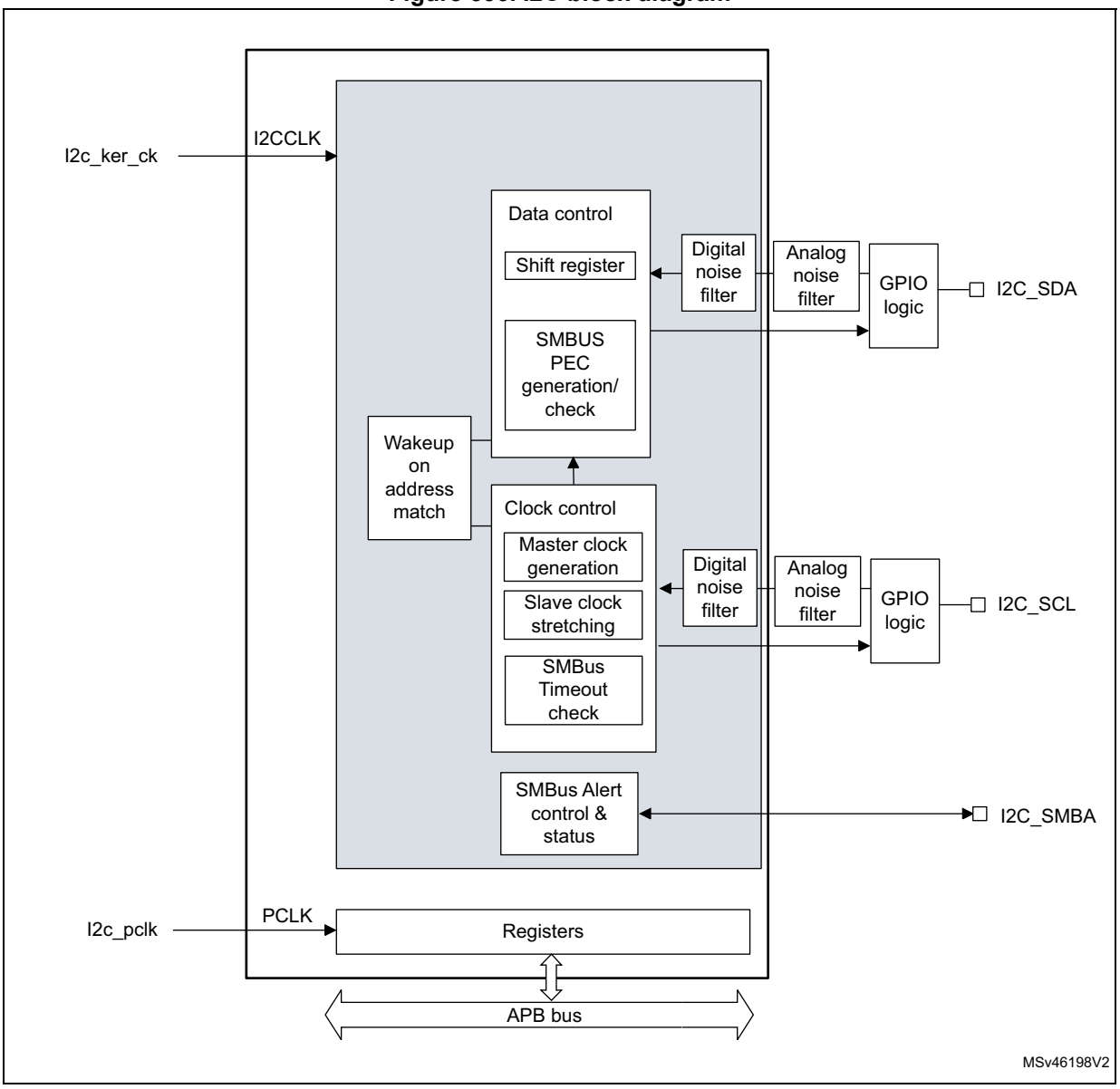
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

46.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 536*.

Figure 536. I2C block diagram



The I2C is clocked by an independent clock source which allows to the I2C to operate independently from the i2c_pclk frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). *Refer to Section 46.3: I2C implementation.*

46.4.2 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$ and $t_{I2CCLK} < t_{HIGH}$

with:

t_{LOW}: SCL low time and t_{HIGH}: SCL high time

$t_{filters:}$ when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is DNF x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the following condition:

$t_{PCLK} < 4/3\ t_{SCL}$

with t_{SCL}: SCL period

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

46.4.3 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

Communication flow

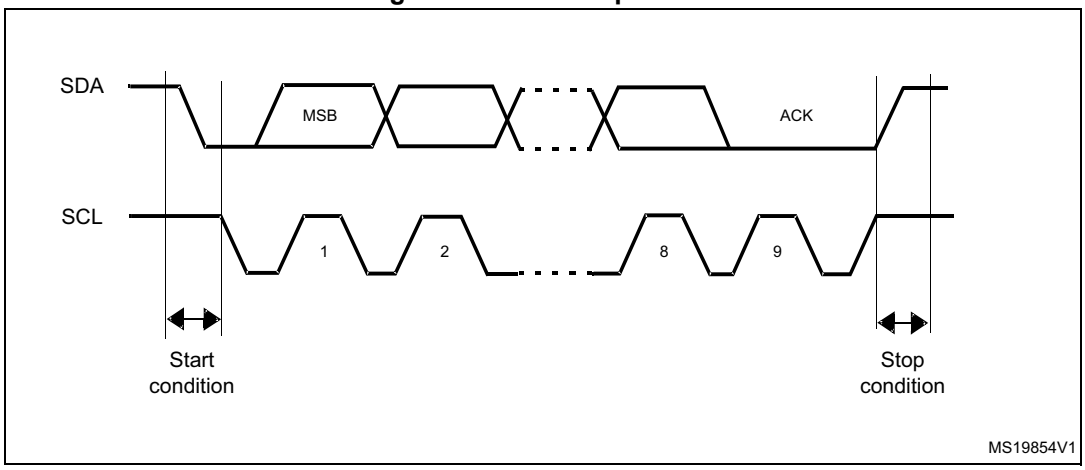
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 537. I²C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

46.4.4 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller.

Then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE=0), the I²C performs a software reset. Refer to *Section 46.4.5: Software reset* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I²C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x i2c_ker_ck periods. This allows to suppress spikes with a programmable length of 1 to 15 i2c_ker_ck periods.

Table 371. Comparison of analog vs. digital filters

-	Analog filter	Digital filter
Pulse width of suppressed spikes	≥ 50 ns	Programmable length from 1 to 15 I2C peripheral clocks
Benefits	Available in Stop mode	– Programmable length: extra filtering capability vs. standard requirements – Stable length
Drawbacks	Variation vs. temperature, voltage, process	Wakeup from Stop mode on address match is not available when digital filter is enabled

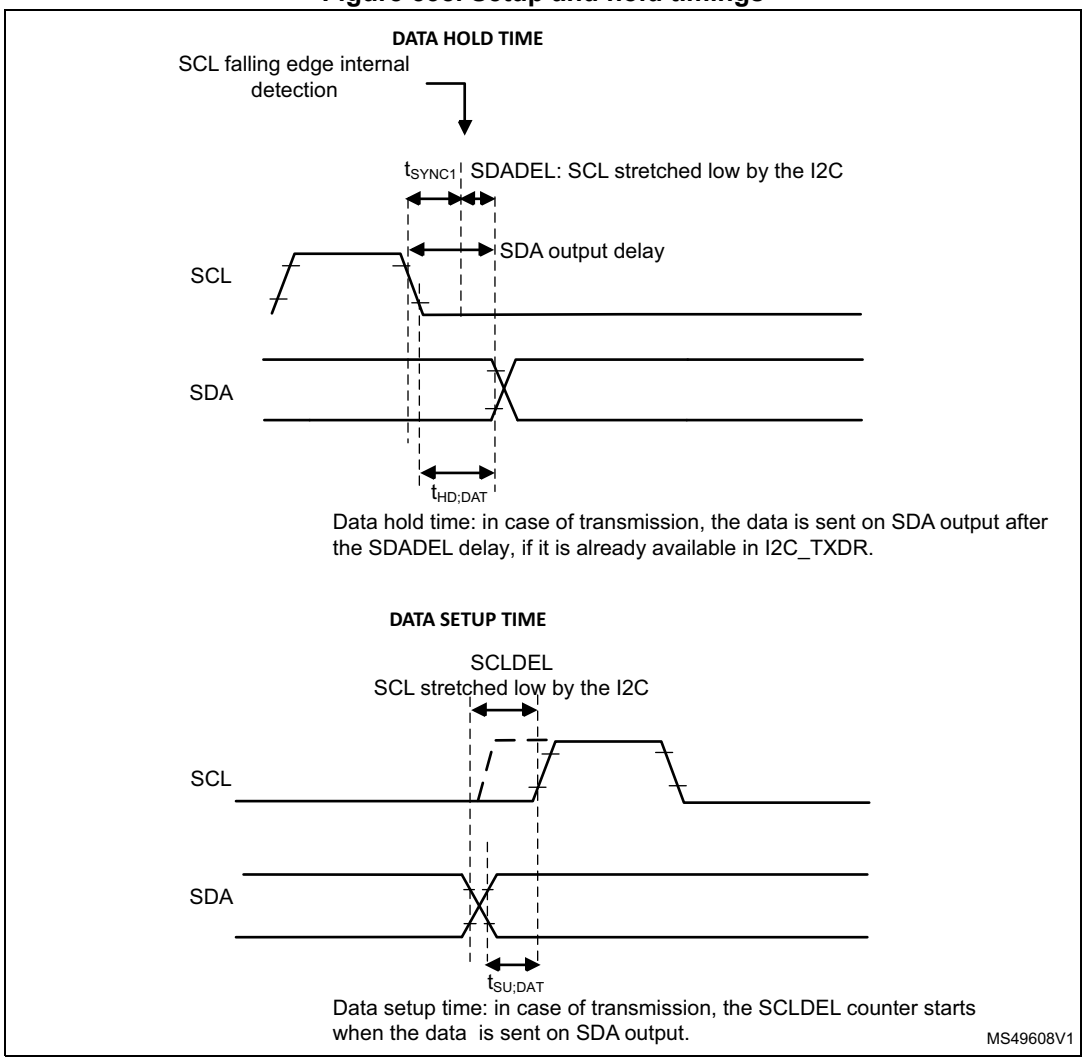
Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window

Figure 538. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  T_{SDADEL} impacts the hold time $t_{HD;DAT}$.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$ ns.
- When enabled, input delay brought by the digital filter: t_{DNF} = DNF x t_{I2CCLK}
- Delay due to SCL synchronization to i2c_ker_ck clock (2 to 3 i2c_ker_ck periods)

In order to bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: *$t_{AF(min)}$ / $t_{AF(max)}$ are part of the equation only when the analog filter is enabled. Refer to device datasheet for t_{AF} values.*

The maximum $t_{HD;DAT}$ can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$.

Note: *This condition can be violated when NOSTRETCH=0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.*

Refer to *Table 372: I2C-SMBUS specification data setup and hold times* for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

- After t_{SDADEL} delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is t_{SCLDEL} = (SCLDEL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 <= SCLDEL$

Refer to *Table 372: I2C-SMBUS specification data setup and hold times* for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Note: At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, in case the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH=1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Table 372. I^2C-SMBUS specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is $t_{SCLL} = (SCLL+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$.
  t_{SCLL} impacts the SCL low time t_{LOW} .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH} = (SCLH+1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLH} impacts the SCL high time t_{HIGH} .

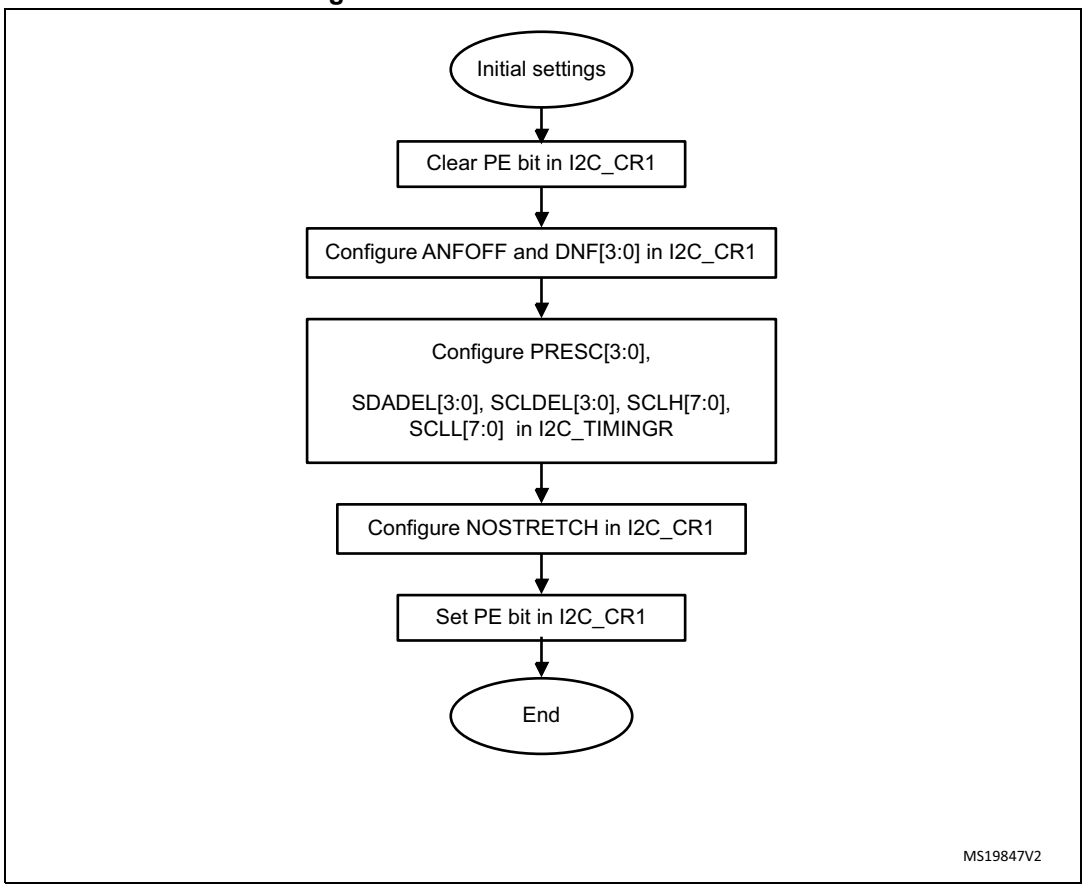
Refer to *I2C master initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C slave initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 539. I2C initialization flowchart



46.4.5 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1.

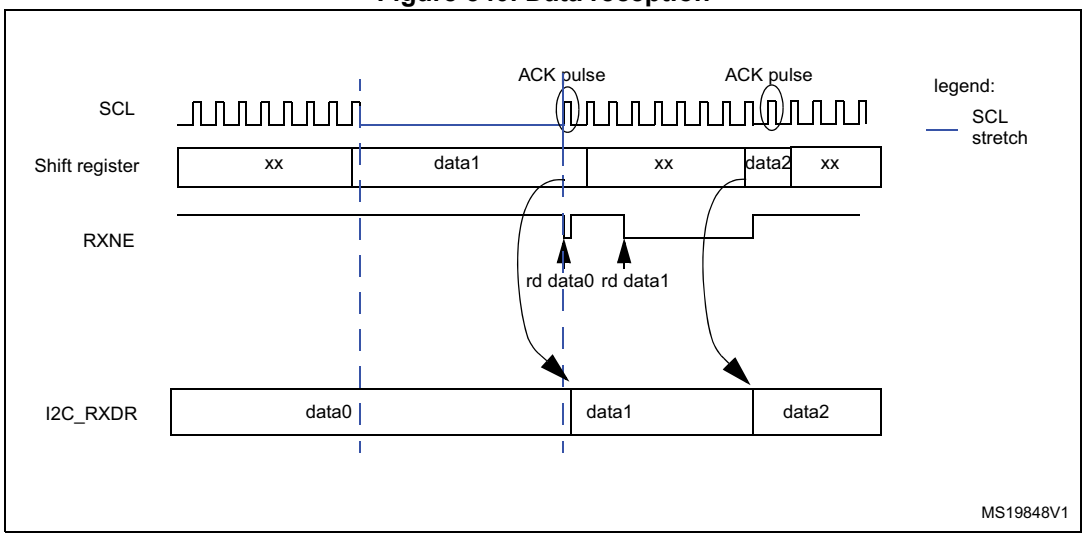
46.4.6 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

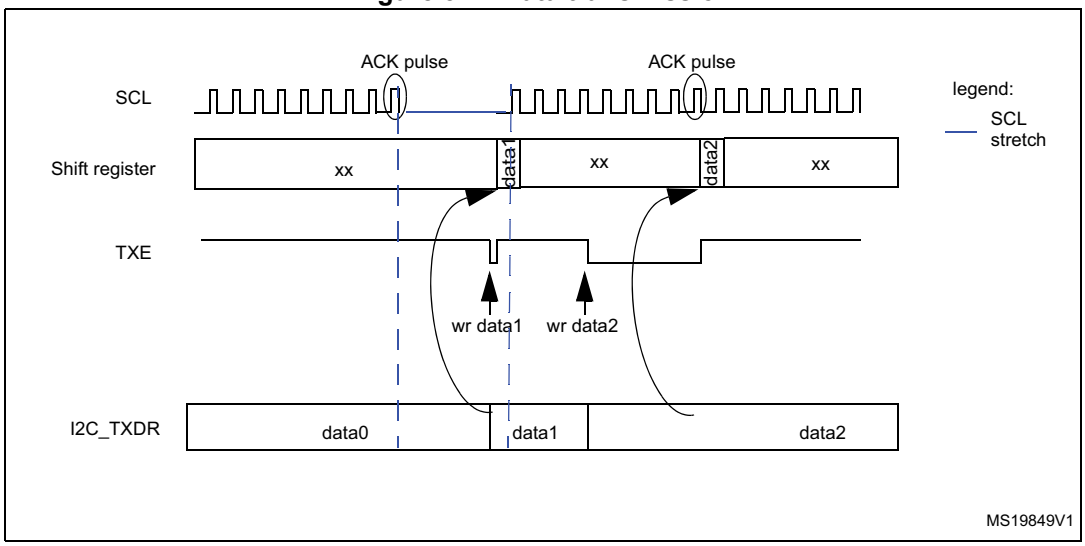
Figure 540. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 541. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2C_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = '1' in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = '0' in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 373. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Master Tx/Rx NBYTES + STOP	x	0	1
Master Tx/Rx + NBYTES + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

46.4.7 I2C slave mode

I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional slave addresses are required, the 2nd slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE=1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}.

Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Slave Byte Control mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

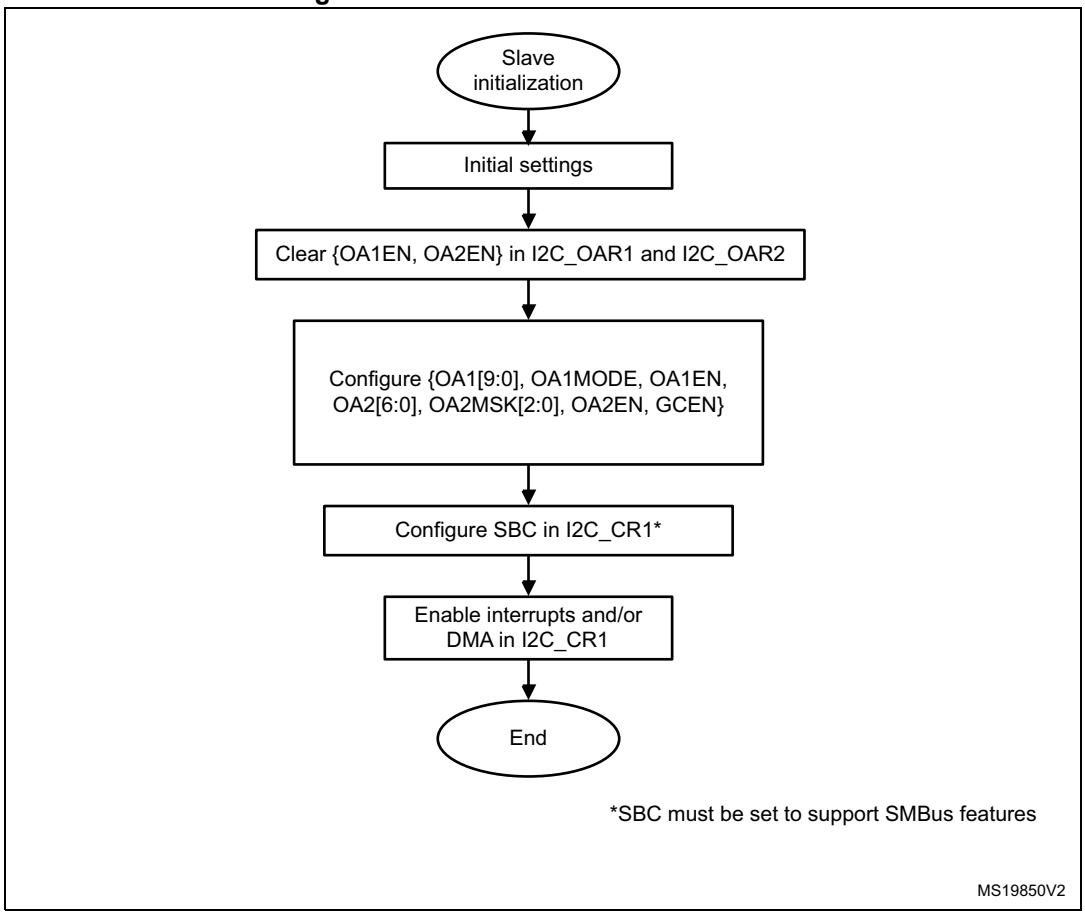
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Caution: Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

Figure 542. Slave initialization flowchart



Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, If TXE = 0 when the slave address is received (ADDR=1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event is needed, (Transmit Interrupt or Transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 543. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0

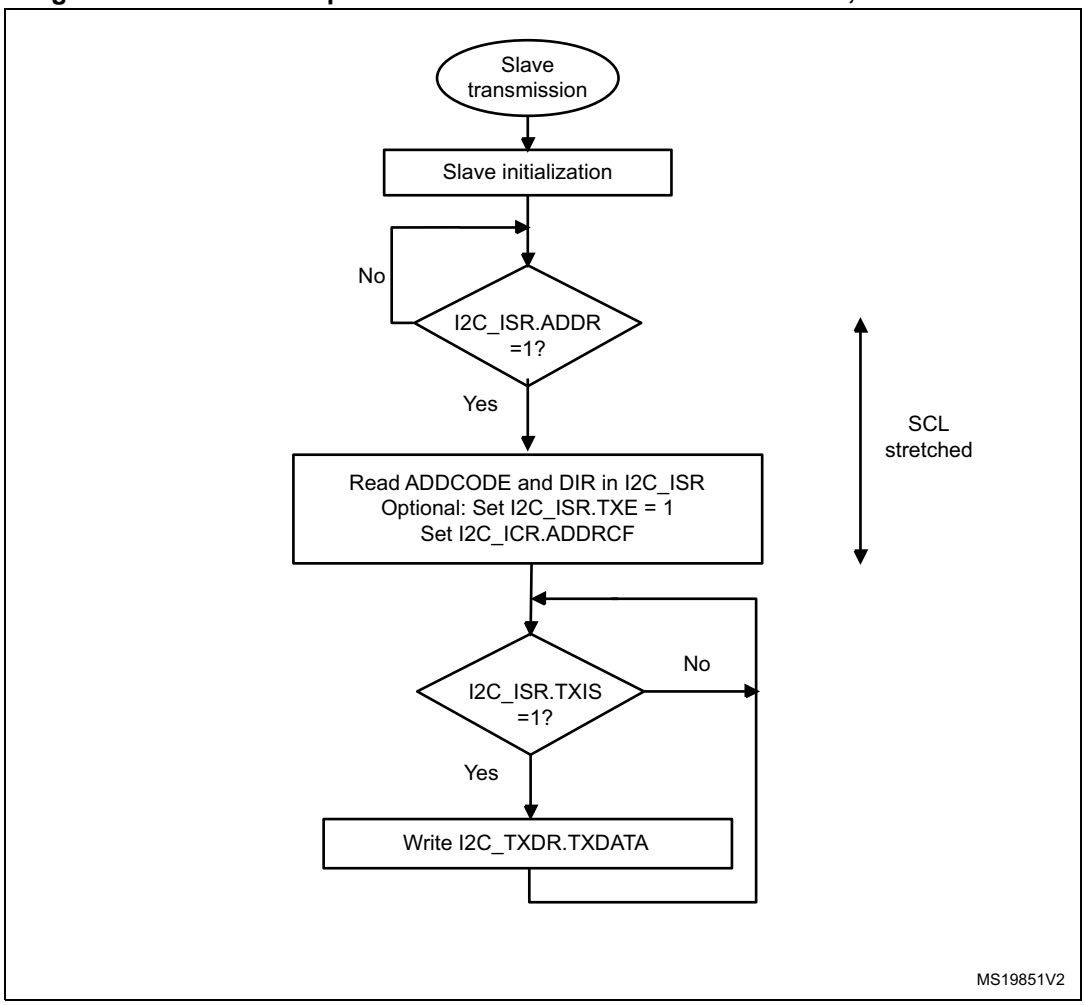


Figure 544. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1

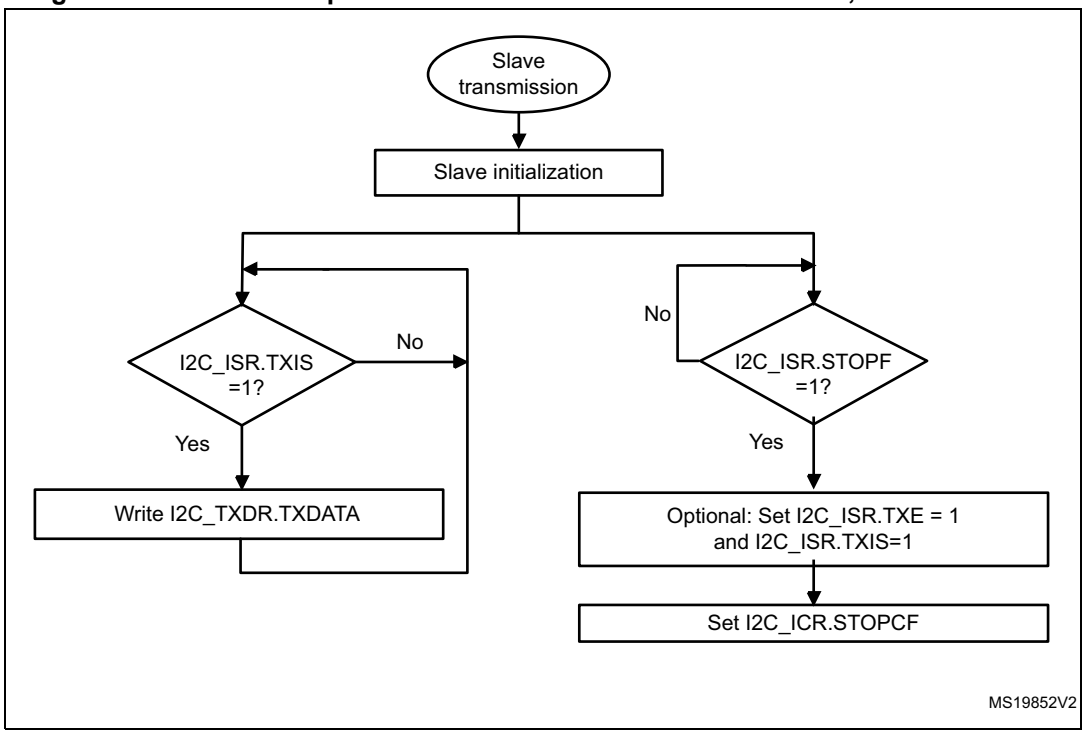
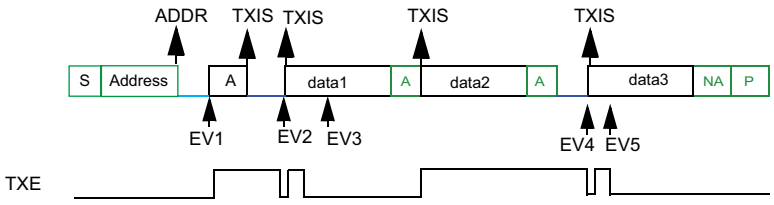


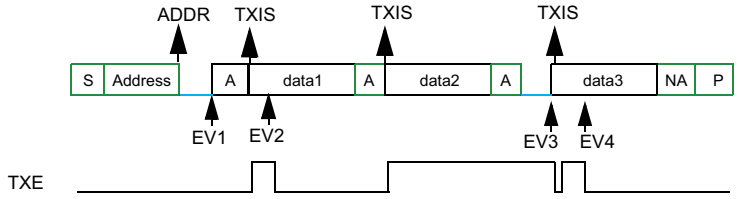
Figure 545. Transfer bus diagrams for I2C slave transmitter

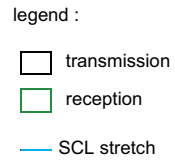
Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

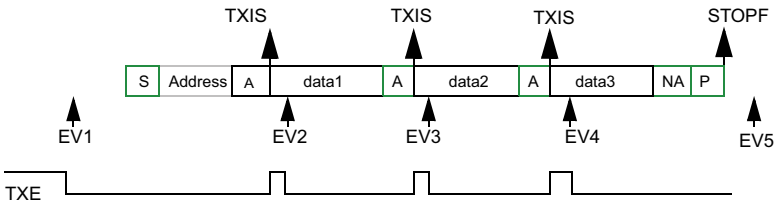
Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:

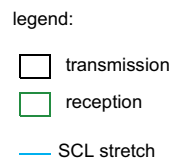




EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:





EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MS19853V1

Slave receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 546. Transfer sequence flowchart for slave receiver with NOSTRETCH=0

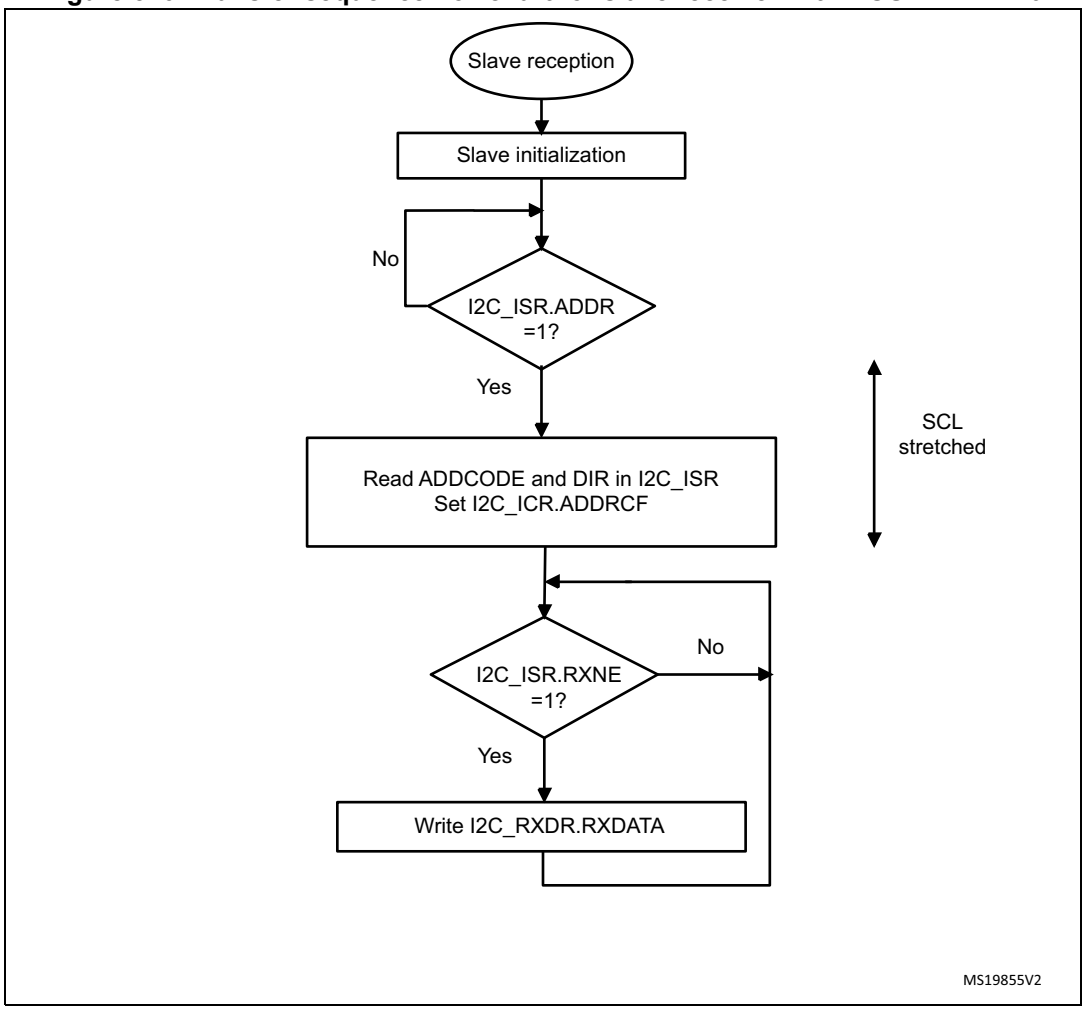


Figure 547. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

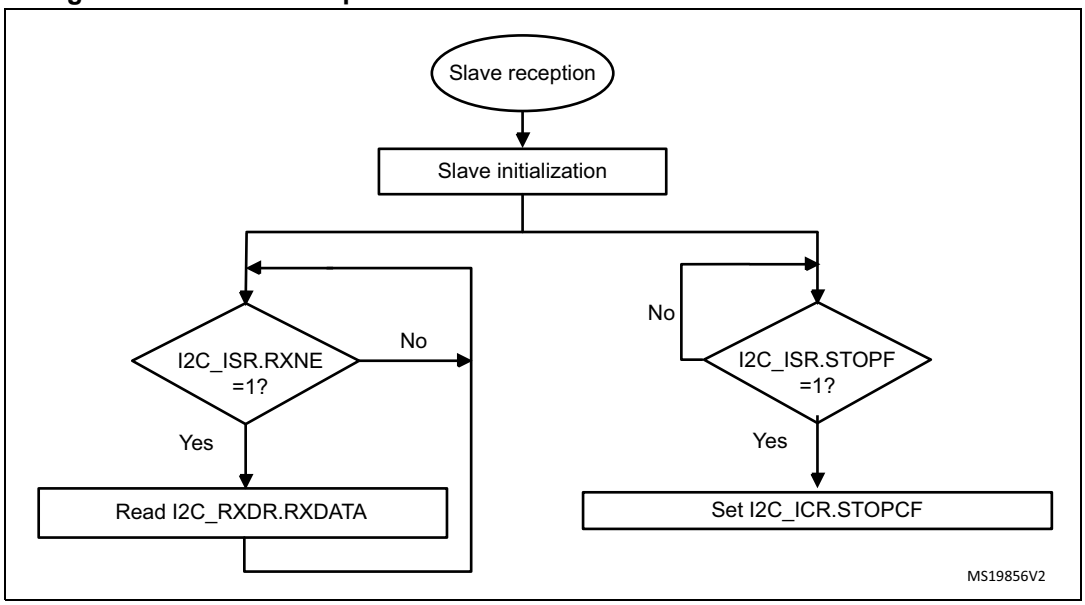
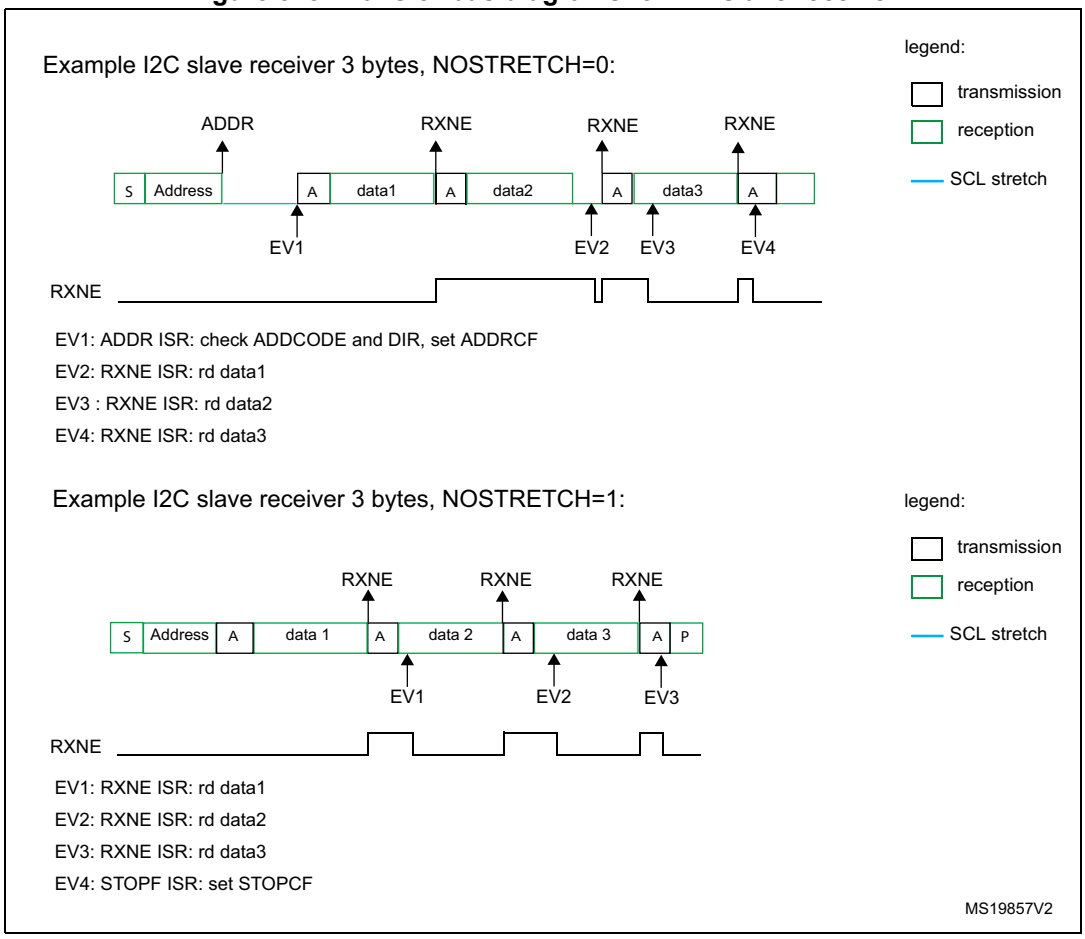


Figure 548. Transfer bus diagrams for I2C slave receiver



46.4.8 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$$

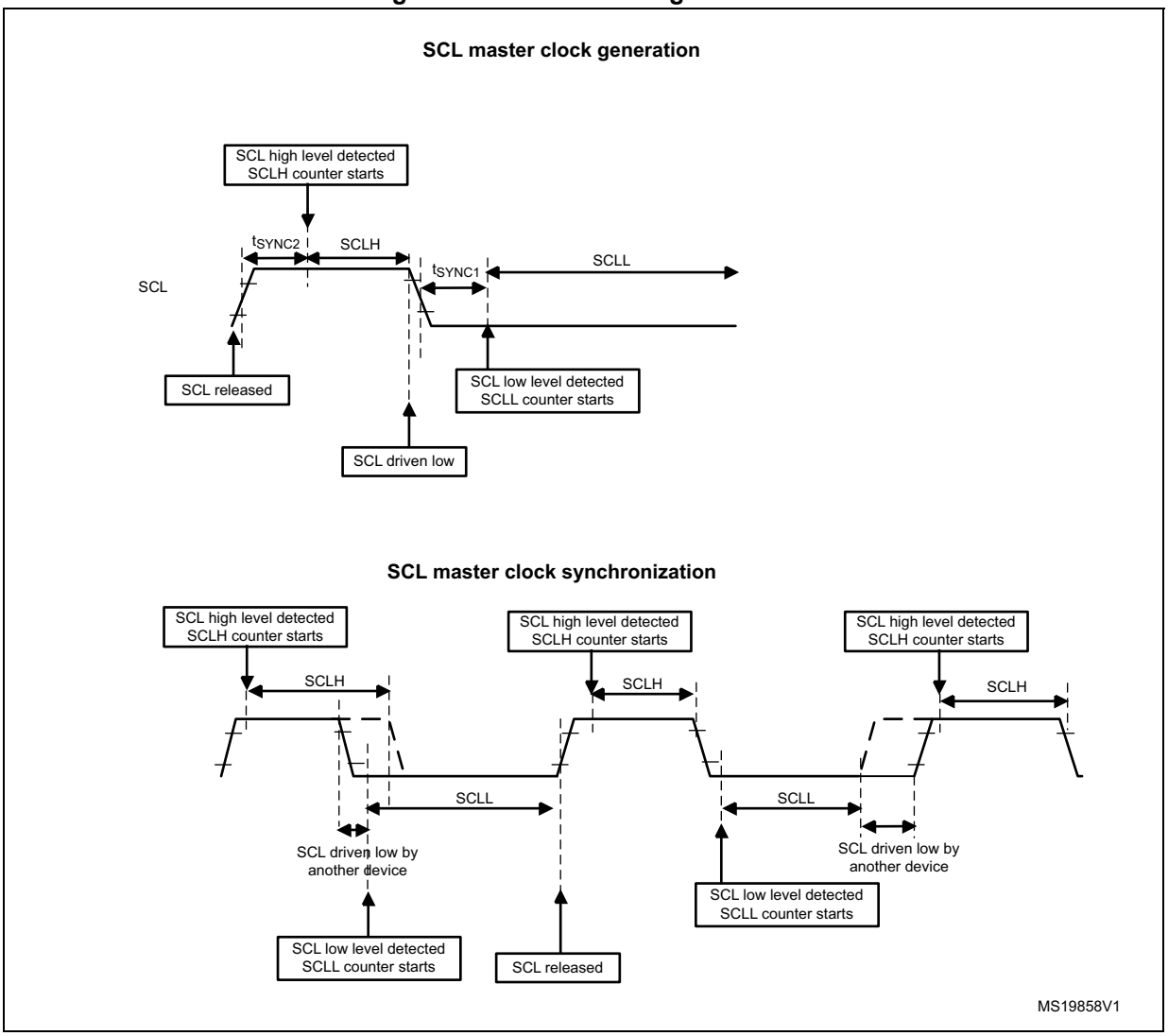
The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with i2c_ker_ck clock (2 to 3 i2c_ker_ck periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with i2c_ker_ck clock (2 to 3 i2c_ker_ck periods)

Figure 549. Master clock generation



Caution: In order to be I^{2}C or SMBus compliant, the master clock must respect the timings given below:

Table 374. I²C-SMBUS specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	µs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	µs
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	µs
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	ns

Note: SCLL is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.

SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 46.4.9: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. i2c_ker_ck frequency.

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

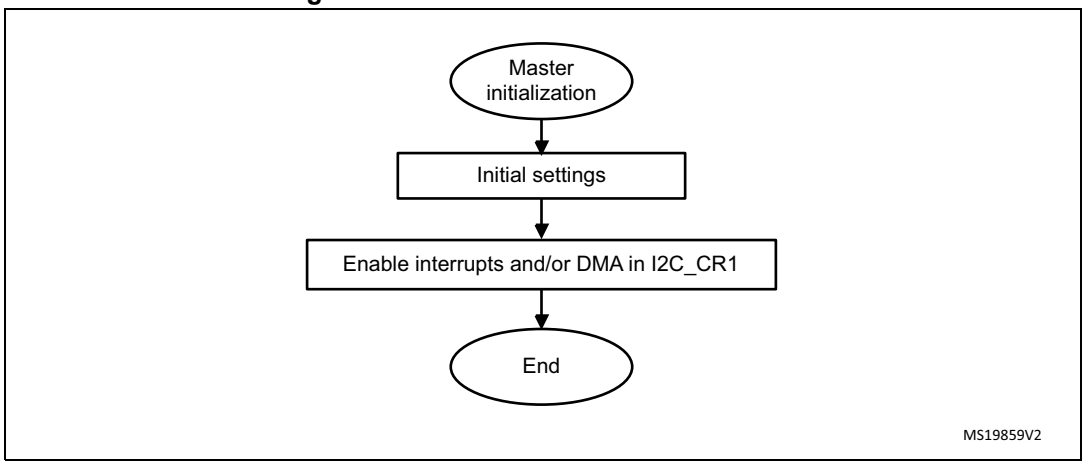
In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

Note: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.
In 10-bit addressing mode, when the Slave Address first 7 bits is NACKed by the slave, the

master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.
If the I2C is addressed as a slave (ADDR=1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared, when the ADDRCF bit is set.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY=1.

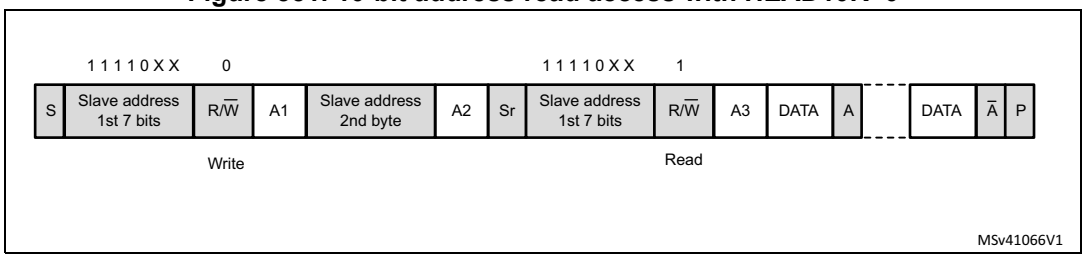
Figure 550. Master initialization flowchart



Initialization of a master receiver addressing a 10-bit address slave

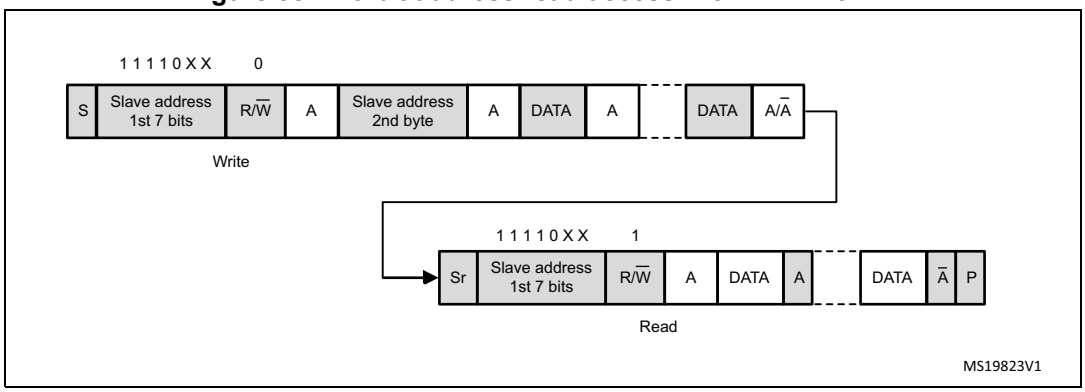
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address 2nd byte + REStart + Slave address 10-bit header Read

Figure 551. 10-bit address read access with HEAD10R=0



- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R=1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

Figure 552. 10-bit address read access with HEAD10R=1



Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 553. Transfer sequence flowchart for I2C master transmitter for N≤255 bytes

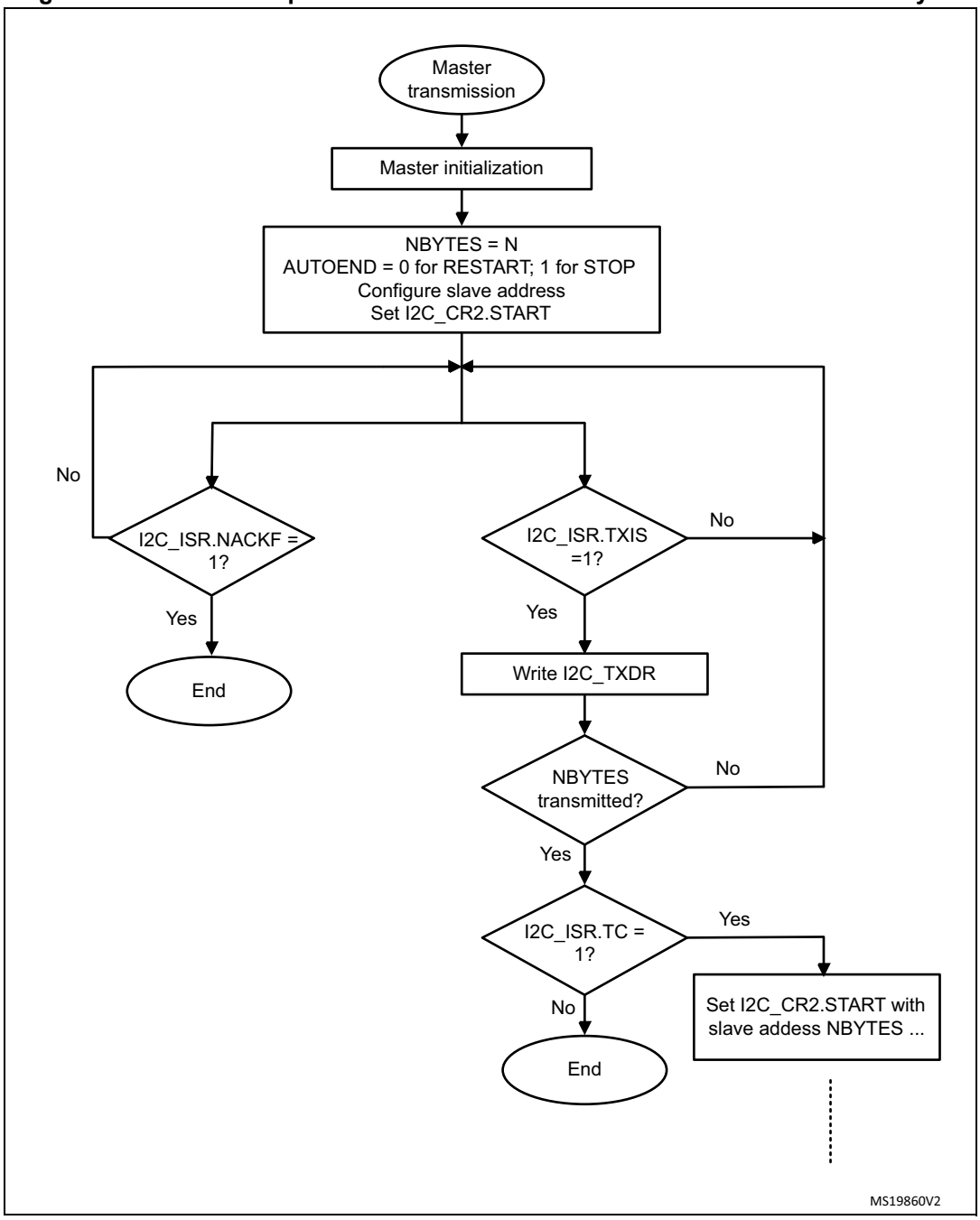


Figure 554. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

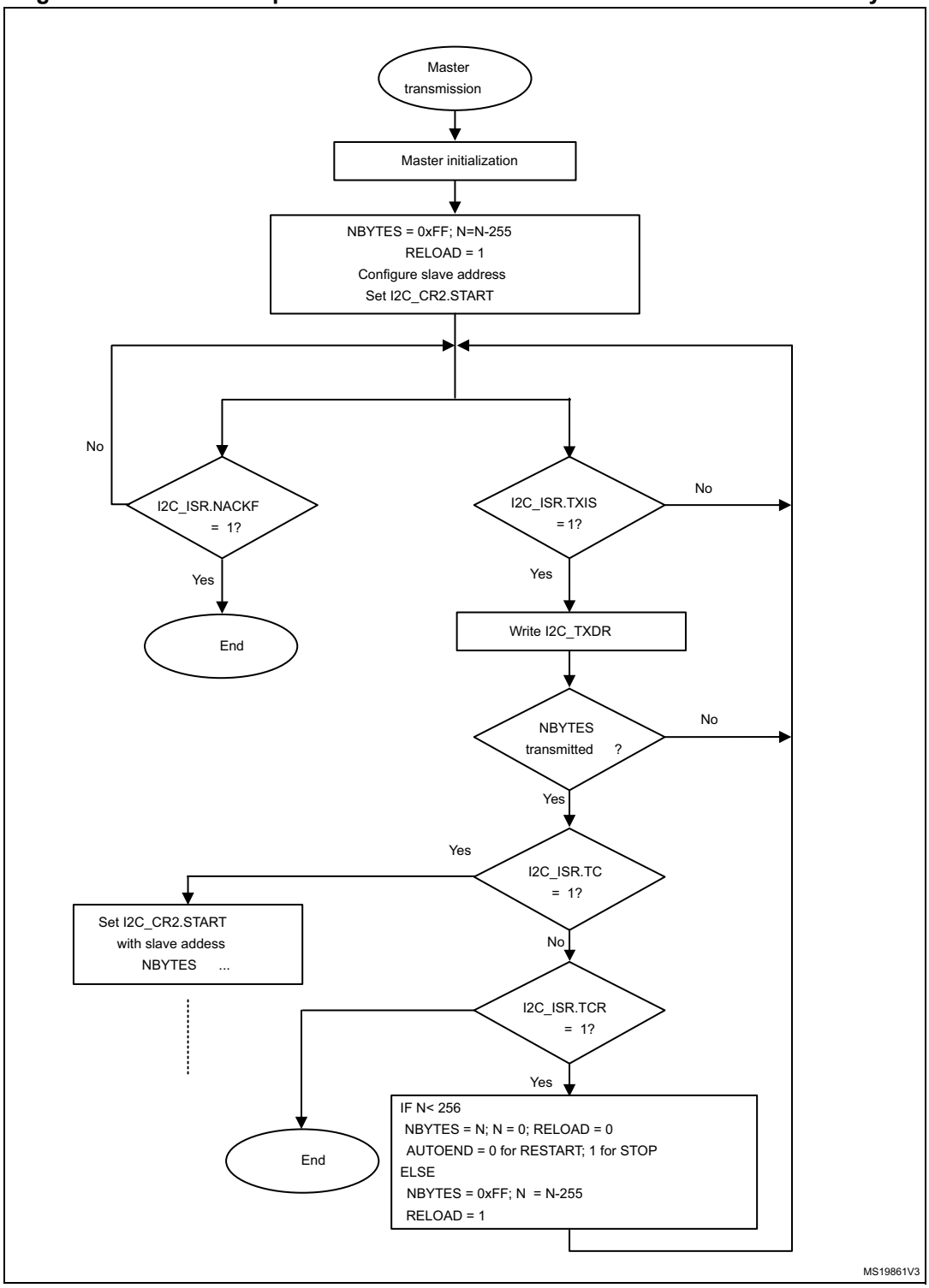
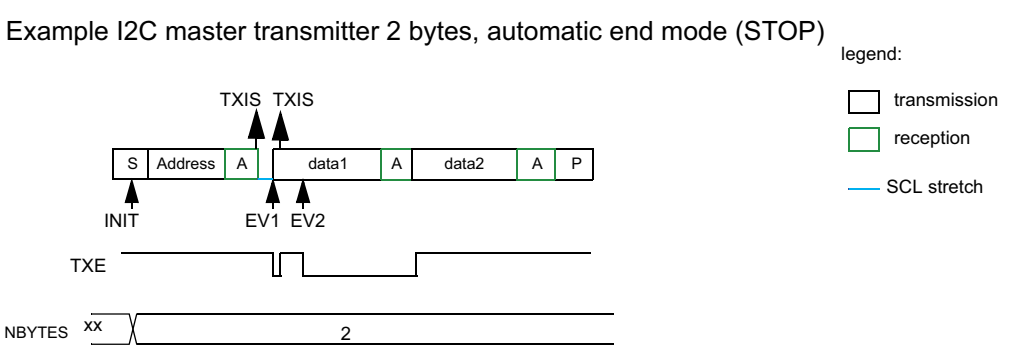


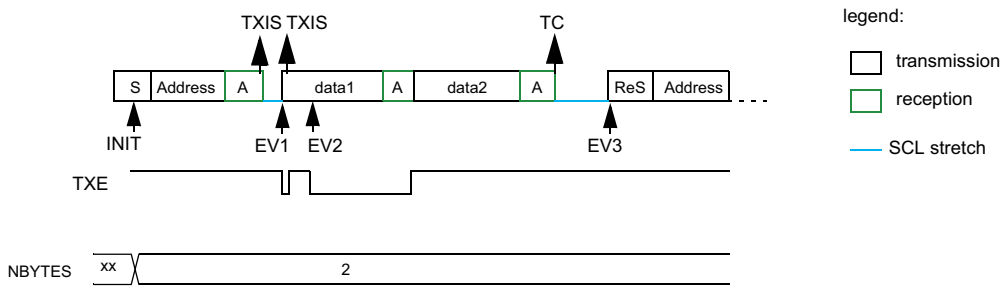
Figure 555. Transfer bus diagrams for I2C master transmitter

Example I2C master transmitter 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19862V1

Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 556. Transfer sequence flowchart for I2C master receiver for N≤255 bytes

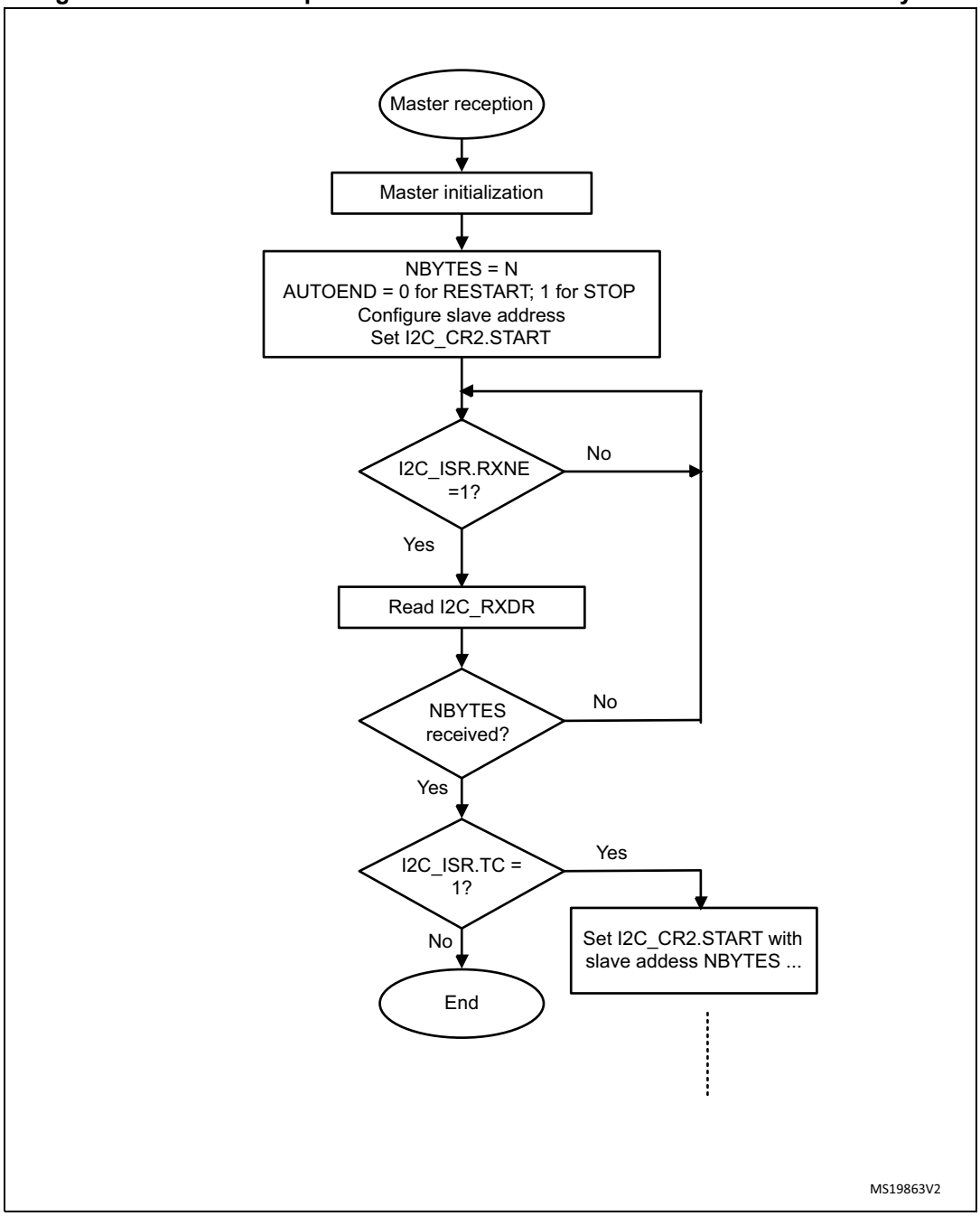


Figure 557. Transfer sequence flowchart for I2C master receiver for N >255 bytes

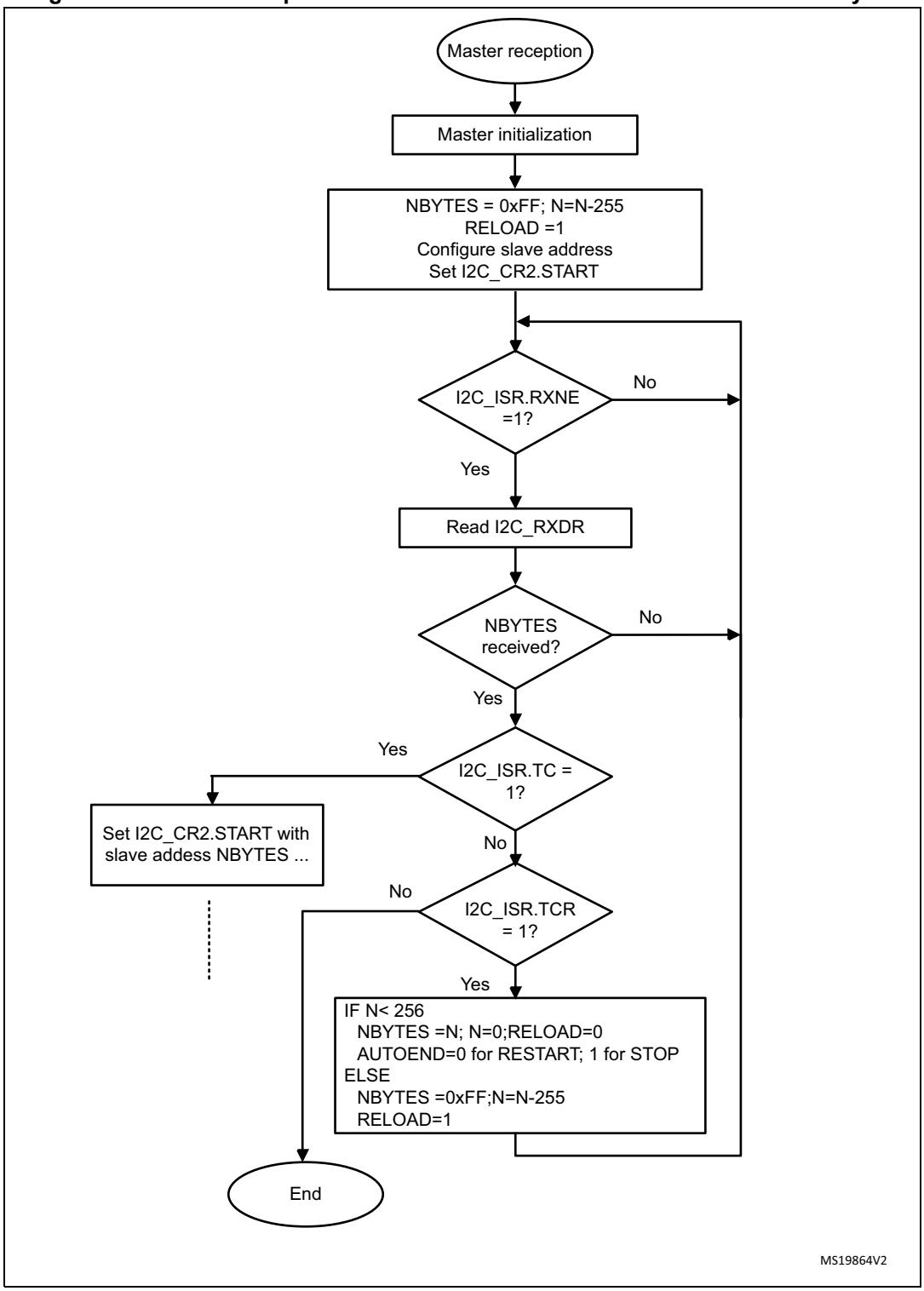
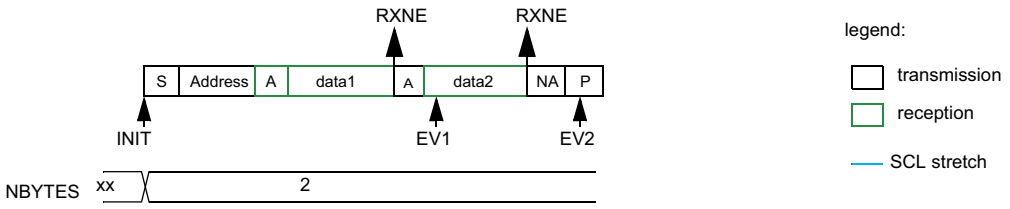


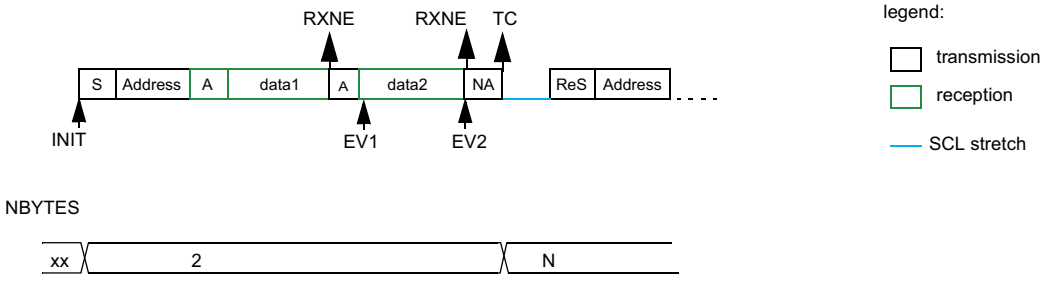
Figure 558. Transfer bus diagrams for I2C master receiver

Example I2C master receiver 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2

Example I2C master receiver 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: read data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19865V1

46.4.9 I2C_TIMINGR register configuration examples

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I^2C specification. In order to get more accurate configuration values, the STM32CubeMX tool (I2C Configuration window) must be used.

Table 375. Examples of timing settings for f_{I2CCLK} = 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
t_{SCLL}	200x250 ns = 50 µs	20x250 ns = 5.0 µs	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
t_{SCLH}	196x250 ns = 49 µs	16x250 ns = 4.0µs	4x125ns = 500ns	4x125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 376. Examples of timings settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.

2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 1000 ns.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 750 ns.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x t_{I2CCLK} = 250 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 500 ns.

Table 377. Examples of timings settings for f_{I2CCLK} = 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL} (1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~875 ns(4)
SDADEL	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. The SCL period t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 1000 ns
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 750 ns
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 250 ns

46.4.10 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification (http://smbus.org).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification (http://smbus.org).

Received Command and Data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *Slave Byte Control mode on page 1948* for more details.

Host Notify protocol

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus Host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the Alert Response Address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

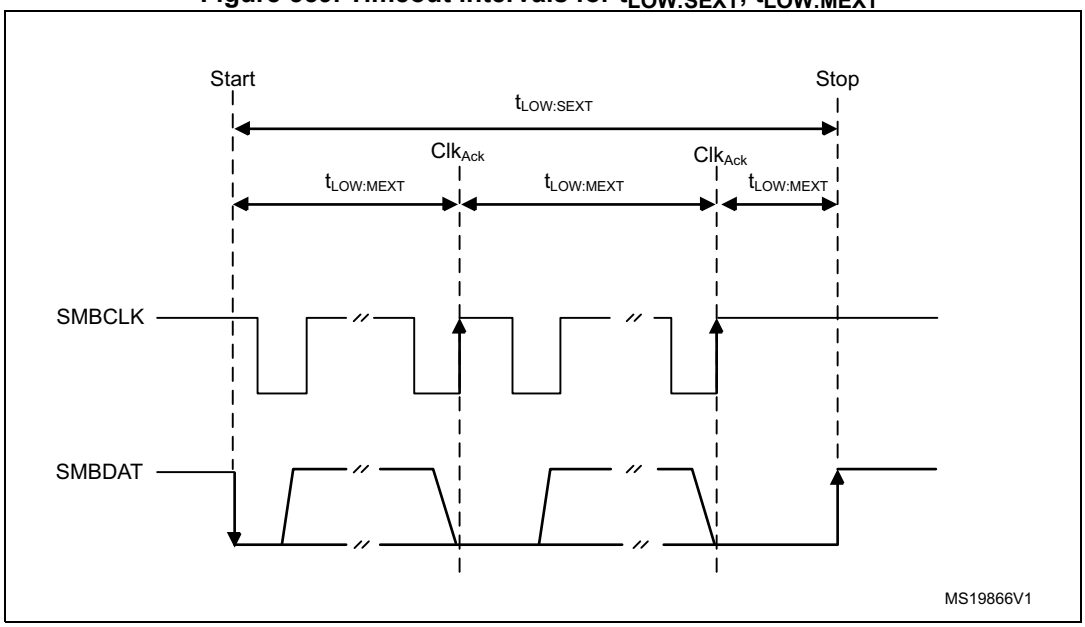
This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

Table 378. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (slave device)	-	25	ms
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (master device)	-	10	ms

1. $t_{LOW:SEXT}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. $t_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 559. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$. (refer to *Table 372: I2C-SMBUS specification data setup and hold times*)

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

46.4.11 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *Slave Byte Control mode on page 1948* for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection on page 1971* for more details.

- The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus Host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 379. SMBUS with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Master Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Master Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- $t_{TIMEOUT}$ check

  In order to enable the $t_{TIMEOUT}$ check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the $t_{TIMEOUT}$ parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA+1) x 2048 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 380: Examples of TIMEOUTA settings for various i2c_ker_ck frequencies (max $t_{TIMEOUT}$ = 25 ms)*.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

  Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check $t_{LOW:SEXT}$ for a slave and $t_{LOW:MEXT}$ for a master. As the standard specifies only a maximum, the user can choose the same value for the both.

  Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

  If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in *Bus idle detection on page 1971* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 381: Examples of TIMEOUTB settings for various i2c_ker_ck frequencies*

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

Bus Idle detection

In order to enable the t_{IDLE} check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the t_{IDLE} parameter. The TIDLE bit must be configured to '1 in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than (TIMEOUTA+1) x 4 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

Refer to *Table 382: Examples of TIMEOUTA settings for various i2c_ker_ck frequencies (max t_{IDLE} = 50 µs)*

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

46.4.12 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 380. Examples of TIMEOUTA settings for various i2c_ker_ck frequencies (max $t_{TIMEOUT}$ = 25 ms)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 381. Examples of TIMEOUTB settings for various i2c_ker_ck frequencies

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 382. Examples of TIMEOUTA settings for various i2c_ker_ck frequencies (max t_{IDLE} = 50 µs)

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

46.4.13 SMBus slave mode

This section is relevant only when SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

In addition to I2C slave transfer management (refer to *Section 46.4.7: I2C slave mode*) some additional software flowcharts are provided to support SMBus.

SMBus Slave transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In

that case the total number of TXIS interrupts is NBYTES-1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 560. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

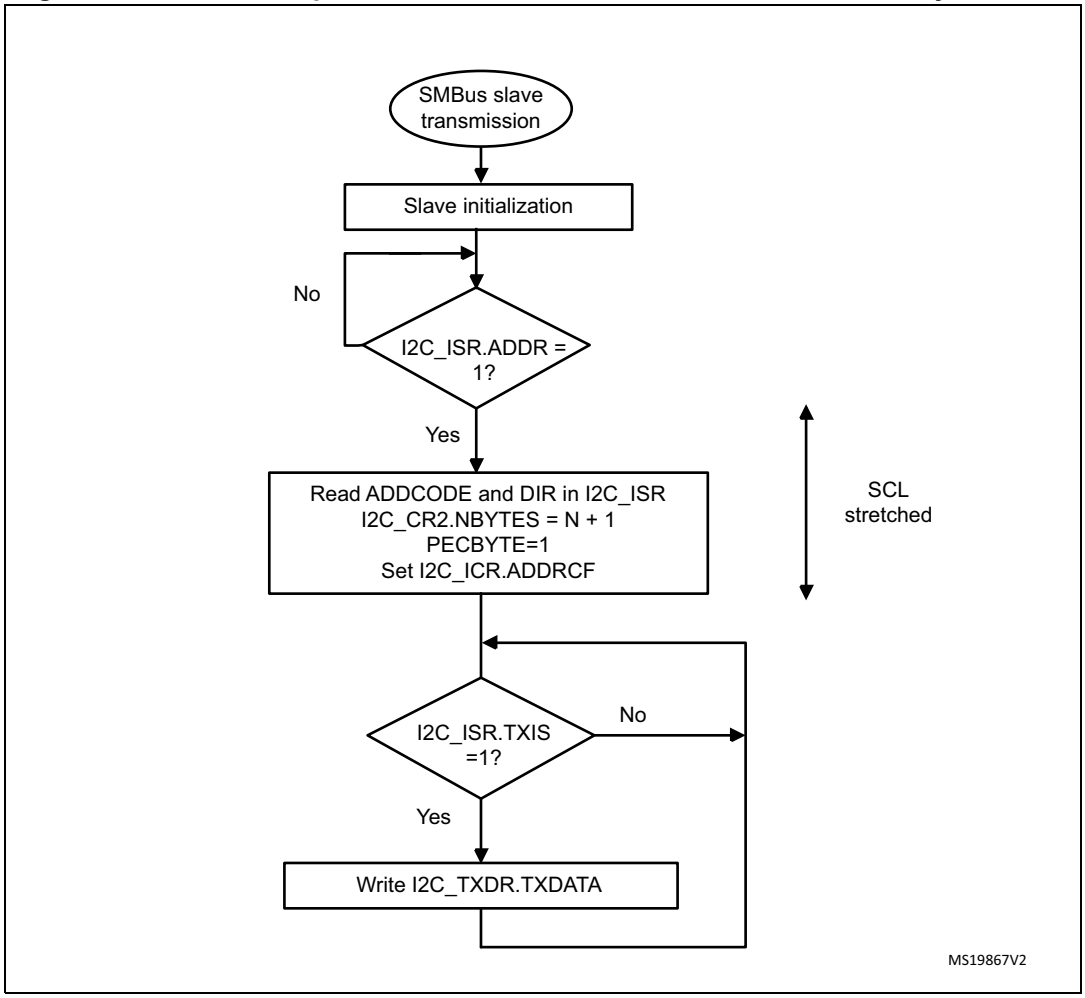
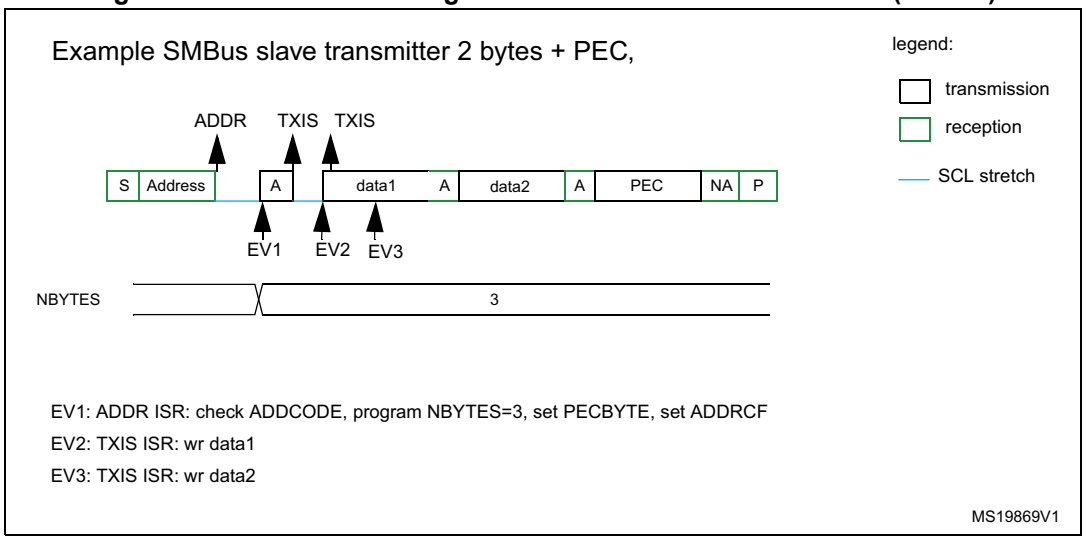


Figure 561. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to *Slave Byte Control mode on page 1948* for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 562. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

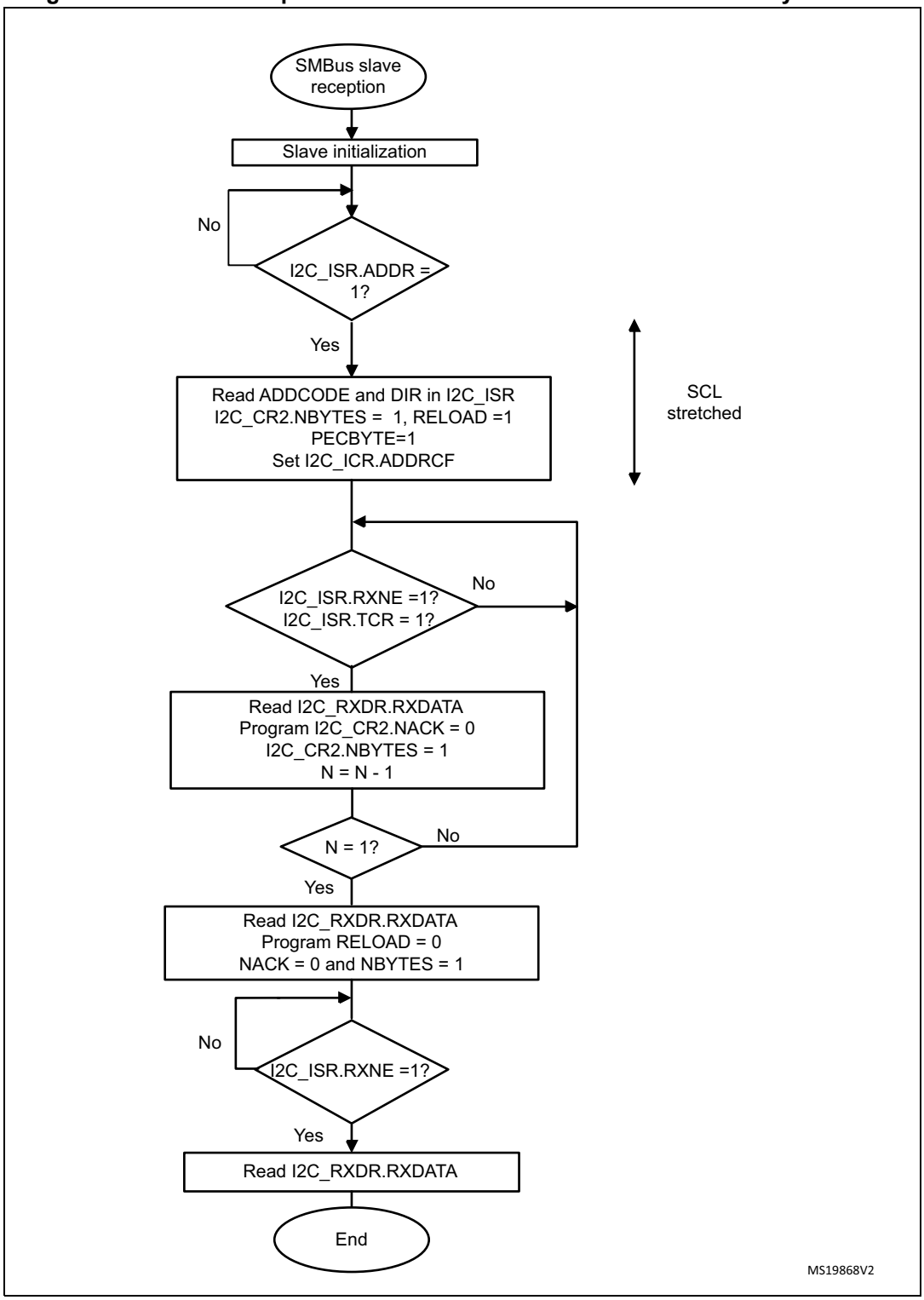
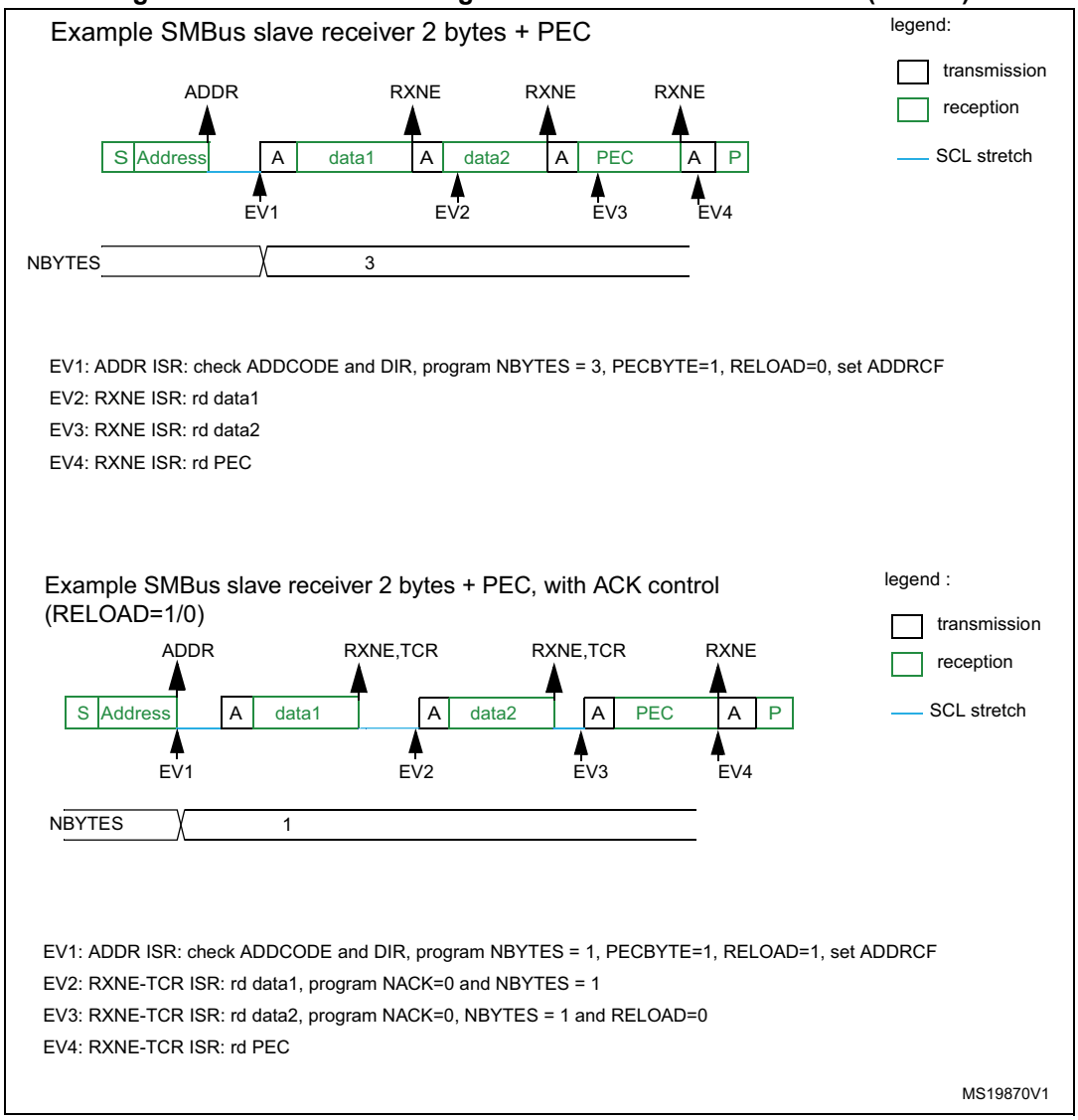


Figure 563. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

In addition to I2C master transfer management (refer to *Section 46.4.8: I2C master mode*) some additional software flowcharts are provided to support SMBus.

SMBus Master transmitter

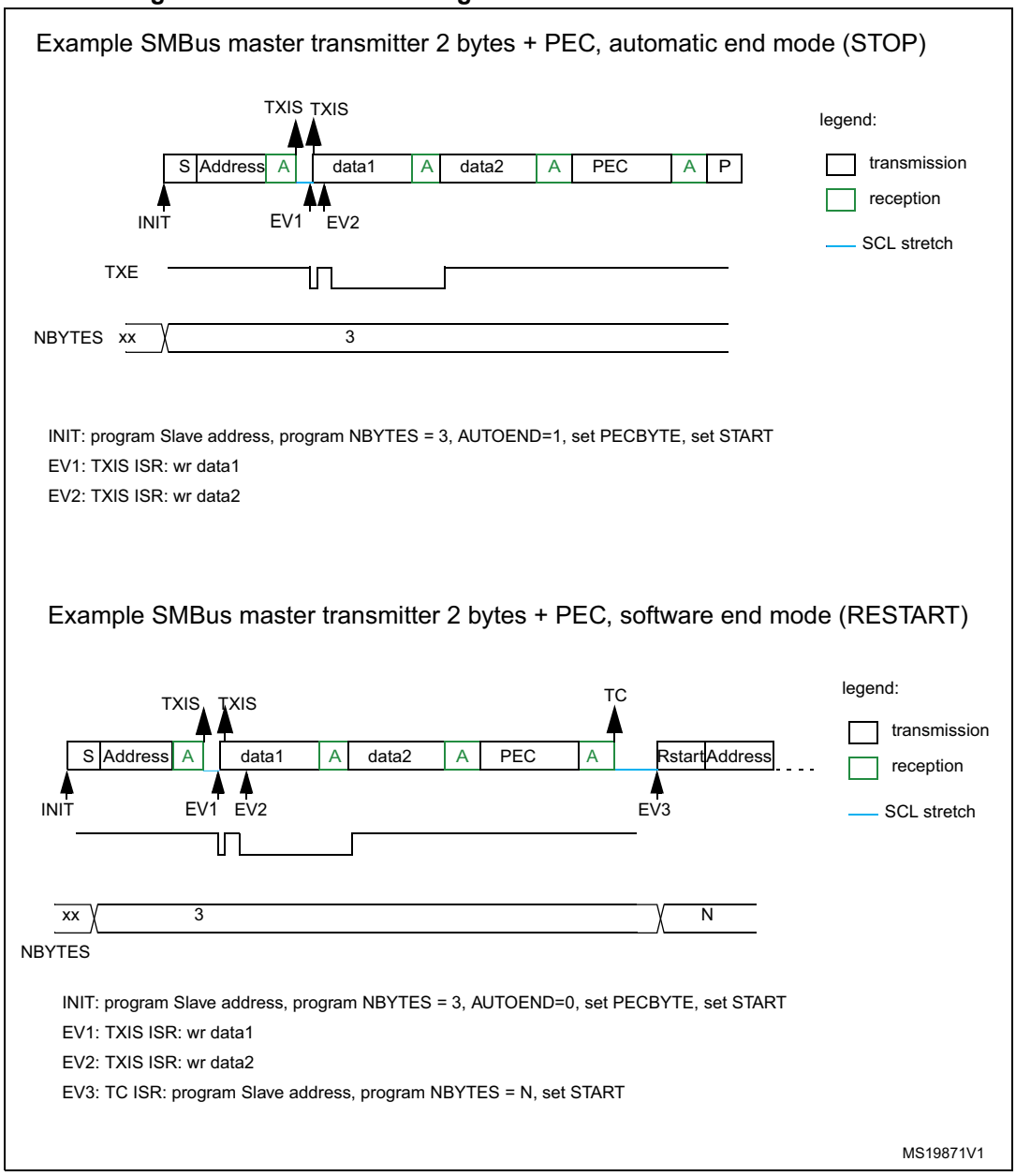
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 564. Bus transfer diagrams for SMBus master transmitter



SMBus Master receiver

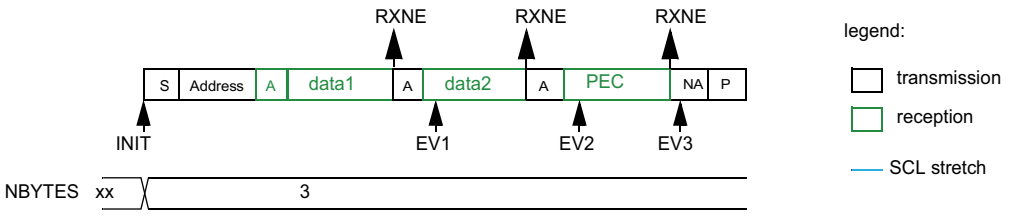
When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

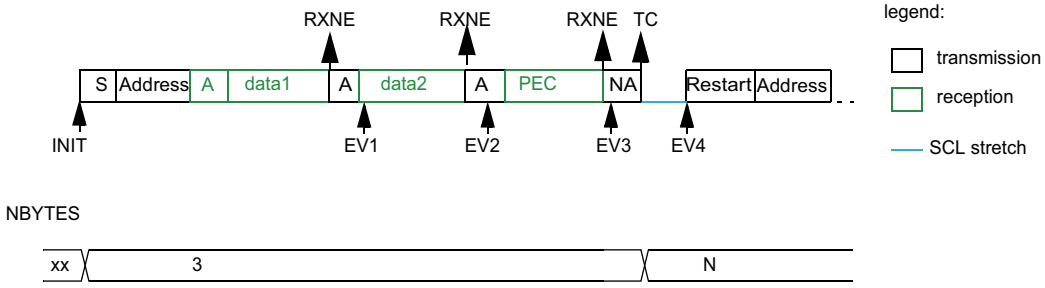
Figure 565. Bus transfer diagrams for SMBus master receiver

Example SMBus master receiver 2 bytes + PEC, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd PEC

Example SMBus master receiver 2 bytes + PEC, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: read PEC
EV4: TC ISR: program Slave address, program NBYTES = N, set START

MS19872V1

46.4.14 Wakeup from Stop mode on address match

This section is relevant only when Wakeup from Stop mode feature is supported. Refer to *Section 46.3: I2C implementation*.

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI or CSI oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI or CSI is switched off. When a START is detected, the I2C interface switches the HSI or CSI on, and stretches SCL low until HSI or CSI is woken up.

HSI or CSI is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI or CSI is switched off again and the MCU is not woken up.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the HSI or CSI *is not switched on after a START is received.*

Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wakeup from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

Caution: This feature is available only when the I2C clock source is the HSI or CSI oscillator.

Caution: Clock stretching must be enabled (NOSTRETCH=0) to ensure proper operation of the wakeup from Stop mode feature.

Caution: If wakeup from Stop mode is disabled (WUPEN=0), the I2C peripheral must be disabled before entering Stop mode (PE=0).

46.4.15 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter)

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Refer to *Section 46.3: I2C implementation*.

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

46.4.16 DMA requests

Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Section 14: Direct memory access controller (DMA) on page 615*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Master transmitter on page 1959*.
- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to *SMBus Slave transmitter on page 1973* and *SMBus Master transmitter on page 1977*.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to *Section 14: Direct memory access controller (DMA) on page 615*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the

DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 46.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Slave receiver on page 1975* and *SMBus Master receiver on page 1979*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

46.4.17 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

46.5 I2C low-power modes

Table 383. Effect of low-power modes on the I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The I2C registers content is kept. If WUPEN = 1 and I2C is clocked by an internal oscillator (HSI or CSI): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If WUPEN=0: the I2C must be disabled before entering Stop mode
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. The ADDR match event can wake-up the device from Stop mode only if the I2C instance supports the Wakeup from Stop mode feature. Refer to I2C implementation table.

46.6 I2C interrupts

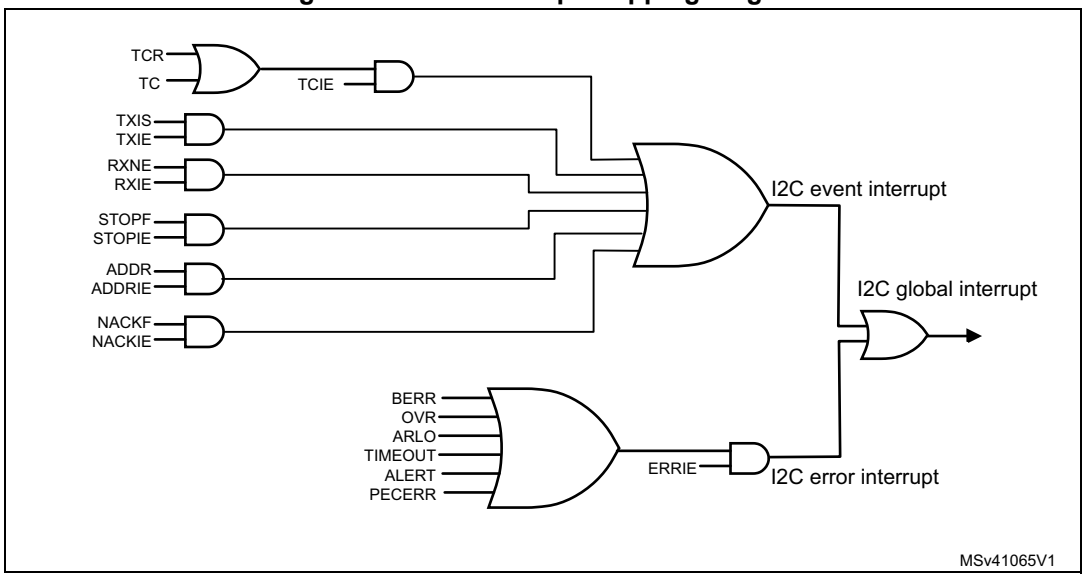
In I2C two interrupts (i2c_event_it and i2c_error_it) and a wakeup event signal (i2c_wkup) can be generated according to the events described in the table below:

Table 384. I2C Interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit	Interrupt/Wakeup activated		
				i2c_event_it	i2c_error_it	i2c_wkup
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE	Yes	No	No
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE			
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE			
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] ≠ 0	TCIE			
Transfer complete	TC	Write START=1 or STOP=1				
Address matched	ADDR	Write ADDRCF=1	ADDRIE			Yes(1)
NACK reception	NACKF	Write NACKCF=1	NACKIE			No
Bus error	BERR	Write BERRCF=1	ERRIE	N0	Yes	
Arbitration loss	ARLO	Write ARLOCF=1				
Overrun/Underrun	OVR	Write OVRCF=1				
PEC error	PECERR	Write PECERRCF=1				
Timeout/t_{LOW} error	TIMEOUT	Write TIMEOUTCF=1				
SMBus Alert	ALERT	Write ALERTCF=1				

1. If WUPEN is set.

Figure 566. I2C interrupt mapping diagram



46.7 I2C registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

46.7.1 I2C2 control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	WUPE N	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF[3:0]				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 22 **ALERTEN**: SMBus alert enable

Device mode (SMBHEN=0):

0: Releases SMBA pin high and Alert Response Address Header disabled: 0001100x followed by NACK.

1: Drives SMBA pin low and Alert Response Address Header enables: 0001100x followed by ACK.

Host mode (SMBHEN=1):

0: SMBus Alert pin (SMBA) not supported.

1: SMBus Alert pin (SMBA) supported.

Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 21 **SMBDEN**: SMBus Device Default address enable

0: Device default address disabled. Address 0b1100001x is NACKed.

1: Device default address enabled. Address 0b1100001x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 20 **SMBHEN**: SMBus Host address enable

0: Host address disabled. Address 0b0001000x is NACKed.

1: Host address enabled. Address 0b0001000x is ACKed.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.

1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wakeup from Stop mode enable

0: Wakeup from Stop mode disable.

1: Wakeup from Stop mode enable.

Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Note: WUPEN can be set only when DNF = '0000'

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.

0: Clock stretching enabled

1: Clock stretching disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 16 **SBC**: Slave byte control

This bit is used to enable hardware byte control in slave mode.

0: Slave byte control disabled

1: Slave byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception

1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission

1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to15 t_{I2CCLK}
Note: If the analog filter is also enabled, the digital filter is added to the analog filter.
This filter can only be programmed when the I2C is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generate an interrupt:
Arbitration Loss (ARLO)
Bus Error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer Complete interrupt enable
0: Transfer Complete interrupt disabled
1: Transfer Complete interrupt enabled
Note: Any of these events generate an interrupt:
Transfer Complete (TC)
Transfer Complete Reload (TCR)

Bit 5 **STOPIE**: Stop detection Interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable
Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

46.7.2 I2C2 control register 2 (I2C_CR2)

Address offset: 0x0C4

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte
This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE=0.
0: No PEC transfer.
1: PEC transmission/reception is requested
Note: Writing '0' to this bit has no effect.
This bit has no effect when RELOAD is set.
This bit has no effect is slave mode when SBC=0.
If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (master mode)
This bit is set and cleared by software.
0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.
1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.
Note: This bit has no effect in slave mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode
This bit is set and cleared by software.
0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).
1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

In Master Mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCF bit in the I2C_ICR register.

0: No Start generation.

1: Restart/Start generation:

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.

Otherwise setting this bit generates a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)

0: The master operates in 7-bit addressing mode,

1: The master operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)
0: Master requests a write transfer.
1: Master requests a read transfer.
Note: Changing this bit when the START bit is set is not allowed.

Bits 9:8 **SADD[9:8]**: Slave address bit 9:8 (master mode)
In 7-bit addressing mode (ADD10 = 0):
These bits are don't care
In 10-bit addressing mode (ADD10 = 1):
These bits should be written with bits 9:8 of the slave address to be sent
Note: Changing these bits when the START bit is set is not allowed.

Bits 7:1 **SADD[7:1]**: Slave address bit 7:1 (master mode)
In 7-bit addressing mode (ADD10 = 0):
These bits should be written with the 7-bit slave address to be sent
In 10-bit addressing mode (ADD10 = 1):
These bits should be written with bits 7:1 of the slave address to be sent.
Note: Changing these bits when the START bit is set is not allowed.

Bit 0 **SADD0**: Slave address bit 0 (master mode)
In 7-bit addressing mode (ADD10 = 0):
This bit is don't care
In 10-bit addressing mode (ADD10 = 1):
This bit should be written with bit 0 of the slave address to be sent
Note: Changing these bits when the START bit is set is not allowed.

46.7.3 I2C2 own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own Address 1 enable
0: Own address 1 disabled. The received slave address OA1 is NACKed.
1: Own address 1 enabled. The received slave address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own Address 1 10-bit mode
0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.
Note: This bit can be written only when OA1EN=0.

Bits 9:8 **OA1[9:8]**: Interface address
7-bit addressing mode: do not care
10-bit addressing mode: bits 9:8 of address
Note: These bits can be written only when OA1EN=0.

Bits 7:1 **OA1[7:1]**: Interface address
7-bit addressing mode: 7-bit address
10-bit addressing mode: bits 7:1 of 10-bit address
Note: These bits can be written only when OA1EN=0.

Bit 0 **OA1[0]**: Interface address
7-bit addressing mode: do not care
10-bit addressing mode: bit 0 of address
Note: This bit can be written only when OA1EN=0.

46.7.4 I2C2 own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own Address 2 enable
0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

46.7.5 I2C2 timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 1940*) and for SCL high and low level counters (refer to *I2C master initialization on page 1955*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

46.7.6 I2C2 timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable
0: Extended clock timeout detection is disabled
1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B
This field is used to configure the cumulative clock extension timeout:
In master mode, the master cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
In slave mode, the slave cumulative clock low extend time ($t_{LOW:SEXT}$) is detected
$t_{LOW:EXT}$= (TIMEOUTB+1) x 2048 x t_{I2CCLK}
Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable
0: SCL timeout detection is disabled
1: SCL timeout detection is enabled: when SCL is low for more than $t_{TIMEOUT}$ (TIDLE=0) or high for more than t_{IDLE} (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection
0: TIMEOUTA is used to detect SCL low timeout
1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)
Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A
This field is used to configure:
The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE=0
$t_{TIMEOUT}$= (TIMEOUTA+1) x 2048 x t_{I2CCLK}
The bus idle condition (both SCL and SDA high) when TIDLE=1
t_{IDLE}= (TIMEOUTA+1) x 4 x t_{I2CCLK}
Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 46.3: I2C implementation.

46.7.7 I2C2 interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR=1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE=0.

If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 10 **OVR**: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting *BERRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 7 **TCR**: Transfer Complete Reload

This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE=0.

This flag is only for master mode, or for slave mode when the SBC bit is set.

Bit 6 **TC**: Transfer Complete (master mode)

This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE=0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 4 **NACKF**: Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE=0.

Bit 3 **ADDR**: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE=0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE=0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).

Note: This bit is cleared by hardware when PE=0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE=0.

46.7.8 I2C2 interrupt clear register (I2C_ICR)(

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERT CF	TIM OUTCF	PECCF	OVRCF	ARLO CF	BERR CF	Res.	Res.	STOP CF	NACK CF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 46.3: I2C implementation.

Bit 10 **OVRCF**: Overrun/Underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

46.7.9 I2C2 PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to Section 46.3: I2C implementation.

46.7.10 I2C2 receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]** 8-bit receive data
Data byte received from the I^2C bus

46.7.11 I2C2 transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data
Data byte to be transmitted to the I^2C bus
Note: These bits can be written only when TXE=1.

46.7.12 I2C register map

The table below provides the I2C register map and reset values.

Table 385. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	**I2C_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	**I2C_CR2**	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	**I2C_OAR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	**I2C_OAR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]	OA2[7:1]									Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	**I2C_TIMINGR**	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**I2C_TIMEOUTR**	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**I2C_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	**I2C_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	**I2C_PECR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	**I2C_RXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 385. I2C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**I2C_TXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

47 Universal synchronous receiver transmitter (USART) /universal asynchronous receiver transmitter (UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

47.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

47.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection

47.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T=0 and T=1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

47.4 USART implementation

Table 386 describes USART implementation on STM32H743/753 and STM32H750 devices.

Table 386. USART/LPUART features

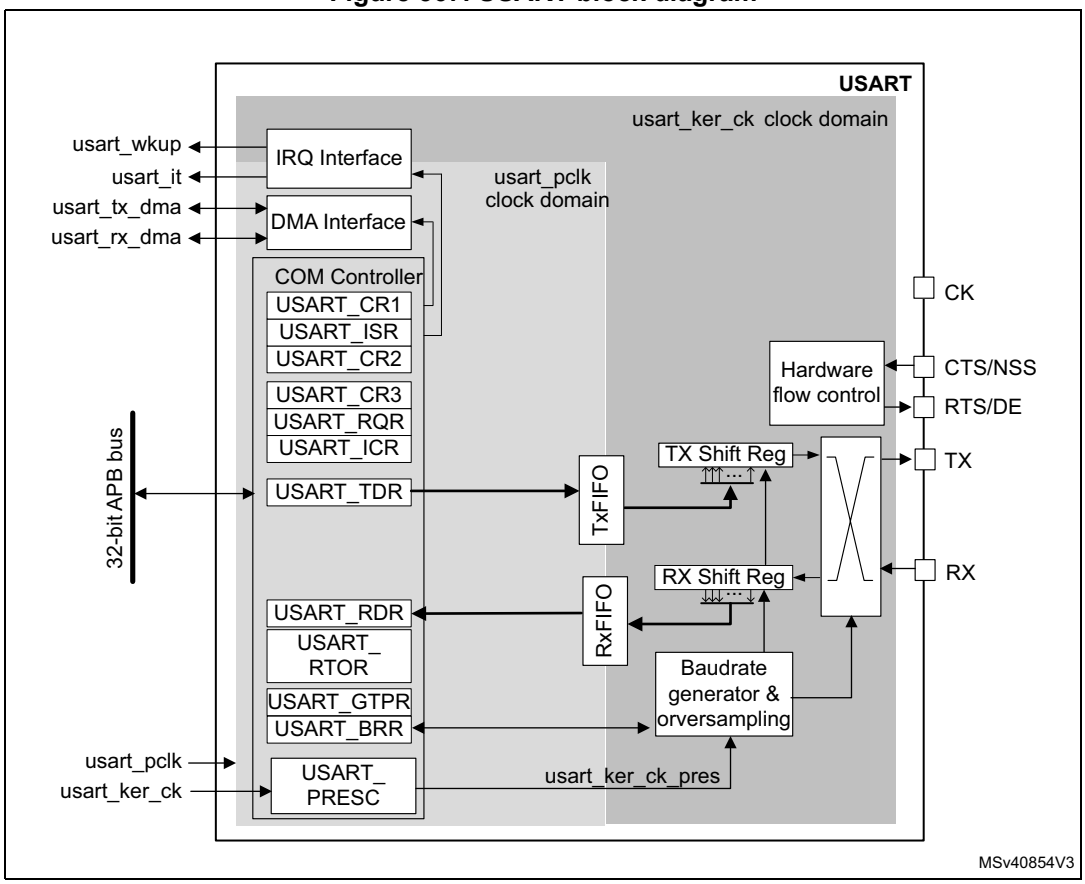
USART modes/features(1)	USART1/2/3/6	UART4/5/7/8	LPUART
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain and wakeup from low-power mode	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size	16		

1. X = supported.

47.5 USART functional description

47.5.1 USART block diagram

Figure 567. USART block diagram



The simplified block diagram given in *Figure 567* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is disabled, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external SCLK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

47.5.2 USART signals

USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

  RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

- **TX** (Transmit Data Output)

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire and Smartcard modes, this I/O is used to transmit and receive data.

RS232 Hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)

  When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request To Send)

  When it is low, this signal indicates that the USART is ready to receive data.

RS485 Hardware control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)

  This signal activates the transmission mode of the external transceiver.

Note: DE and RTS share the same pin.

Synchronous master/slave mode and Smartcard mode

The following pin is required in synchronous master/slave mode and Smartcard mode:

- CK

  This pin acts as Clock output in Synchronous master and Smartcard modes.

  It acts as Clock input is Synchronous slave mode.

  In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

  In Smartcard mode, CK output provides the clock to the smartcard.

- NSS

  This pin acts as Slave Select input in Synchronous slave mode.

Note: NSS and CTS share the same pin.

47.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 568*):

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

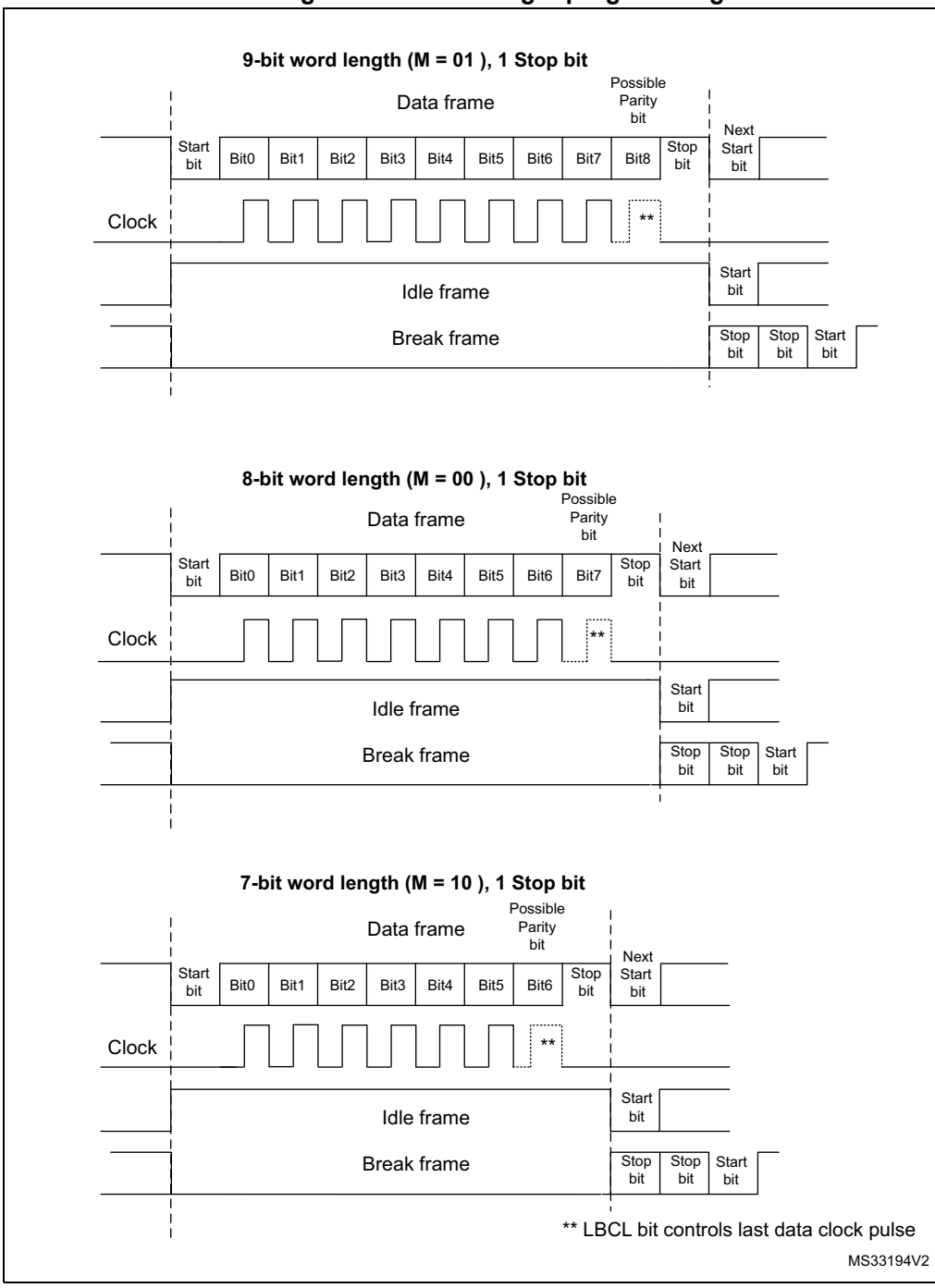
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 568. Word length programming



47.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and Smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG data have been received: 1 data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag will be set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data will not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

47.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the SCLK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

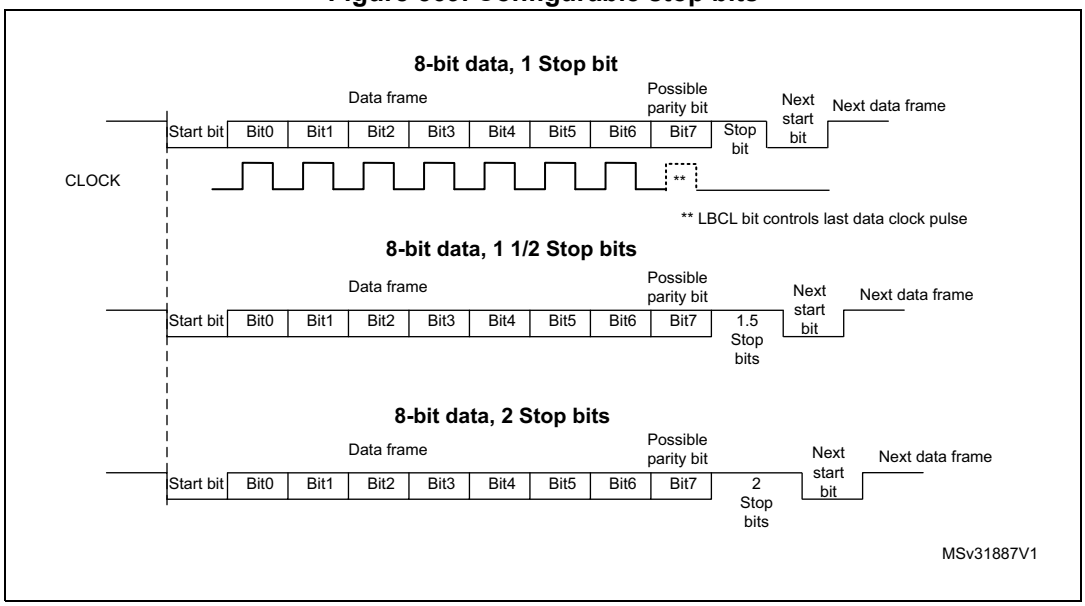
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This will be supported by normal USART, Single-wire and Modem modes.
- ***1.5 stop bits***: To be used in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits (see *Figure 569*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 569. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 47.5.10: USART multiprocessor communication*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC=1.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:

  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;

  – the USART_TDR register is empty;

  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:

  – the TXFIFO is not full;

  – the USART_TDR register is empty;

  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

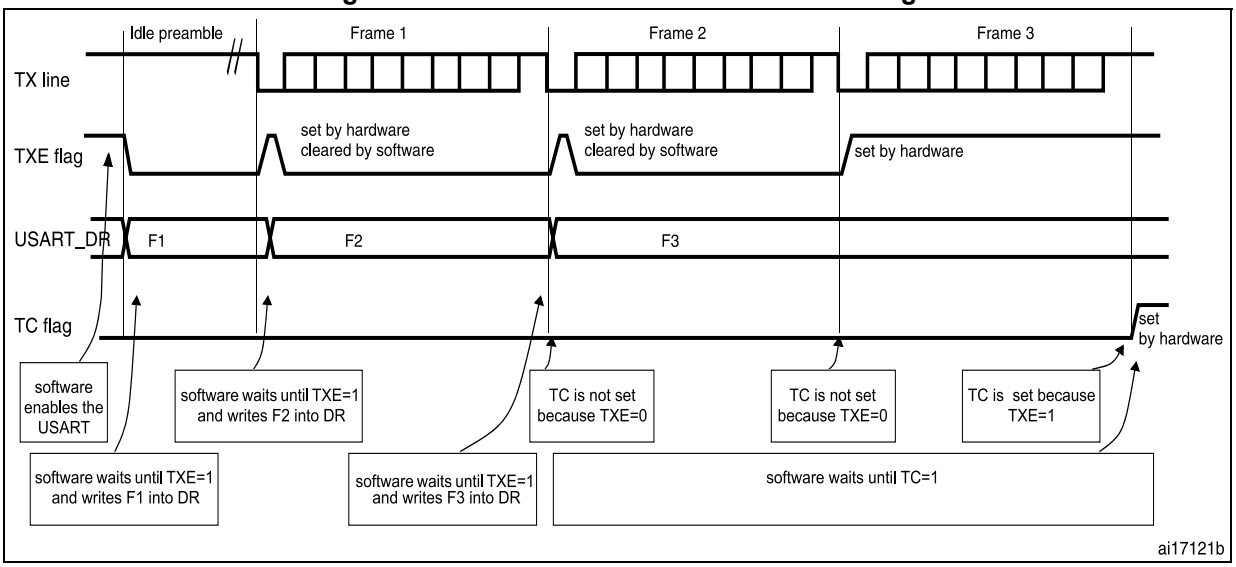
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 570: TC/TXE behavior when transmitting*).

Figure 570. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 568*).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

47.5.6 USART receiver

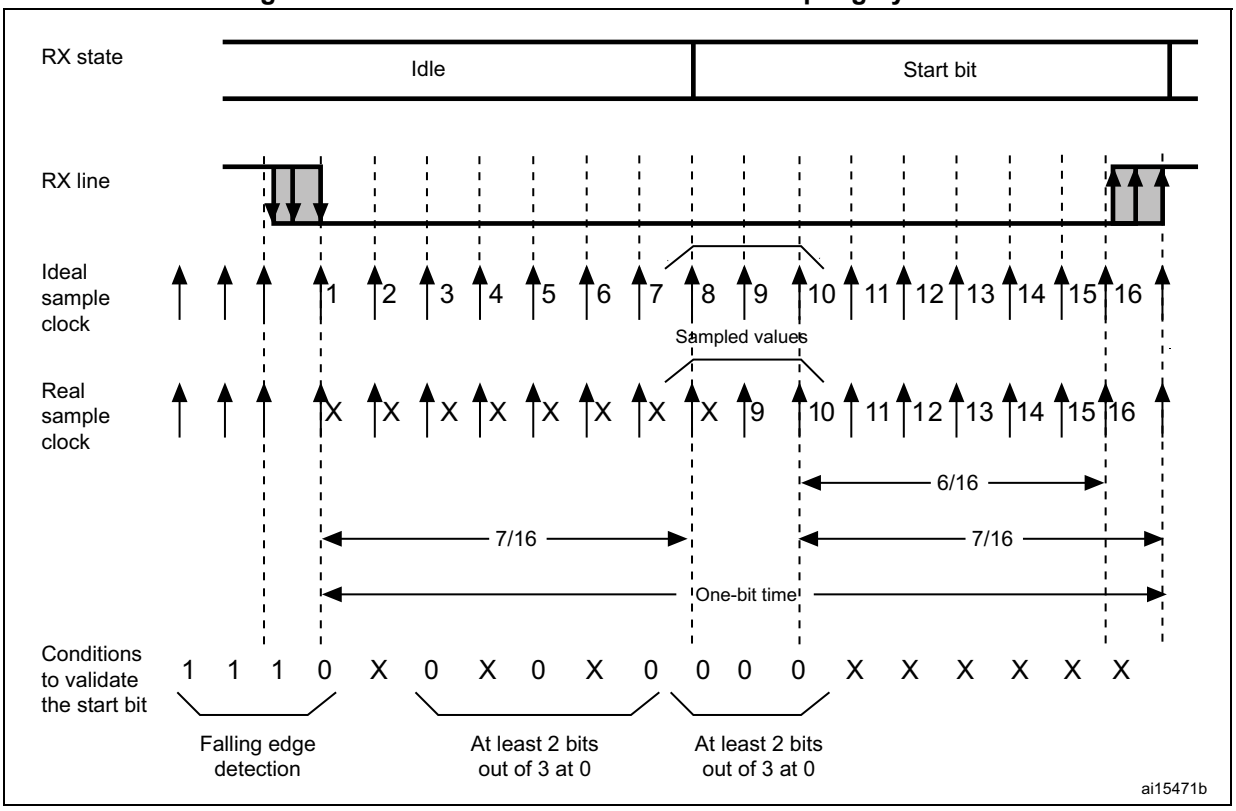
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 571. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE=1, or RXFNE flag set and interrupt generated if RXFNEIE=1 if FIFO mode enabled) if the 3 sampled bits are at ‘0’ (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at ‘0’ and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at ‘0’).

The start bit is validated but the NE noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at ‘0’ (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at ‘0’.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 47.5.10: USART multiprocessor communication*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to ‘1’ in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to ‘1’ in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content will not be lost. The previous data is available by reading the USART_RDR register.
  – the shift register will be overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO will not be lost. It is available by reading the USART_RDR register.
  – The shift register will be overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE=0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section 7: Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

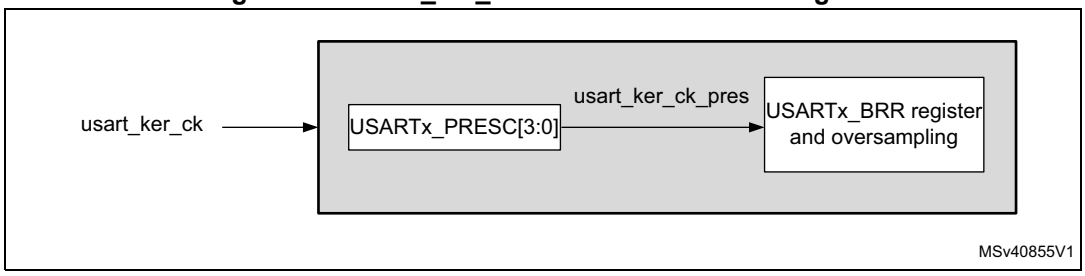
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section 7: Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 572. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources allow the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to allow USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This allows obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 573* and *Figure 574*).

Depending on your application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 47.5.8: Tolerance of the USART receiver to clock deviation on page 2023*)
- select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on your application:

  – select the three sample majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Figure 387*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 47.5.8: Tolerance of the USART receiver to clock deviation on page 2023*). In this case the NE bit will never be set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI mode.

Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to ‘0 ’ by hardware.

Figure 573. Data sampling when oversampling by 16

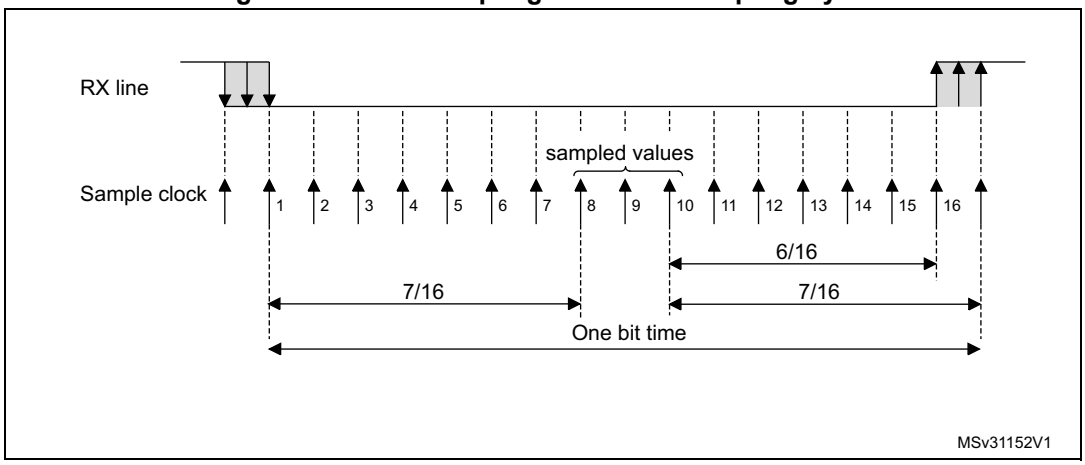


Figure 574. Data sampling when oversampling by 8

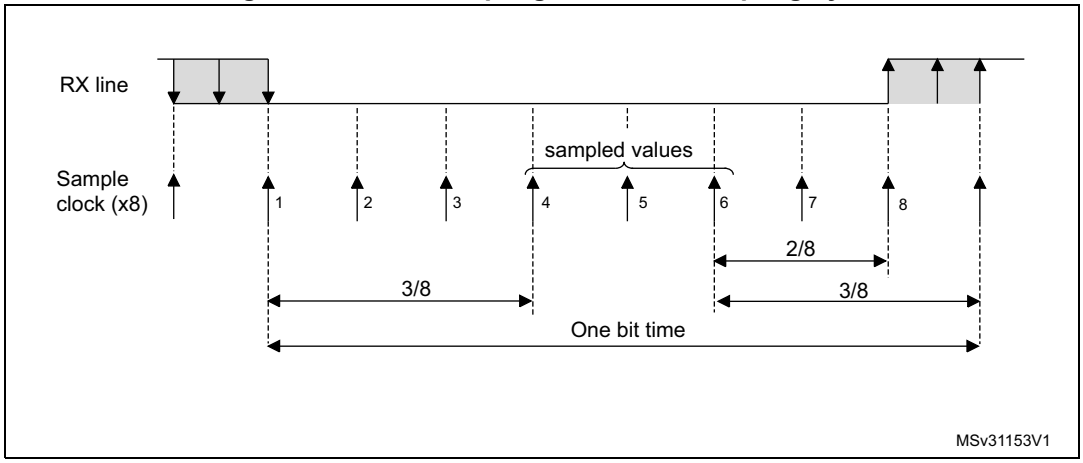


Table 387. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing '1' to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- 1.5 stop bits (Smartcard mode)

  When transmitting in Smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE =1 in USART_CR1) and the stop bit is checked to test if the Smartcard has detected a parity error.

  In the event of a parity error, the Smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 47.5.16: USART receiver timeout on page 2036* for more details).
- 2 stop bits

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

47.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = '0' or '1')

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ckpres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ckpres}}{\text{USARTDIV}}$$

Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ckpres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 baud with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 833d = 0341h
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (1667d = 683h)
  BRR[3:0] = 3h >>1 = 1h
  BRR = 0x681

Example 2

To obtain 921.6 Kbaud with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR = USARTDIV = 52d = 34h
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (104d = 68h)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 8h >> 1 = 4h
  BRR = 0x64

47.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 388*, *Table 389*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 388. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 389. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 388 and Table 389 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

47.5.9 USART Auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is

measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at ‘1’.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a ‘0’).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE=0) during an auto baud rate operation.

47.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

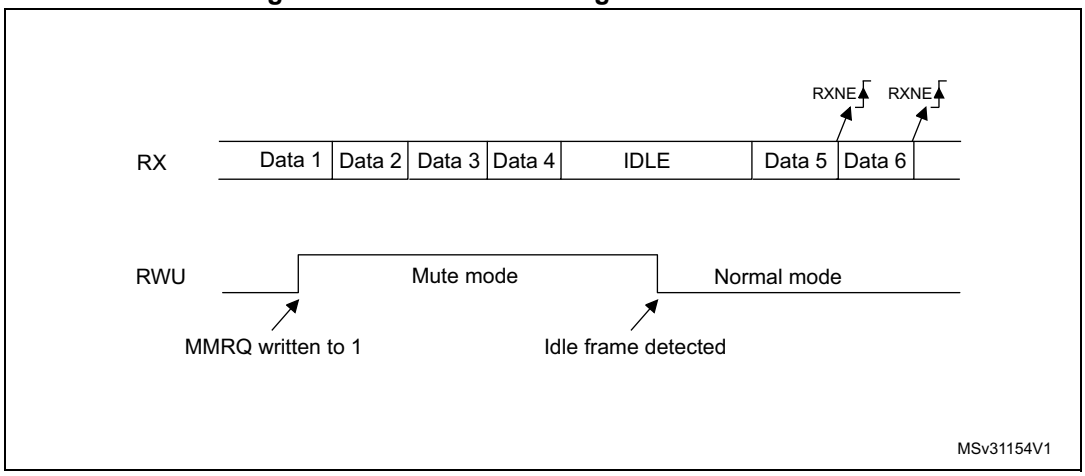
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters Mute mode when the MMRQ bit is written to ‘1’ and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of Mute mode behavior using Idle line detection is given in *Figure 575*.

Figure 575. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode will not be entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

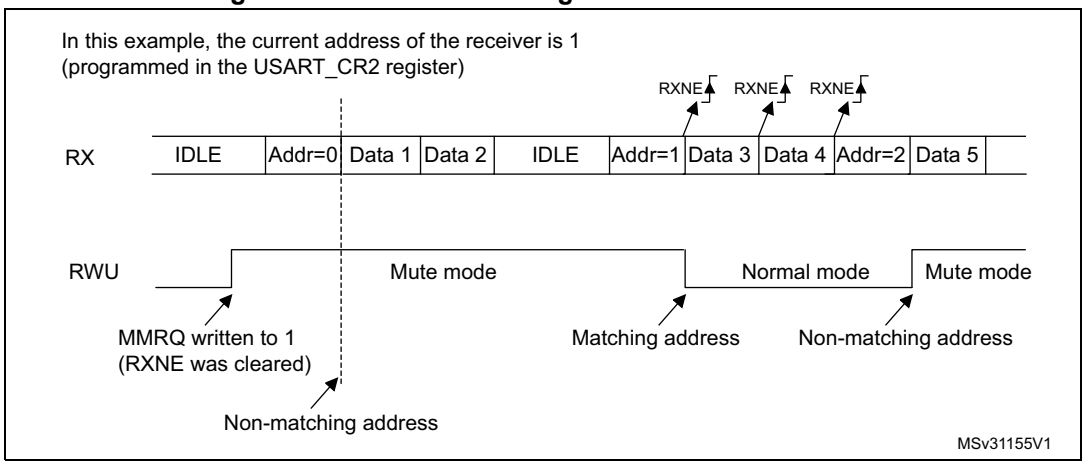
The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in *Figure 576*.

Figure 576. Mute mode using address mark detection



47.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

47.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 390*.

Table 390. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

47.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 47.4: USART implementation on page 2005*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 47.5.4* has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0 bits as a break character. Then 2 bits of value ‘1 are sent to allow the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 577: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 2031*.

Examples of break frames are given on *Figure 578: Break detection in LIN mode vs. Framing error detection on page 2032*.

Figure 577. Break detection in LIN mode (11-bit break length - LBDL bit is set)

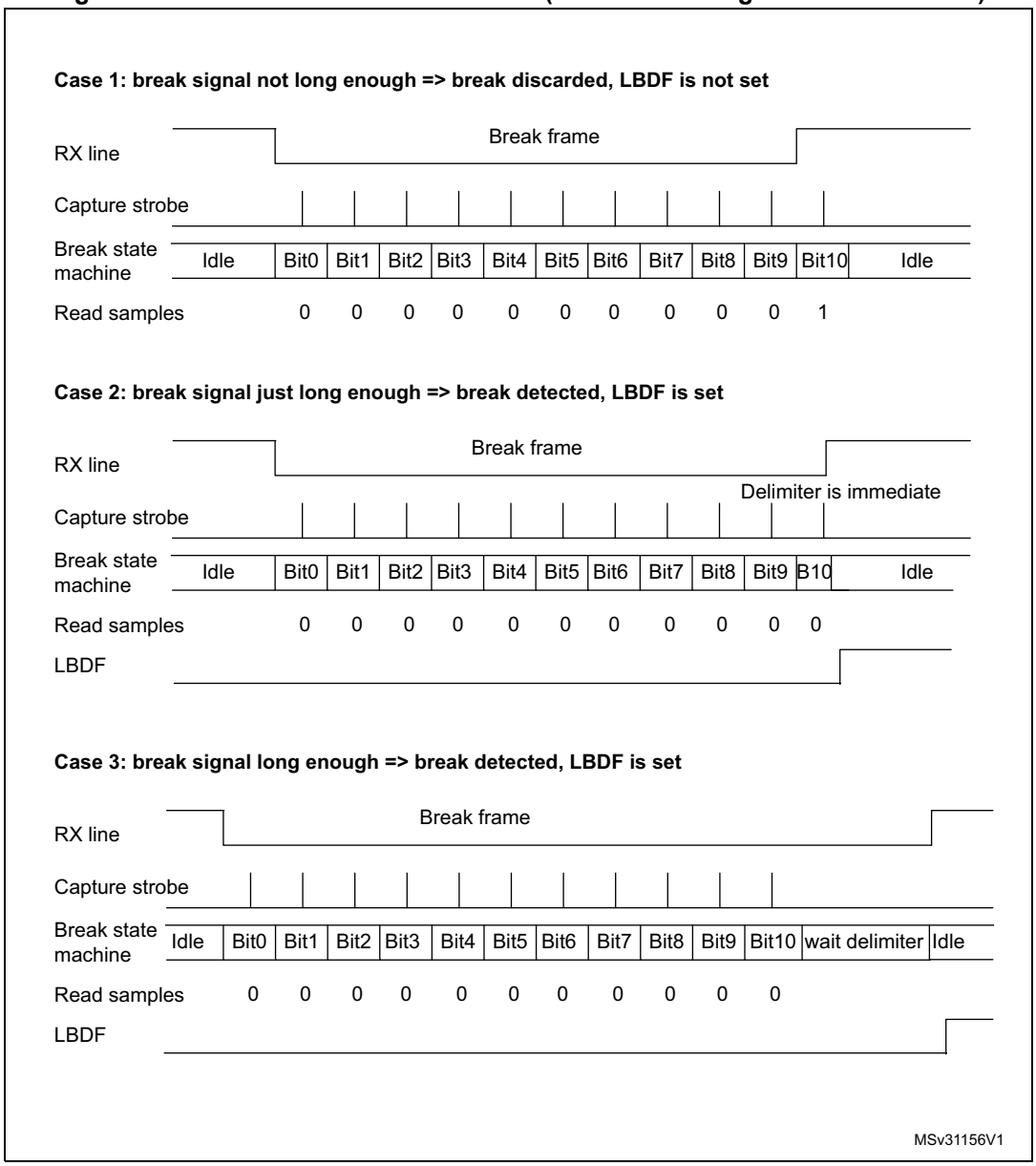
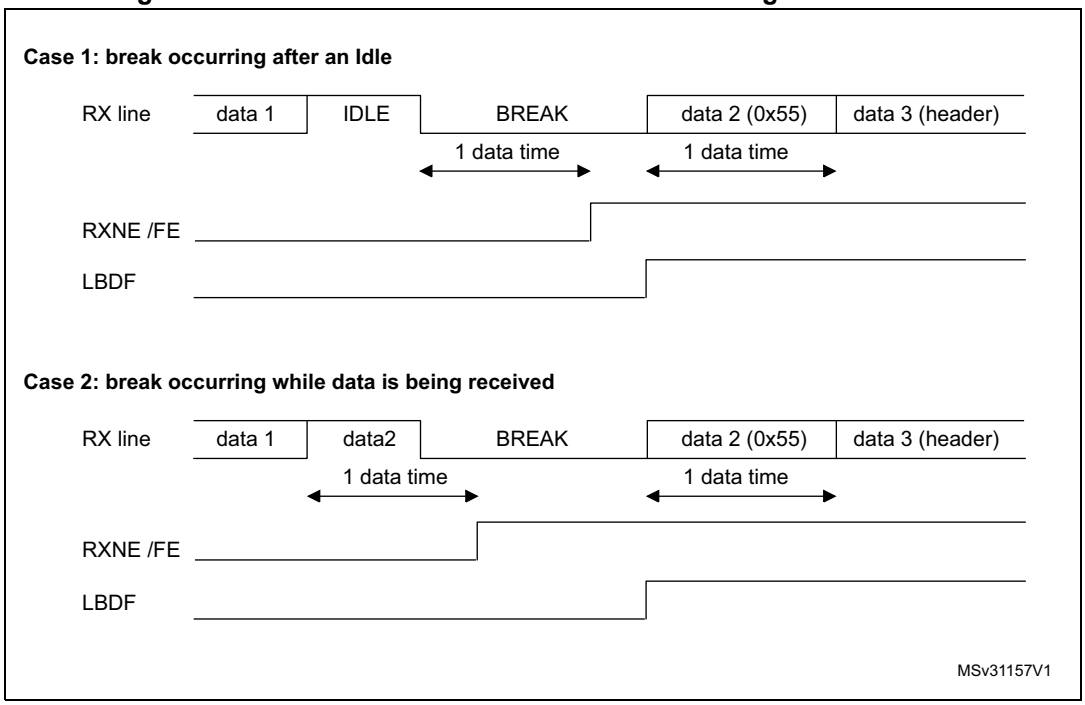


Figure 578. Break detection in LIN mode vs. Framing error detection



47.5.14 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to '1'. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 579*, *Figure 580* and *Figure 581*).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the SCLK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 579. USART example of synchronous master transmission

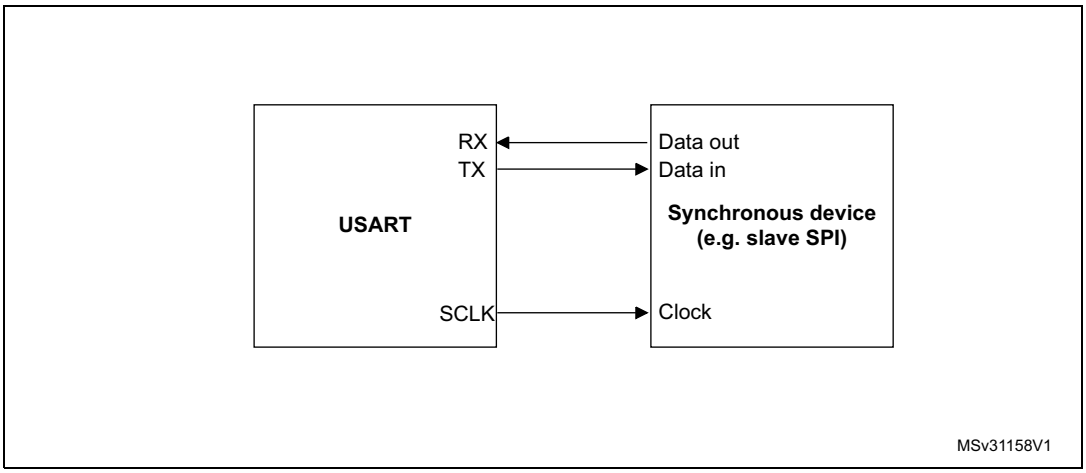


Figure 580. USART data clock timing diagram in synchronous master mode (M bits =00)

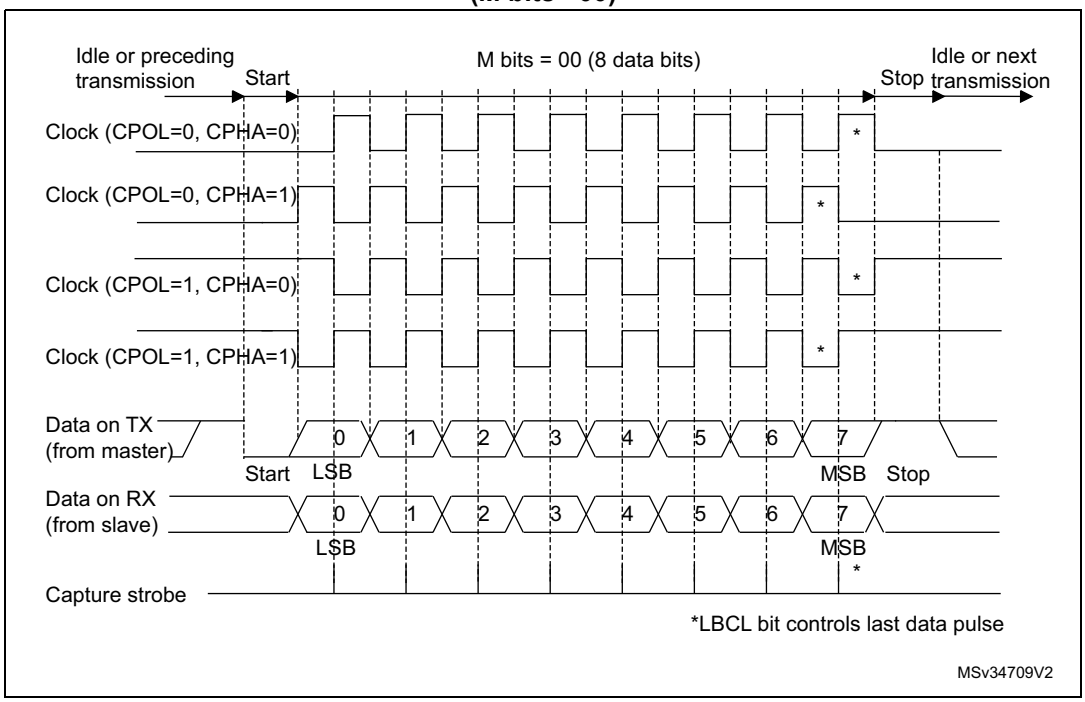
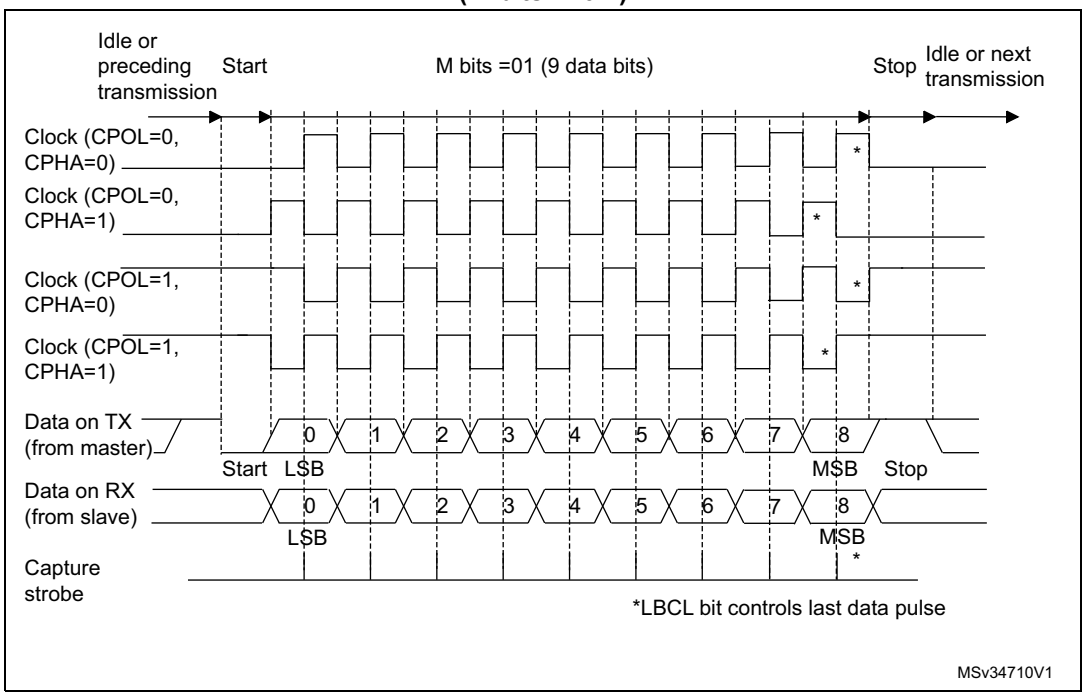


Figure 581. USART data clock timing diagram in synchronous master mode (M bits = '01')



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to '1'. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The SCLK pin is the input of the USART in slave mode.

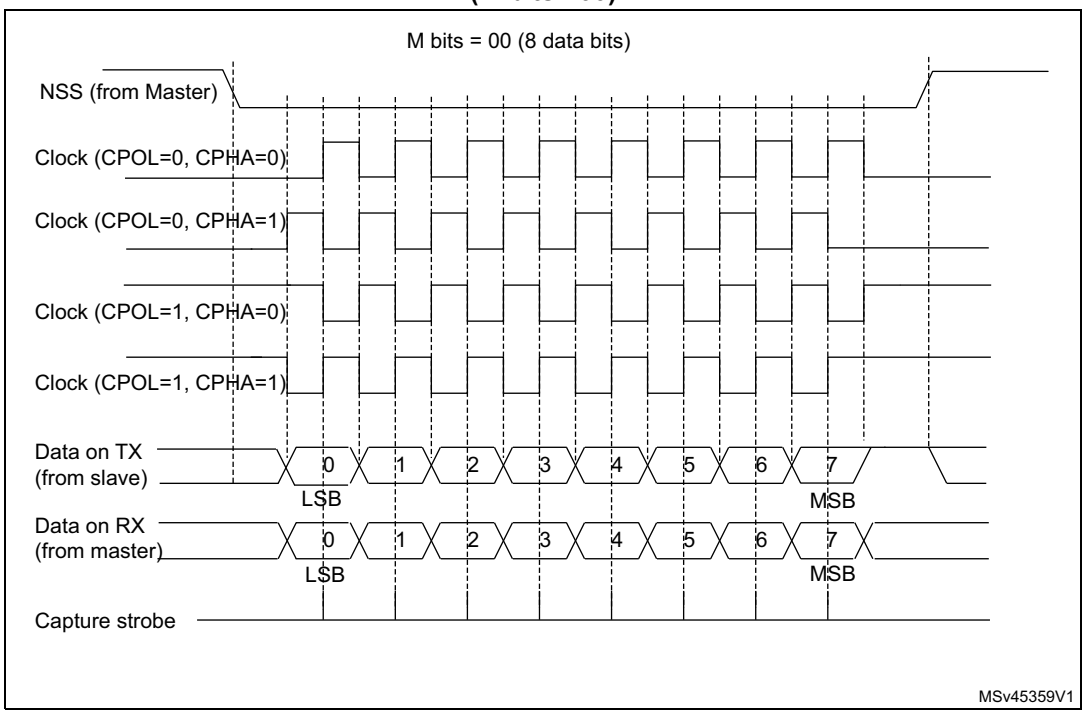
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 582*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 582. USART data clock timing diagram in synchronous slave mode (M bits =00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave will always be selected and NSS input pin will be ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it will become desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave will transmit zeros.

SPI Slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error will allow sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first SCLK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 usart_ker_ck *cycles before the first SCLK edge.*

47.5.15 USART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

47.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = ‘00’ or STOP = ‘11’
- from the end of the second stop bit if STOP = ‘10’.
- from the beginning of the stop bit if STOP = ‘01’.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout will be generated if RTOIE bit in USART_CR1 register is set.

47.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to *Section 47.4: USART implementation on page 2005*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous Smartcard protocol as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

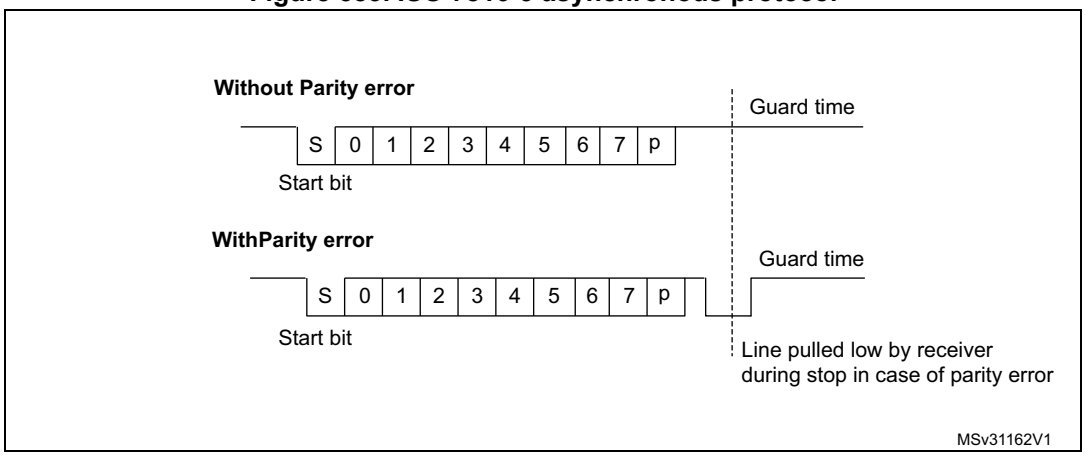
The USART should be configured as:

- 8 bits plus parity: M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP='11' in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 583 shows examples of what can be seen on the data line with and without parity error.

Figure 583. ISO 7816-3 asynchronous protocol



When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

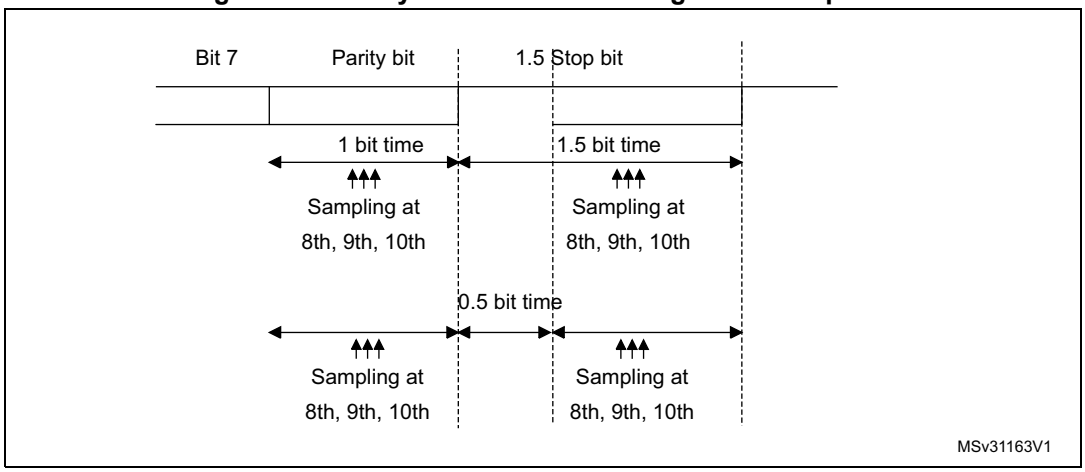
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 584 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 584. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. SCLK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T=1)

In T=1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt will be generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the Smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART will signal it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

Note: When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character will be ‘03’ and the parity will be odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it will be correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103 -> inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

47.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 47.4: USART implementation on page 2005*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 585*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not

encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A ‘0‘ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 586*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to ‘1 stop bit’.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 585. IrDA SIR ENDEC block diagram

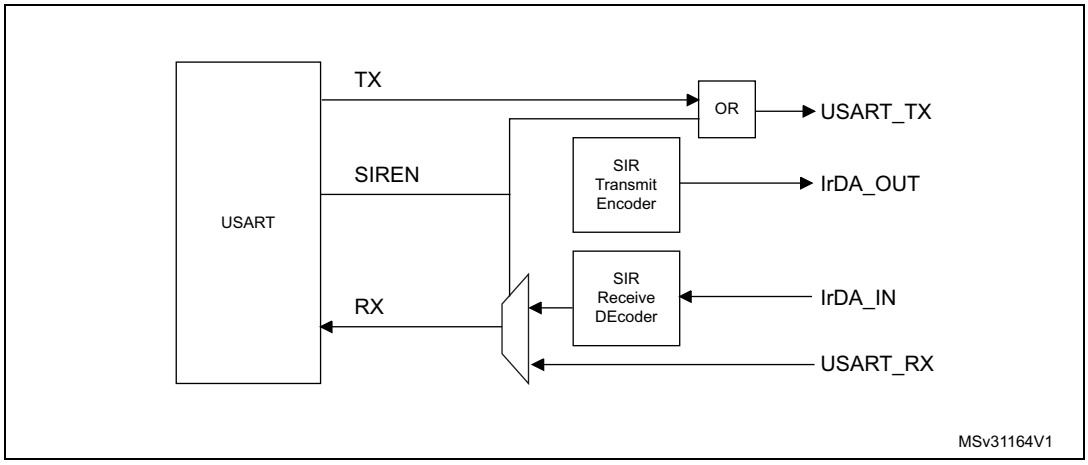
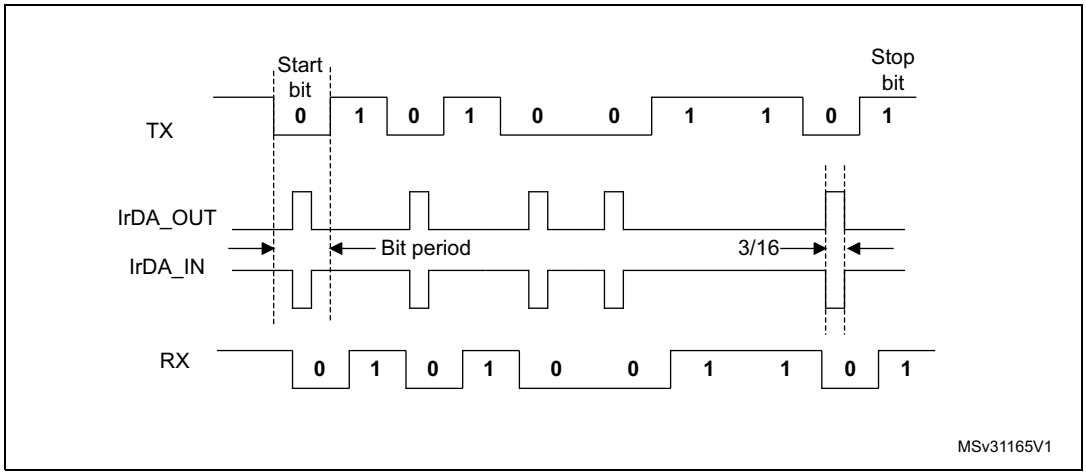


Figure 586. IrDA data modulation (3/16) - Normal mode



47.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 47.4: USART implementation on page 2005 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 47.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

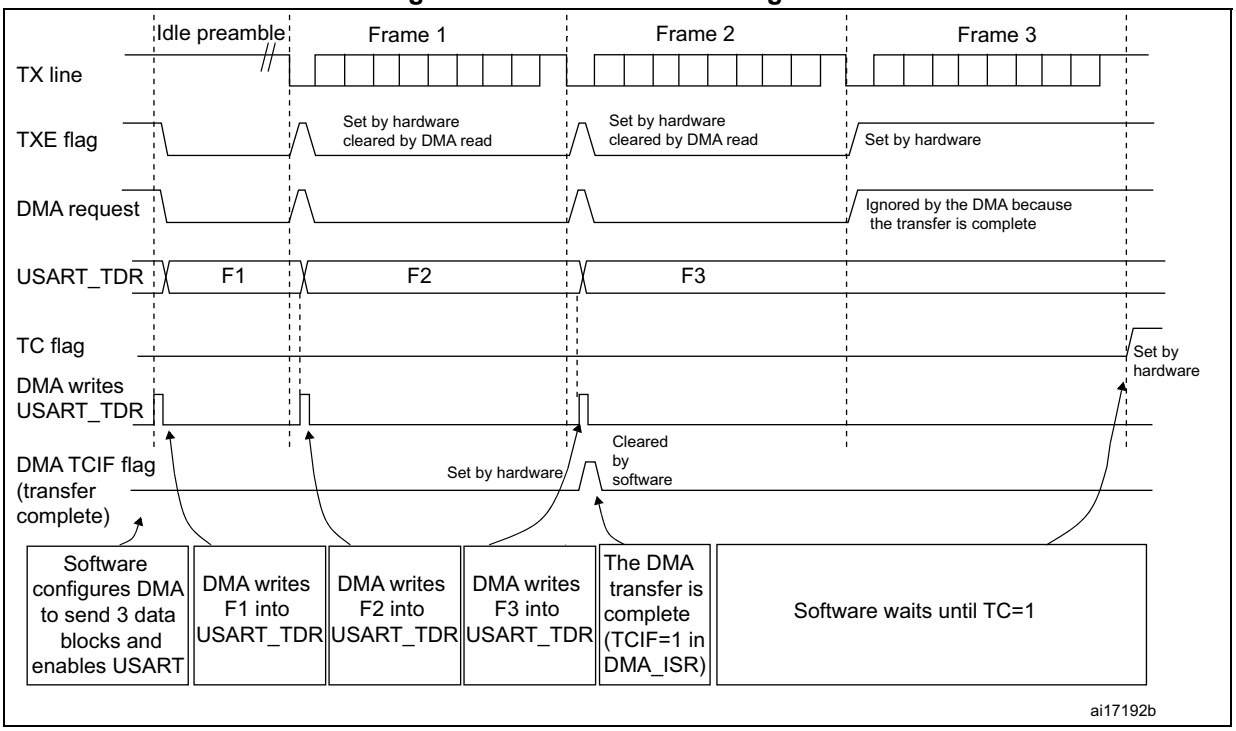
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *Section 14: Direct memory access controller (DMA)* and *Section 11: Direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 587. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

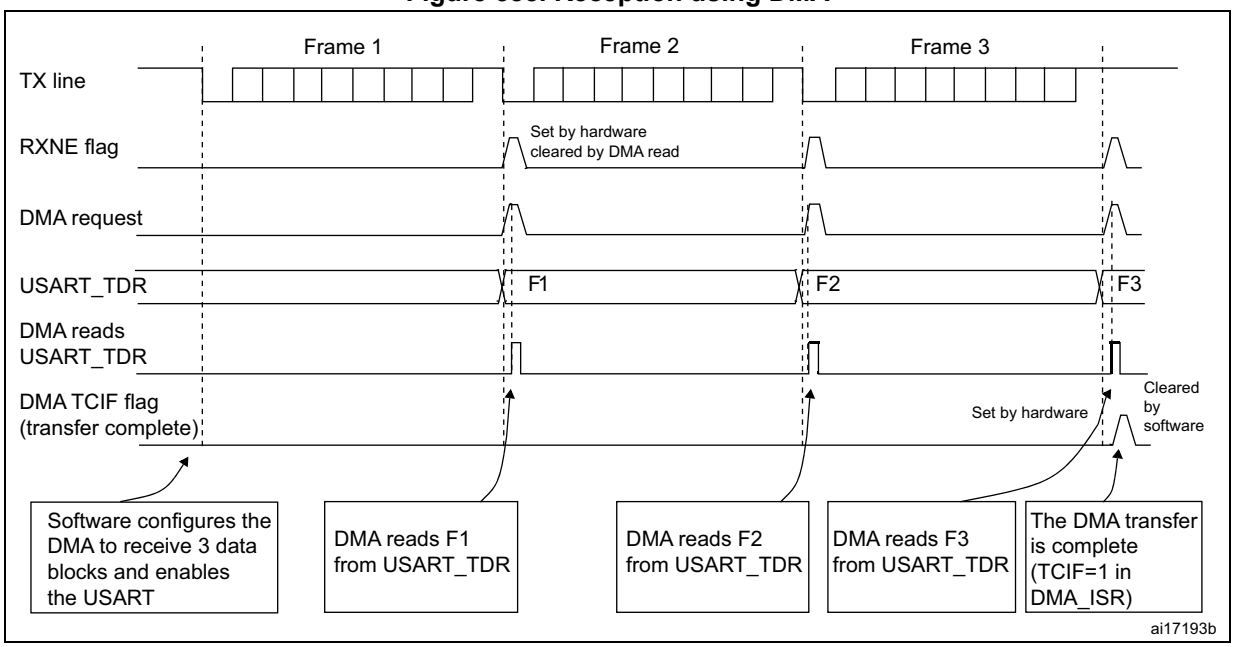
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *Section 14: Direct memory access controller (DMA)* and *Section 11: Direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 588. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

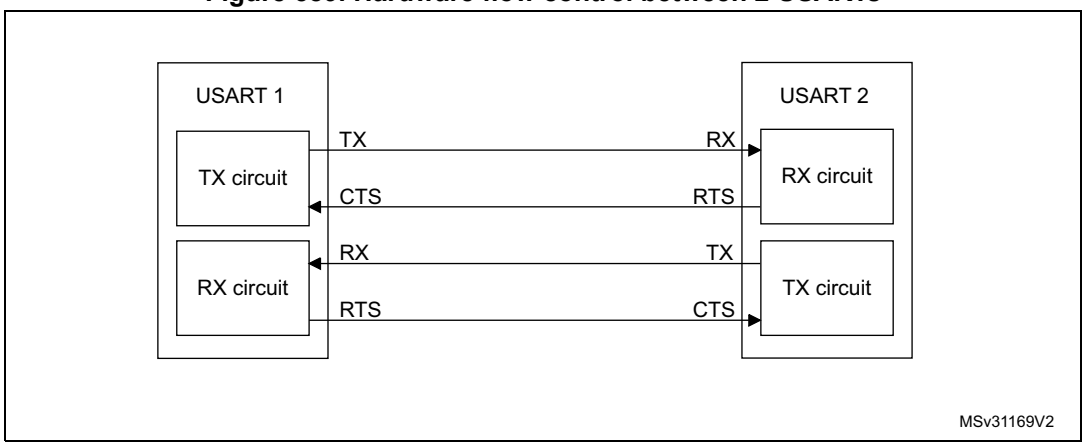
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

47.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 589* shows how to connect 2 devices in this mode:

Figure 589. Hardware flow control between 2 USARTs

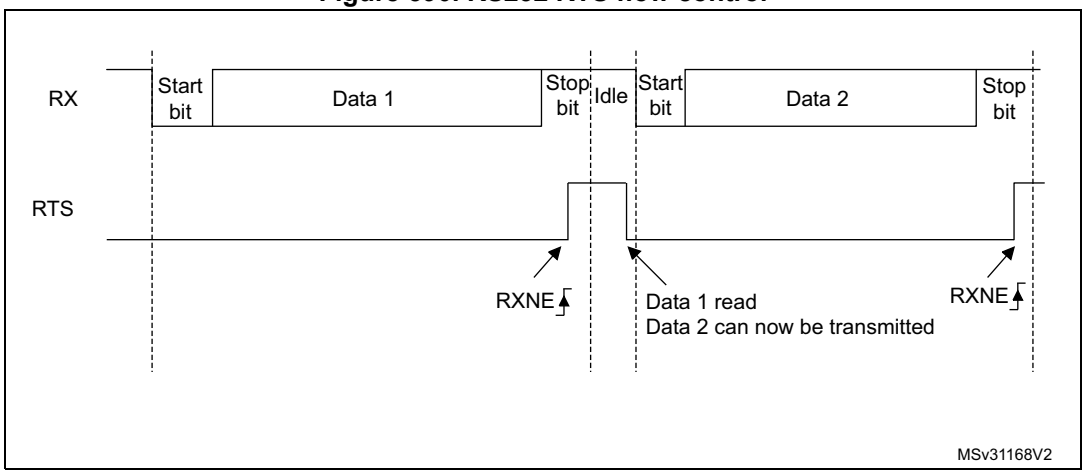


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 590* shows an example of communication with RTS flow control enabled.

Figure 590. RS232 RTS flow control



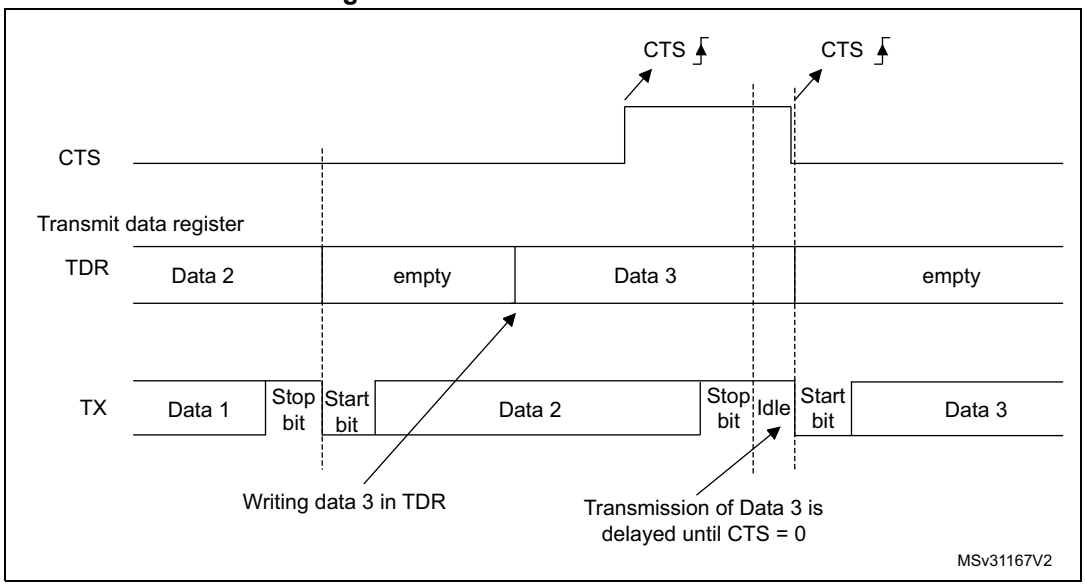
Note: When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 591* shows an example of communication with CTS flow control enabled.

Figure 591. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

47.5.21 USART low-power management

The USART has advanced low-power mode functions, that allow transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wakeup interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to '1'. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - to fill the TXFIFO
  - or to empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This allows sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode allows doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set. In this case the **usart_wkup** interrupt is not mandatory and setting the WUF being is sufficient to wake up the MCU from low-power mode.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the USART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface will remain in Mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, Mute mode can be used with wakeup from low-power mode without any constraints (i.e.the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wakeup event programmed to “address match detection” and FIFO management disabled.

Figure 592 shows the USART behavior when the wakeup event is verified.

Figure 592. Wakeup event verified (wakeup event = address match, FIFO disabled)

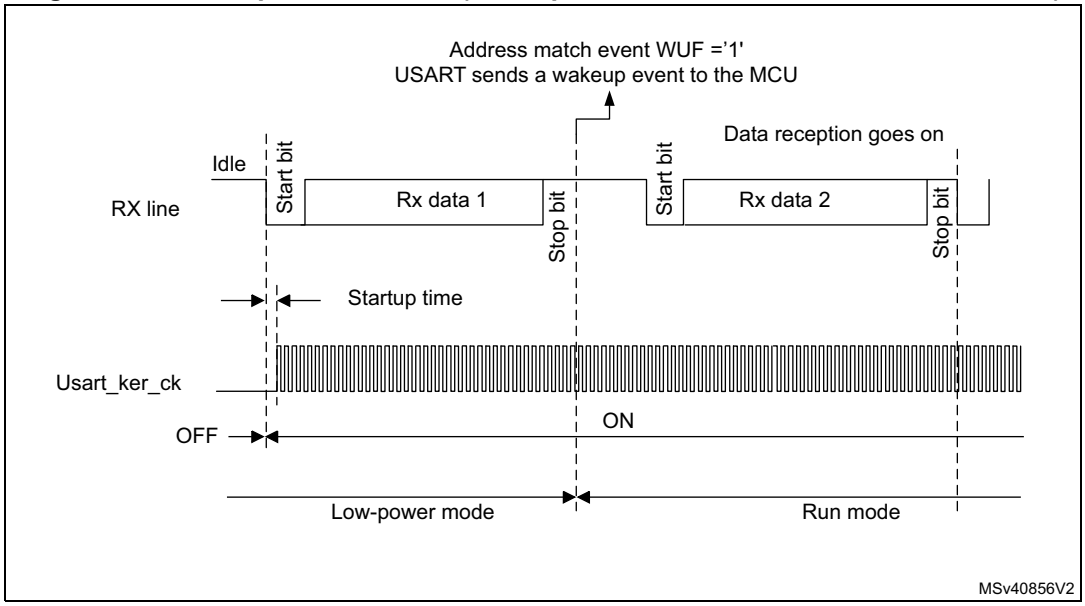
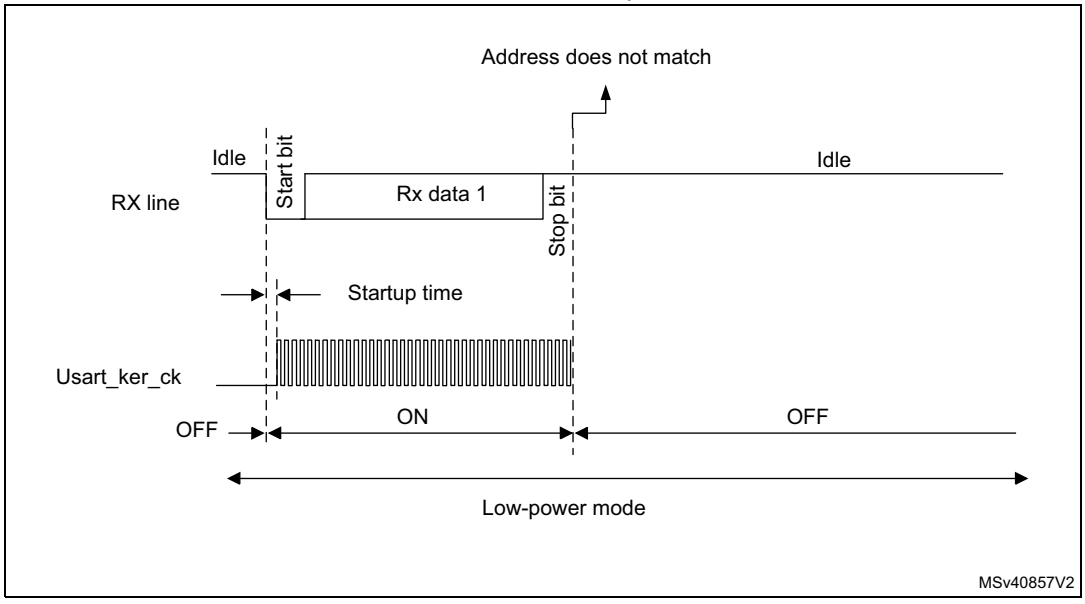


Figure 593 shows the USART behavior when the wakeup event is not verified.

Figure 593. Wakeup event not verified (wakeup event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as wakeup event. If the wakeup event is the start bit detection, the USART sends the wakeup event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that allows to correctly wake up the microcontroller from low-power mode

The maximum baud rate allowing to correctly wake up the microcontroller from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 47.5.8: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 388: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the usart_ker_ck inaccuracy.

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate allowing to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

47.6 USART interrupts

During USART communications, an interrupt (usart_it) can be generated by different events. The USART block can also generate a wakeup interrupt (usart_wkup).

Refer to *Table 391* for a detailed description of all USART interrupt requests.

Table 391. USART interrupt requests

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				usart_it	usart_wkup
Transmit data register empty	TXE	TXEIE	TXE cleared when a data is written in TDR	YES	NO
Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFNF cleared when TXFIFO is full.	YES	NO
Transmit FIFO Empty	TXFE	TXFEIE	TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.	YES	YES

Table 391. USART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				usart_it	usart_wkup
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
CTS interrupt	CTSIF	CTSIE	CTSIF cleared by software by setting CTSCF bit.	YES	NO
Transmission Complete	TC	TCIE	TC cleared when a data is written in TDR or by setting TCCF bit.	YES	NO
Transmission Complete Before Guard Time	TCBGT	TCBGTIE	TCBGT cleared when a data is written in TDR or by setting TCBGTCF bit.	YES	NO
Receive data register not empty (data ready to be read)	RXNE	RXNEIE	RXNE cleared by reading RDR or by setting RXFRQ bit.	YES	YES
Receive FIFO Not Empty	RXFNE	RXFNEIE	RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.	YES	YES
Receive FIFO Full	RXFF(1)	RXFFIE	RXFF cleared when the RXFIFO contains at least one data.	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold	YES	YES
Overrun error detected	ORE	RX-NEIE/RX FNEIE	ORE cleared by setting ORECF bit.	YES	NO
Idle line detected	IDLE	IDLEIE	IDLE cleared by setting IDLECF bit.	YES	NO
Parity error	PE	PEIE	PE cleared by setting PECF bit.	YES	NO
LIN break	LBDF	LBDIE	LBDF cleared by setting LBDCF bit.	YES	NO
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NE or ORE or FE	EIE	NE cleared by setting NFCF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.	YES	NO
Character match	CMF	CMIE	CMF cleared by setting CMCF bit.	YES	NO
Receiver timeout	RTOF	RTOFIE	RTOF cleared by setting RTOCCF bit.	YES	NO

Table 391. USART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				usart_it	usart_wkup
End of Block	EOBF	EOBIE	EOBF is cleared by setting EOBCF bit.	YES	NO
Wakeup from low-power mode	WUF(2)	WUFIE	WUF is cleared by setting WUCF bit.	NO	YES
SPI slave underrun error	UDR	EIE	UDR is cleared by setting UDRCF bit.	YES	NO
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold.	YES	YES

1. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register will not be written and once n data are received and written in the RXFIFO, the RXFF interrupt will be asserted (RXFF flag is not set).
2. The WUF interrupt is active only in low-power mode.

47.7 USART registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

47.7.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE/TXFNFIE	TCIE	RXNEIE/RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when RXFF=1 in the USART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when TXFE=1 in the USART_ISR register

Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE=0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = ‘00’: 1 start bit, 8 Data bits, n Stop bit

M[1:0] = ‘01’: 1 start bit, 9 Data bits, n Stop bit

M[1:0] = ‘10’: 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE=0).

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 47.4: USART implementation on page 2005.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7 **TXEIE/TXFNFIE**: Transmit data register empty/TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE/TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5 **RXNEIE/RXFNEIE**: Receive data register not empty/RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE=1 or RXNE/RXFNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE=1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the SCLK is always available when CLKEN = 1, regardless of the UE bit value.

47.7.2 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:28 **ADD[7:4]**: Address of the USART node

This bitfield gives the address of the USART node or a character code to be recognized.

It is used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or low-power mode. The MSB of the character sent by the transmitter should be equal to 1. It can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

This bitfield can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the USART node

This bitfield gives the address of the USART node or a character code to be recognized.

This is used for wakeup with address mark detection, in multiprocessor communication during Mute mode or low-power mode.

This bitfield can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE=0).

Note: If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 20 **ABREN**: Auto baud rate enable
This bit is set and cleared by software.
0: Auto baud rate detection is disabled.
1: Auto baud rate detection is enabled.
Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 19 **MSBFIRST**: Most significant bit first
This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion
This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion
This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: TX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the TX line.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion
This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the RX line.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins
This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 14 **LINEN**: LIN mode enable
This bit is set and cleared by software.
0: LIN mode disabled
1: LIN mode enabled
The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.
This bitfield can only be written when the USART is disabled (UE=0).
Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 13:12 **STOP[1:0]**: stop bits

These bits are used for programming the stop bits.
00: 1 stop bit
01: 0.5 stop bit.
10: 2 stop bits
11: 1.5 stop bits
This bitfield can only be written when the USART is disabled (UE=0).

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the SCLK pin.
0: SCLK pin disabled
1: SCLK pin enabled
This bit can only be written when the USART is disabled (UE=0).

Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

In Smartcard mode, in order to provide correctly the SCLK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship
0: Steady low value on SCLK pin outside transmission window
1: Steady high value on SCLK pin outside transmission window
This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 573* and *Figure 574*)
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.
0: The clock pulse of the last data bit is not output to the SCLK pin
1: The clock pulse of the last data bit is output to the SCLK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable
Break interrupt mask (break detection using break delimiter).
0: Interrupt is inhibited
1: An interrupt is generated whenever LBDF=1 in the USART_ISR register
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 5 **LBDL**: LIN break detection length
This bit is for selection between 11 bit or 10 bit break detection.
0: 10-bit break detection
1: 11-bit break detection
This bit can only be written when the USART is disabled (UE=0).
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 4 **ADDM7**: 7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the USART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:
When the DSI_NSS bit is set, the NSS pin input is ignored.
0: SPI slave selection depends on NSS input pin.
1: SPI slave is always selected and NSS input pin is ignored.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

47.7.3 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register
Note: If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF=1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
The WUF interrupt is active only in low-power mode.
If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE=0).

If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 14 **DEM**: Driver enable mode

This bit allows the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE=0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 47.4: USART implementation on page 2005.

Bit 13 **DDRE**: DMA Disable on Reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred. (used for Smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO will be bypassed and data will be written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE=0).

Note: This control bit allows checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable

This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE=0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the USART is disabled (UE=0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bitfield can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE=0).

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NE=1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE=1 or ORE=1 or NE=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

47.7.4 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

Bits 3:0 **BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].
When OVER8 = 1:
BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
BRR[3] must be kept cleared.

47.7.5 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE=0).

Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA low-power and normal IrDA mode:

PSC[7:0] = IrDA Normal and Low-Power Baud Rate

Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

In Smartcard mode:

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE=0).

Note: Bits [7:5] must be kept cleared if Smartcard mode is used.

This bitfield is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to Section 47.4: USART implementation on page 2005.

47.7.6 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In Smartcard mode, the Block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of baud clocks.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Refer to *Section 47.4: USART implementation on page 2005**.*

47.7.7 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing '1' to this bit sets the TXE flag. This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE.

This allows to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

47.7.8 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 00 if FIFO is disabled

XX = 28 if FIFO/Smartcard mode is enabled

XX = 08 if FIFO is enabled and Smartcard mode is disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE/TX FNF	TC	RXNE/ RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the USART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the USART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to '101', RXFT flag will be set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART_RDR. Consequently, the 17th received data will not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is '1'. Refer to Section 47.4: USART implementation on page 2005.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.

An interrupt is generated if the RXFFIE bit =1 in the USART_CR1 register.

0: RXFIFO not full.

1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the USART_CR1 register.

0: TXFIFO not empty.

1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE=1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in low-power mode.

If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character will be transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE=1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE will also be set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE=1 in the USART_CR2 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF will be set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 7 **TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When the FIFO mode is disabled, TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

When the FIFO mode is enabled, TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXEIE/TXFNFIE bit =1 in the USART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE will be set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE/TXFE is set.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.

Bit 5 **RXNE/RXFNE**: Read data register not empty/RXFIFO not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXNE/RXFNE is cleared when the RXFIFO is empty. The RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE/RXFNEIE=1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the USART_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 47.5.8: Tolerance of the USART receiver to clock deviation on page 2023).

In FIFO mode, this error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: In FIFO mode, this error is associated with the character in the USART_RDR.

47.7.9 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag
Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag
Writing 1 to this bit clears the UDRF flag in the USART_ISR register.
Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005

Bit 12 **EOBCF**: End of block clear flag
Writing 1 to this bit clears the EOBF flag in the USART_ISR register.
Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 2005.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

47.7.10 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 567*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

47.7.11 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 567*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

47.7.12 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaining combinations: Reserved

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value will be '1011' i.e. input clock divided by 256.

47.7.13 USART register map

The table below gives the USART register map and reset values.

Table 392. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	OVER8	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD1	ABRMOD0	ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3	TXFTCFG			RXFTIE	RXFTCFG			TCBGTIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0

Table 392. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

48 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

48.1 LPUART introduction

The LPUART is an UART which allows bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to allow UART communications up to 9600 baud/s. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports Half-duplex Single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

48.2 LPUART main features

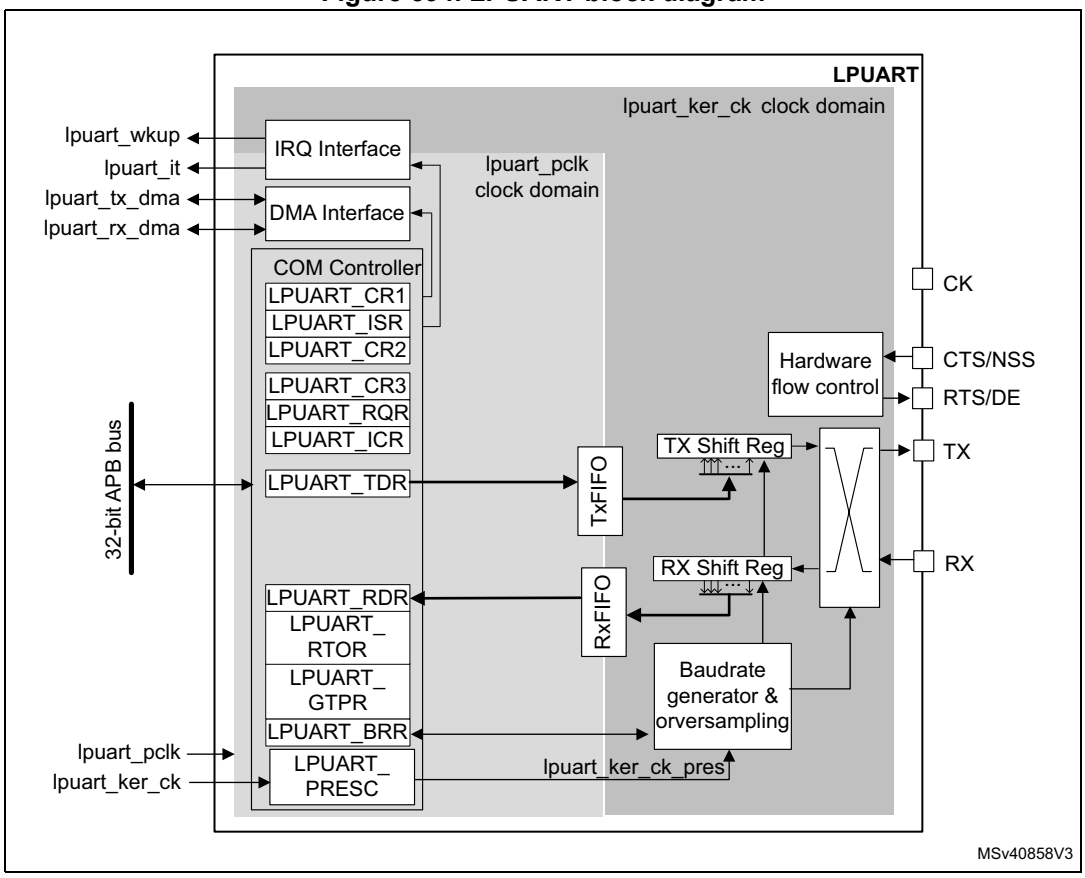
- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud/s to 9600 baud/s using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wakeup from Mute mode by idle line detection or address mark detection

48.3 LPUART functional description

48.3.1 LPUART block diagram

Figure 594. LPUART block diagram



The simplified block diagram given in *Figure 594* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is disabled, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

48.3.2 LPUART signals

LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)
  RX is the serial data input.
- **TX (**Transmit Data Output)
  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire mode, this I/O is used to transmit and receive the data.

RS232 hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS (**Clear To Send)
  When driven high, this signal blocks the data transmission at the end of the current transfer.
- **RTS** (Request to send)
  When it is low, this signal indicates that the USART is ready to receive data.

RS485 hardware flow control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)
  This signal activates the transmission mode of the external transceiver.

Note: DE and RTS share the same pin.

48.3.3 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 568*).

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

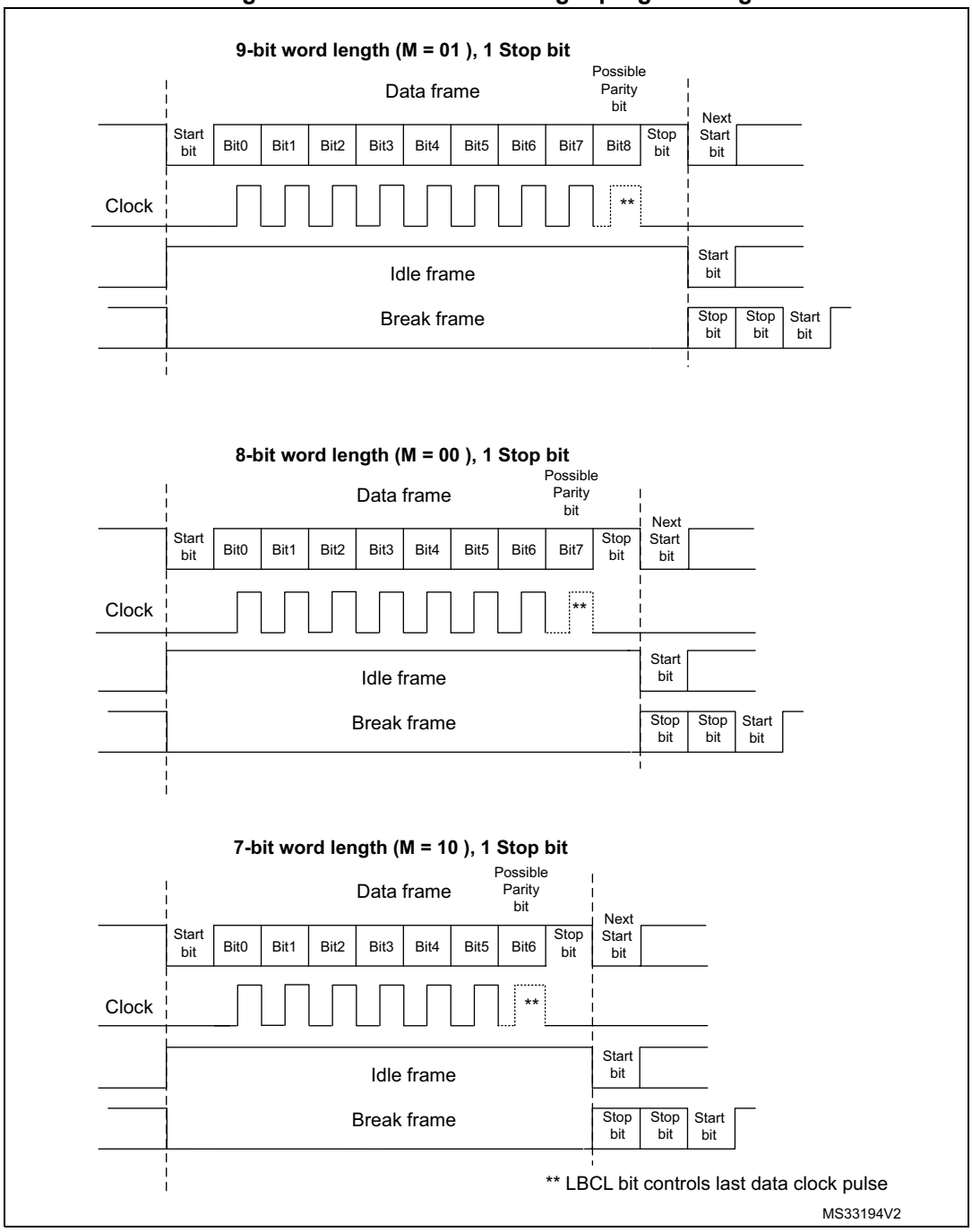
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s will include the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 595. LPUART word length programming



48.3.4 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store

the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG data have been received: 1 data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag will be set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data will not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

48.3.5 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 594*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame will be sent after the TE bit is enabled.

Configurable stop bits

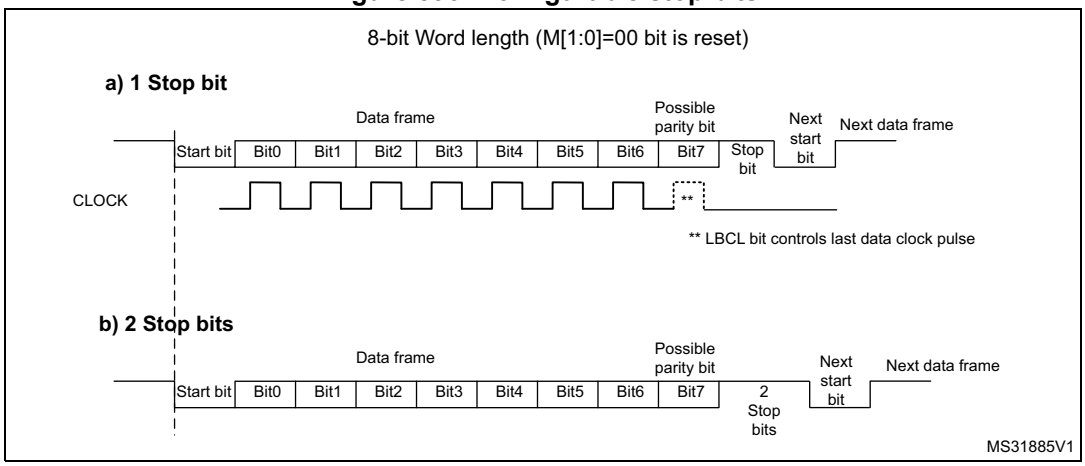
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 Stop bits***: This will be supported by normal LPUART, Single-wire and Modem modes.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when M[1:0] = ‘00’) or 11 low bits (when M[1:0] = ‘01’) or 9 low bits (when M[1:0] = ‘10’) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 596. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAT) in LPUART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in *Section 47.5.10: USART multiprocessor communication*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.

– When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:

  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;

  – the LPUART_TDR register is empty;

  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:

  – the TXFIFO is not full;

  – the LPUART_TDR register is empty;

  – the next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

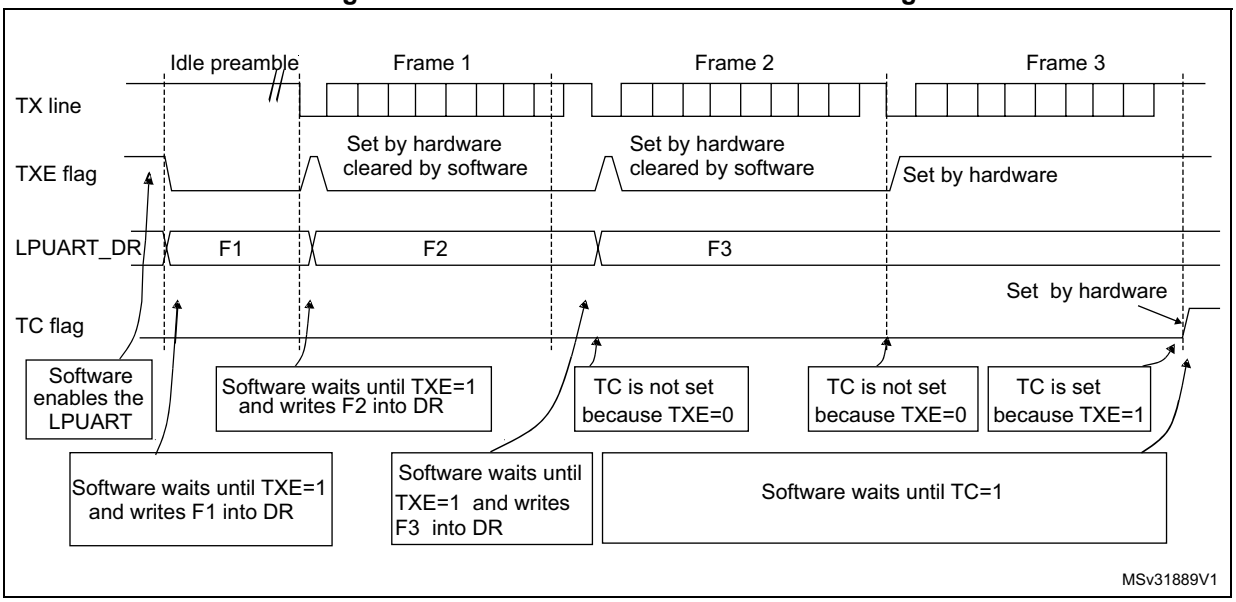
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

  After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 597: TC/TXE behavior when transmitting*).

Figure 597. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 595*).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

48.3.6 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still '0'. If the start sample is at '1', then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 47.5.10: USART multiprocessor communication*.
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content will not be lost. The previous data is available when a read to LPUART_RDR is performed.;
  – the shift register will be overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – the ORE bit is set;
  – the first entry in the RXFIFO will not be lost. It is available when a read to LPUART_RDR is performed.
  – the shift register will be overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE=0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the Clock Control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

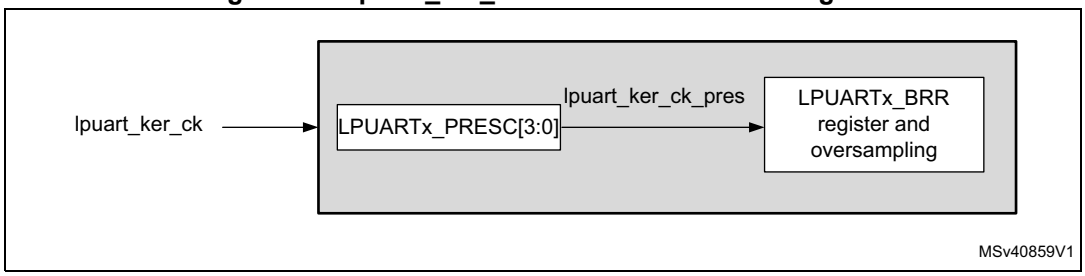
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wakeup from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 598. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources allow the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to allow LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming baud as close as possible to the middle of the baud-period. Only a single sample is taken of each of the incoming bauds.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt will be issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

48.3.7 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuartckpres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value should not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

f_{CK} must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbaud.

Table 393. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32,768 KHz

Baud rate		lpuart_ker_ck_pres= 32,768 KHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	0.3 KBps	0.3 KBps	0x6D3A	0
2	0.6 KBps	0.6 KBps	0x369D	0
3	1200 Bps	1200.087 Bps	0x1B4E	0.007
4	2400 Bps	2400.17 Bps	0xDA7	0.007
5	4800 Bps	4801.72 Bps	0x6D3	0.035
6	9600 KBps	9608.94 Bps	0x369	0.093

Table 394. Error calculation for programmed baud rates at f_{CK} = 100 MHz

Baud rate		f_{CK} = 100MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	38400 Baud	38400,04 Baud	A2C2A	0,0001
2	57600 Baud	57600,06 Baud	6C81C	0,0001
3	115200 Baud	115200,12 Baud	3640E	0,0001
4	230400 Baud	230400,23 Baud	1B207	0,0001
5	460800 Baud	460804,61 Baud	D903	0,001
6	921600 Baud	921625,81 Baud	6C81	0,0028
7	4000 KBaud	4000000,00 Baud	1900	0
8	10000 Kbaud	10000000,00 Baud	A00	0
9	20000 Kbaud	20000000,00 Baud	500	0
10	30000 Kbaud	33032258,06 Baud	307	0,1

48.3.8 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 395*:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 395. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M=00'), 1 Stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M=01'), 1 Stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M='10'), 1 Stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M=00'), 2 Stop bit	2.08%	2.86%	4.35%	4.42%
9 bits (M=01'), 2 Stop bit	1.82%	2.56%	3.90%	4.42%
7 bits (M='10'), 2 Stop bit	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 395 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = '00' (11-bit times when M='01' or 9- bit times when M = '10').

48.3.9 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise Mute mode might remain active.*

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

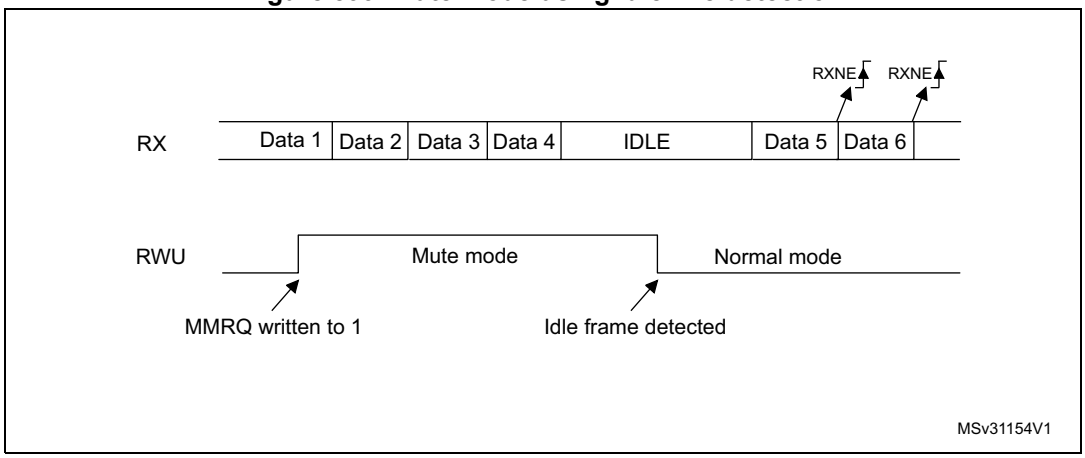
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of Mute mode behavior using Idle line detection is given in *Figure 599*.

Figure 599. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode will not be entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a '1' otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-

bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters Mute mode.

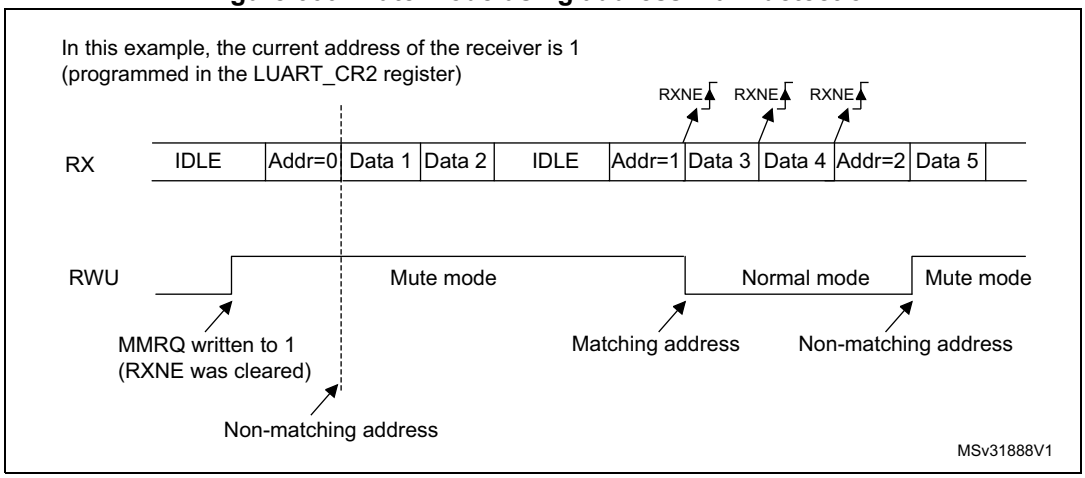
The LPUART also enters Mute mode when the MMRQ bit is written to '1'. The RWU bit is also automatically set in this case.

The LPUART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in Mute mode.

An example of Mute mode behavior using address mark detection is given in *Figure 600*.

Figure 600. Mute mode using address mark detection



48.3.10 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 396*.

Table 396: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |

Table 396: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

48.3.11 LPUART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the LPUART_CR2 register,
- SCEN and IREN bits in the LPUART_CR3 register.

The LPUART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

48.3.12 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 47.4: USART implementation on page 2005 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 47.5.6. To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

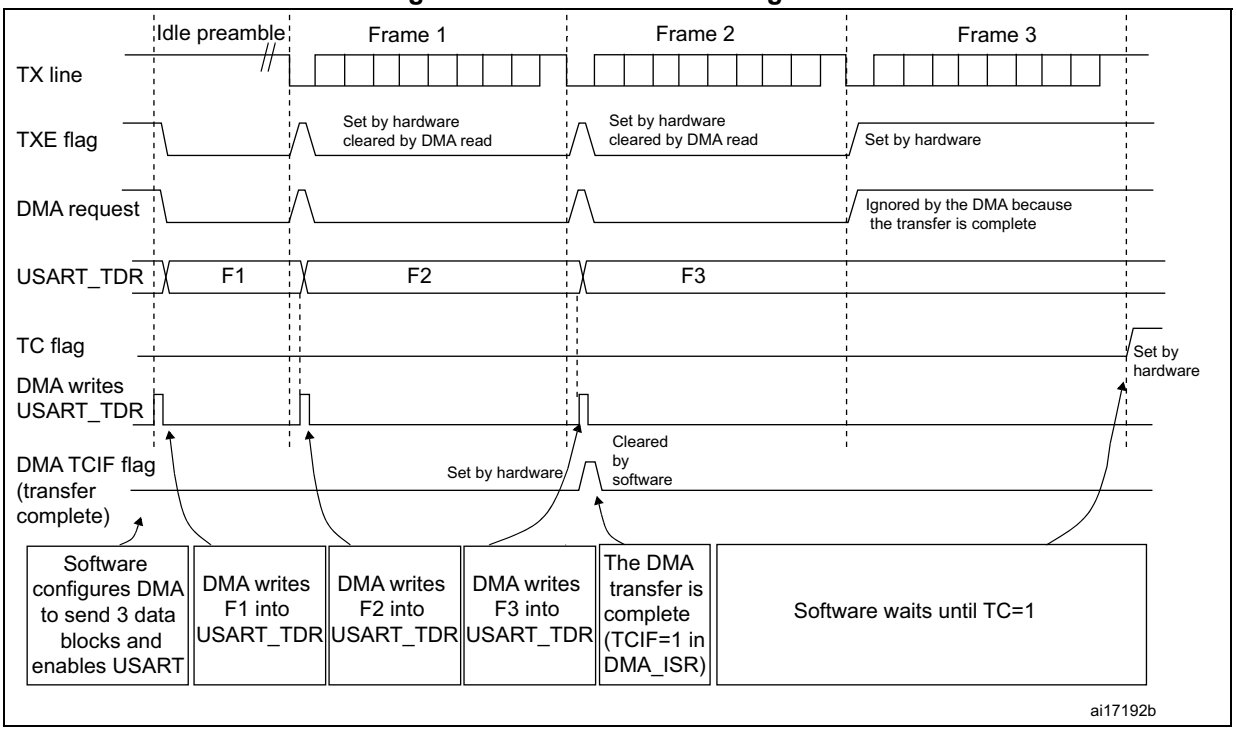
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the *Direct memory access controller sections*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 601. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

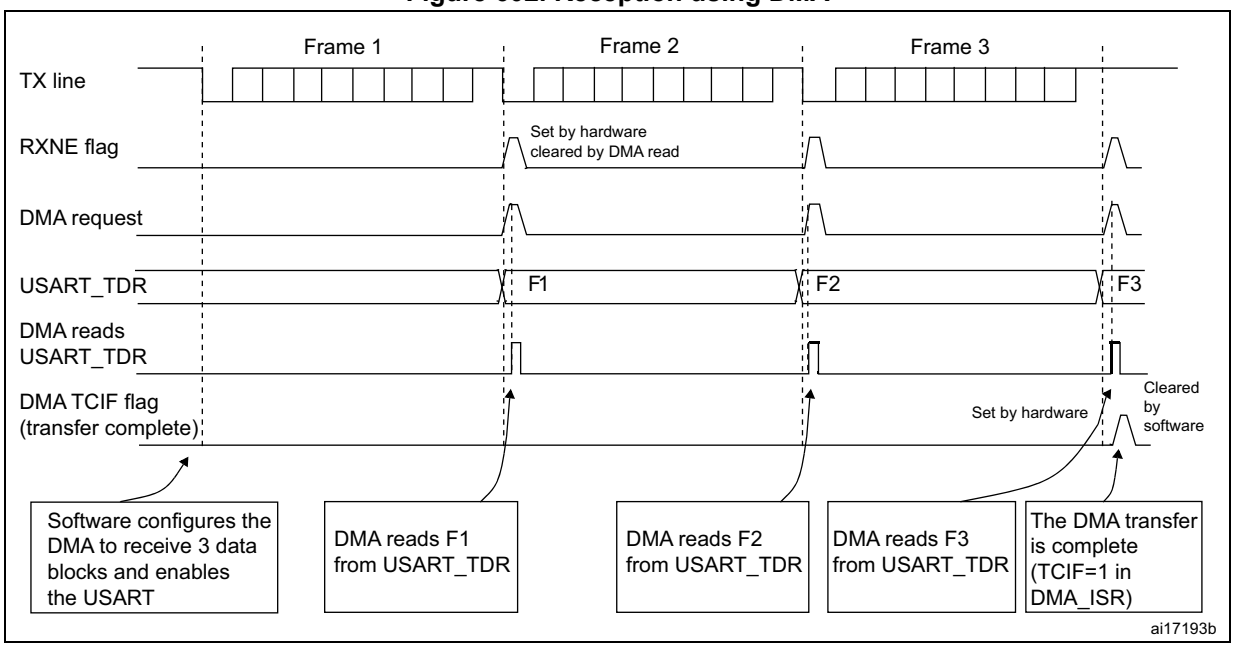
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to the *Direct memory access controller (DMA) sections*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 602. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

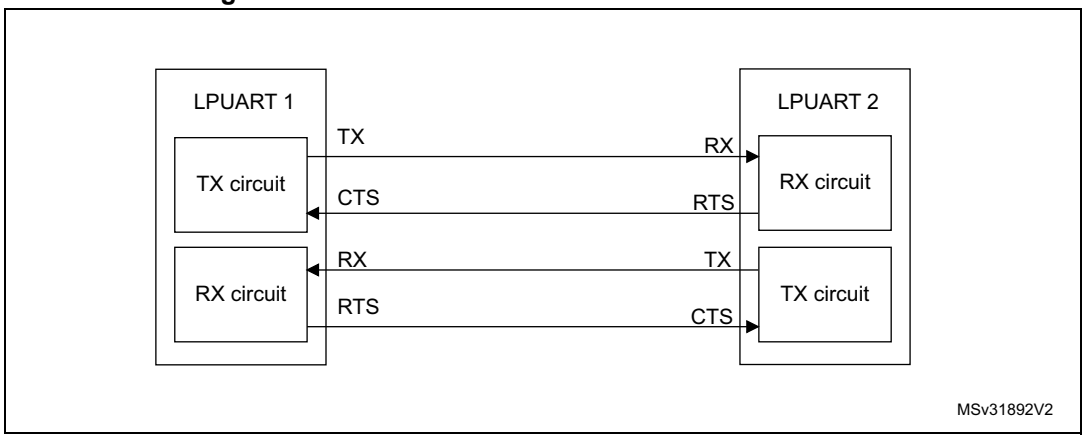
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

48.3.13 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 589* shows how to connect 2 devices in this mode:

Figure 603. Hardware flow control between 2 LPUARTs

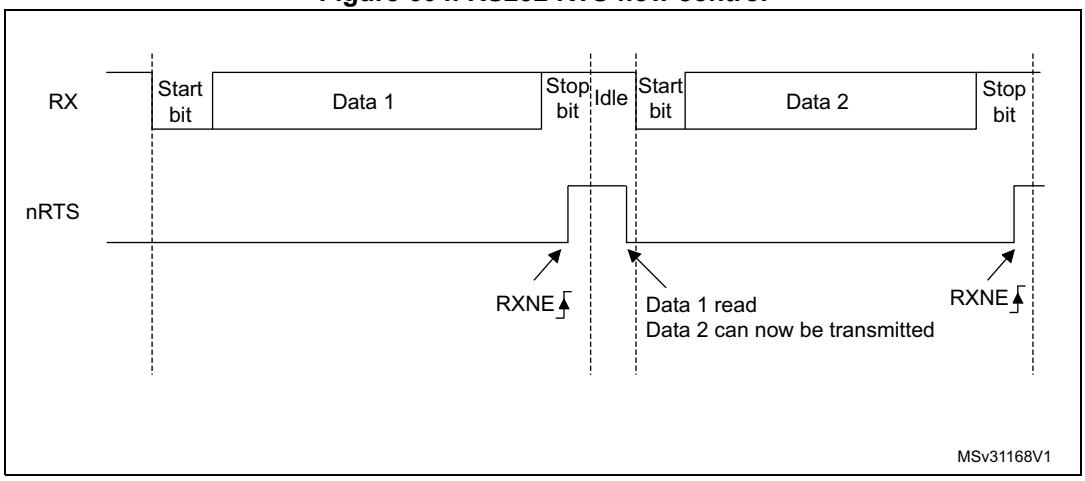


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 604* shows an example of communication with RTS flow control enabled.

Figure 604. RS232 RTS flow control



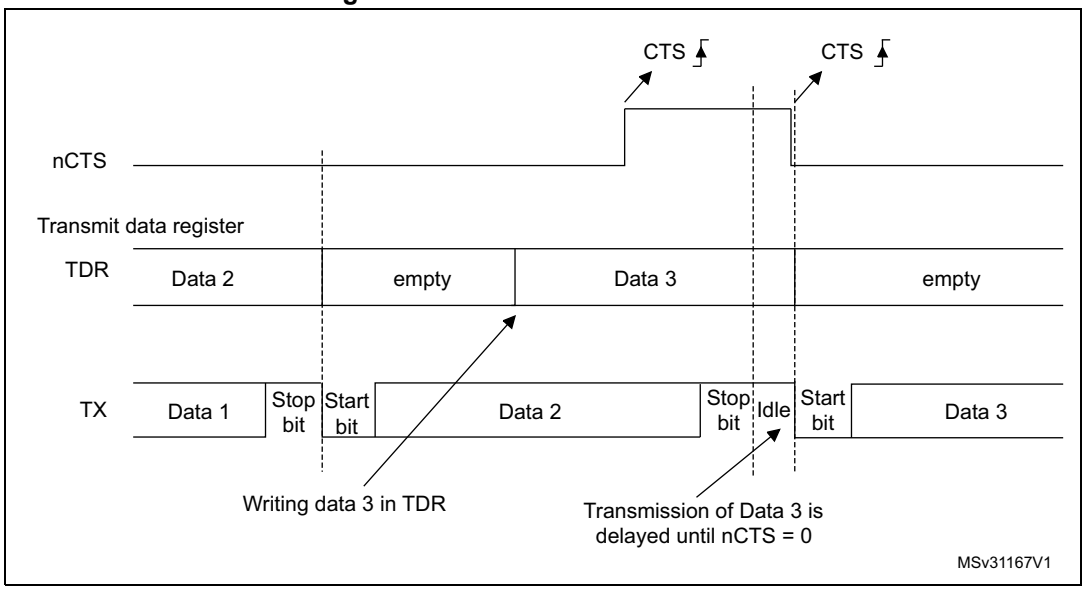
Note: When FIFO mode is enabled, nRTS is de-asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 605* shows an example of communication with CTS flow control enabled.

Figure 605. RS232 CTS flow control



Note: For correct behavior, nCTS must be asserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This allows activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source (f_{CK}) cycles:

- The Driver enable assertion time equals
  - (1 + (DEAT x P)) x f_{CK}, if P # 0
  - (1 + DEAT) x f_{CK} , if P = 0
- The Driver enable de-assertion time equals
  - (1 + (DEDT x P)) x f_{CK}, if P # 0
  - (1 + DEDT) x f_{CK}, if P = 0

where P = BRR[20:11]

48.3.14 LPUART low-power management

The LPUART has advanced low-power mode functions allowing it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the usart_pclk is gated, the LPUART provides a wakeup interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - to fill the TXFIFO
  - or to empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set .
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This allows sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wakeup time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wakeup the MCU from low-power mode allows doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **lpuart_wkup** interrupt may be selected through the WUS bitfields.

When the wakeup event is detected, the WUF flag is set by hardware and **lpuart_wkup** interrupt is generated if the WUFIE bit is set. In this case the **lpuart_wkup** interrupt is not mandatory for the wakeup. The WUF being set is sufficient to wakeup the MCU from low-power mode.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.

When FIFO is enabled, the wakeup from low-power mode on address match is only possible when Mute mode is enabled.

Using Mute mode with low-power mode

If the LPUART is put into Mute mode before entering low-power mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wakeup from Mute mode on address match is used, then the low-power mode wakeup source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface will remain in Mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, Mute mode is used with wakeup from low-power mode without any constraints (i.e.the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wakeup from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wakeup event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wakeup event is not verified, the usart_ker_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wakeup event programmed to “address match detection” and FIFO management disabled.

Figure 606 shows the behavior when the wakeup event is verified.

Figure 606. Wakeup event verified (wakeup event = address match, FIFO disabled)

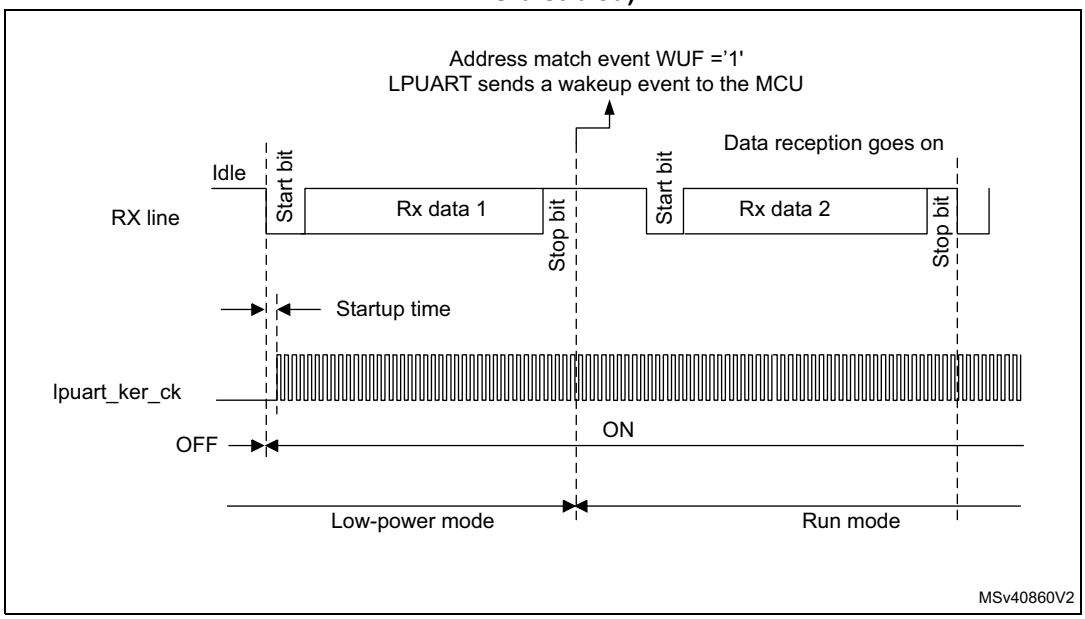
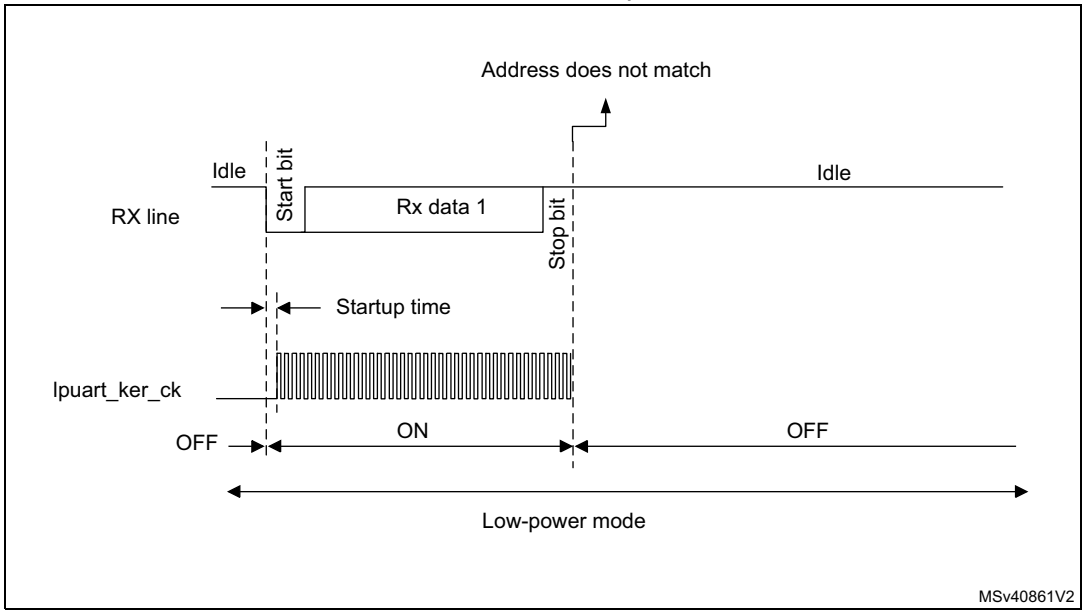


Figure 607 shows the behavior when the wakeup event is not verified.

Figure 607. Wakeup event not verified (wakeup event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wakeup event. In the case the wakeup event is the start bit detection, the LPUART sends the wakeup event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that allows to correctly wake up the MCU from low-power mode

The maximum baud rate allowing to correctly wake up the MCU from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 48.3.8: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 395: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wakeup time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate allowing to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

48.4 LPUART interrupts

Refer to *Table 391* for a detailed description of all LPUART interrupt requests.

Table 397. LPUART interrupt requests

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
Transmit data register empty	TXE	TXEIE	TXE cleared when a data is written in TDR	YES	NO
Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFNF cleared when TXFIFO is full.	YES	NO
Transmit FIFO Empty	TXFE	TXFEIE	TXFE cleared when the TXFIFO contains at least one data or by setting TXFRQ bit.	YES	YES
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES

Table 397. LPUART interrupt requests (continued)

Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Interrupt activated	
				lpuart_it	lpuart_wkup
CTS interrupt	CTSIF	CTSIE	CTSIF cleared by software by setting CTSCF bit.	YES	NO
Transmission Complete	TC	TCIE	TC cleared when a data is written in TDR or by setting TCCF bit.	YES	NO
Receive data register not empty (data ready to be read)	RXNE	RXNEIE	RXNE cleared by reading RDR or by setting RXFRQ bit.	YES	YES
Receive FIFO Not Empty	RXFNE	RXFNEIE	RXFNE cleared when the RXFIFO is empty or by setting RXFRQ bit.	YES	YES
Receive FIFO Full	RXFF(1)	RXFFIE	RXFF cleared when the RXFIFO contains at least one data.	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold	YES	YES
Overrun error detected	ORE	RX-NEIE/RX FNEIE	ORE cleared by setting ORECF bit.	YES	NO
Idle line detected	IDLE	IDLEIE	IDLE cleared by setting IDLECF bit.	YES	NO
Parity error	PE	PEIE	PE cleared by setting PECF bit.	YES	NO
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NE or ORE or FE	EIE	NE cleared by setting NFCF bit. ORE cleared by setting ORECF bit. FE flag cleared by setting FECF bit.	YES	NO
Character match	CMF	CMIE	CMF cleared by setting CMCF bit.	YES	NO
Wakeup from low-power mode	WUF(2)	WUFIE	WUF is cleared by setting WUCF bit.	NO	YES
Transmit FIFO threshold reached	TXFT	TXFTIE	TXFT is cleared by hardware when the TXFIFO content is less than the programmed threshold	YES	YES
Receive FIFO threshold reached	RXFT	RXFTIE	RXFT is cleared by hardware when the RXFIFO content is less than the programmed threshold.	YES	YES

1. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register will not be written and once n data are received and written in the RXFIFO, the RXFF interrupt will be asserted (RXFF flag is not set).

2. The WUF interrupt is active only in low-power mode.

48.5 LPUART registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

48.5.1 Control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE/TXFNFIE	TCIE	RXNEIE/RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register
Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register
Note: When FIFO mode is disabled, this bit is reserved and must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = ‘00’: 1 Start bit, 8 Data bits, n Stop bit
M[1:0] = ‘01’: 1 Start bit, 9 Data bits, n Stop bit
M[1:0] = ‘10’: 1 Start bit, 7 Data bits, n Stop bit
This bit can only be written when the LPUART is disabled (UE=0).
Note: In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 47.5.20: RS232 Hardware flow control and RS485 Driver Enable*.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 48.3.13: RS232 Hardware flow control and RS485 Driver Enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE=0).

Bit 11 **WAKE**: Receiver wakeup method

This bit determines the LPUART wakeup method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXEIE/TXFNFIE**: Transmit data register empty/TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever TXE/TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXNEIE/RXFNEIE**: Receive data register not empty/RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever ORE=1 or RXNE/RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.
When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

48.5.2 Control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:28 **ADD[7:4]**: Address of the LPUART node

This bitfield gives the address of the LPUART node or a character code to be recognized.
It is used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during Mute mode or Stop mode. The MSB of the character sent by the transmitter should be equal to 1. It can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.
This bitfield can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the LPUART node

This bitfield gives the address of the LPUART node or a character code to be recognized.
This is used for wakeup with address mark detection in multiprocessor communication during Mute mode or low-power mode.
This bitfield can only be written when reception is disabled (RE = 0) or the LPUART is disabled (UE=0)

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: TX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This allows the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: STOP bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

48.5.3 Control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFTCFG			RXFTIE	RXFTCFG			Res.	TXFTIE	WUFIE	WUS[2:0]		Res.	Res.	Res.	Res.
rw			rw	rw				rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HD SEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Remaining combinations: Reserved.

Bit28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wakeup from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever WUF=1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
The WUF interrupt is active only in low-power mode.
If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wakeup from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (Wakeup from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01:Reserved.
10: WUF active on Start bit detection
11: WUF active on RXNE.
This bitfield can only be written when the LPUART is disabled (UE=0).

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode
This bit allows the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on Reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE=0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 :**OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE=0).
Note: This control bit allows checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.
This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half duplex mode is not selected
1: Half duplex mode is selected
This bit can only be written when the LPUART is disabled (UE=0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.

48.5.4 Baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]**

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care should be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].

48.5.5 Request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This will set the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This allows discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

48.5.6 Interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 00 if FIFO is disabled

XX = 08 if FIFO is enabled and Smartcard mode is disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE/TXFNF	TC	RXNE/RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit =1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from low-power mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

The WUF interrupt is active only in low-power mode.

If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value

Bit 19 **RWU**: *Receiver wakeup from Mute mode*

This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character will be transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1in the LPUART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE/TXFNF:** Transmit data register empty/TXFIFO not full

When FIFO mode is disabled, TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

When FIFO mode is enabled, TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO. (TXFNF and TXFE will be set at the same time).

An interrupt is generated if the TXEIE/TXFNFIE bit =1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE/TXFF is set. An interrupt is generated if TCIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART_ICR register or by a write to the LPUART_TDR register.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.

Bit 5 **RXNE/RXFNE**: Read data register not empty/RXFIFO not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXNE/RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE/RXFNEIE=1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE/ RXFNEIE=1 or EIE = 1 in the LPUART_CR1 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR1 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

48.5.7 Interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from low-power mode clear flag
Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.
Note: If the LPUART does not support the wakeup from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag
Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

48.5.8 Receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 594*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

48.5.9 Transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 594*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.

48.5.10 Prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaining combinations: Reserved.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value will be «1011» i.e. input clock divided by 256.

48.5.11 LPUART register map

The table below gives the LPUART register map and reset values.

Table 398. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	Res.	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3	TXFTCFG			RXFTIE	RXFTCFG			Res.	TXFTIE	WUFIE	WUS[1:0]		Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0	0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																											0	0	0	0		
0x1C	LPUART_ISR	Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE/TXFNF	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0		0	0	0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 398. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	LPUART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

49 Serial peripheral interface (SPI)

49.1 Introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The (SPI) interface supports a half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and is capable of operating in multi slave or multi master configurations. In case of master configuration it provides the communication clock (SCK) to the external slave device. The slave select signal can be provided by the master and accepted by the slave optionally, too. The Motorola data format is used by default, but some other specific modes are supported as well.

49.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4-bit to 32-bit data size selection
- Multi master or multi slave mode capability
- Dual clock domain, separated clock for the peripheral kernel which can be independent of PCLK
- 8 master mode baud rate prescalers up to kernel frequency/2
- Slave mode frequency up to kernel frequency/2
- Protection of configuration and setting
- Hardware or software management of SS for both master and slave
- Adjustable minimum delays between data and between SS and data flow
- Configurable SS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transaction to control SS and CRC
- Dedicated transmission and reception flags with interrupt capability
- Slave's transmission and/or reception capability in Stop mode (no clock provided to the peripheral) with wake up
- SPI Motorola and TI formats support
- Hardware CRC feature can secure communication at the end of transaction by:
  - Adding CRC value at Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun at slave, mode fault at master
- Two 16x or 8x 8-bit embedded Rx and TxFIFOs with DMA capability
- Programmable number of data in transaction
- Configurable FIFO thresholds (data packing)
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)

49.3 SPI implementation

Table 399. STM32H7xx SPI features

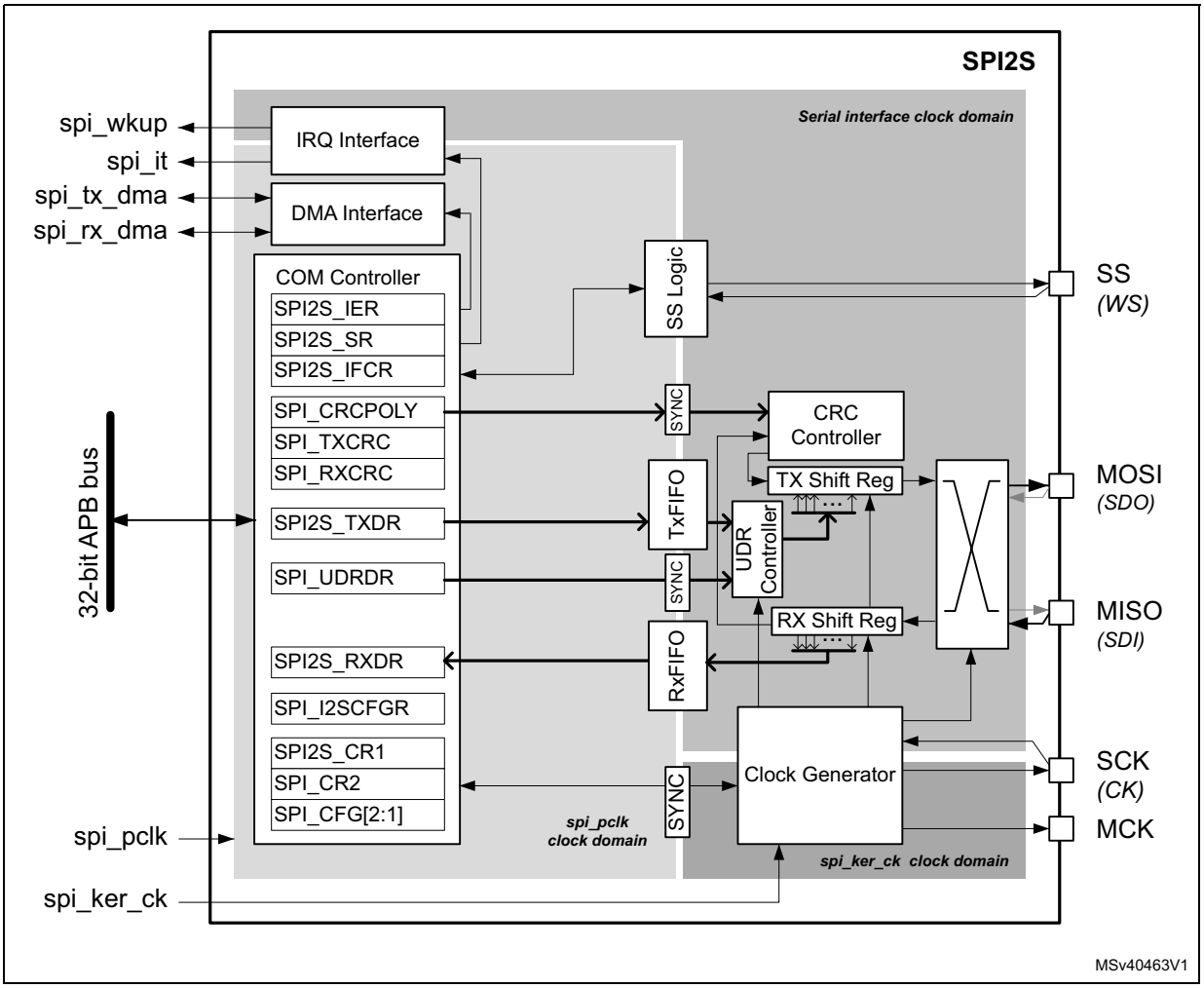
SPI modes/features	SPI2S1	SPI2S2	SPI2S3	SPI4	SPI5	SPI6
Rx & TxFIFO size (N) [x 8-bit]	16	16	16	8	8	8
Maximum configurable data size [bits]	32	32	32	16	16	16
I2S feature	Yes	Yes	Yes	No	No	No

49.4 SPI functional description

49.4.1 SPI block diagram

The SPI allows a synchronous, serial communication between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram at *Figure 608*.

Figure 608. SPI2S block diagram



The simplified scheme of *Figure 608* shows three fully independent clock domains:

- The **spi_pclk** clock domain,
- The **spi_ker_ck** kernel clock domain,
- The serial interface clock domain,

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed in order to avoid any data underrun or overrun events only.

The **spi_pclk** clock signal feeds the peripheral bus interface. It has to be active when it accesses to the SPI registers are required.

The SPI working in slave mode handles data flow using the serial interface clock derived from the external SCK signal provided by external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive.

This is not the case for the SPI master as it needs an active **spi_ker_ck** kernel clock coming from the RCC to feed the clock generator at least. On the other side, a specific slave logic working within the serial interface clock domain needs some additional traffic to be setup correctly (e.g. when underrun or overrun is evaluated). This cannot be done when the bus becomes into idle. At specific case the slave even requires the clock generator working (see *Section 49.5.1: TI mode*).

49.4.2 SPI signals

Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **SS:** Slave select pin. Depending on the SPI and SS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame or
  - Detect a conflict between multiple masters

  See *Section 49.4.6: Multi-master communication* for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management. the functionality between MOSI and MISO pins can be inverted in any SPI mode (see the IOSWP bit at SPI_CFG2 register).

49.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software SS management) or 3/4 wires (with hardware SS management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slave(s) for communication by SS line when it is managed by HW. The data between the master and the slave, flow on the MOSI and/or MISO lines.

49.4.4 Communications between one master and one slave

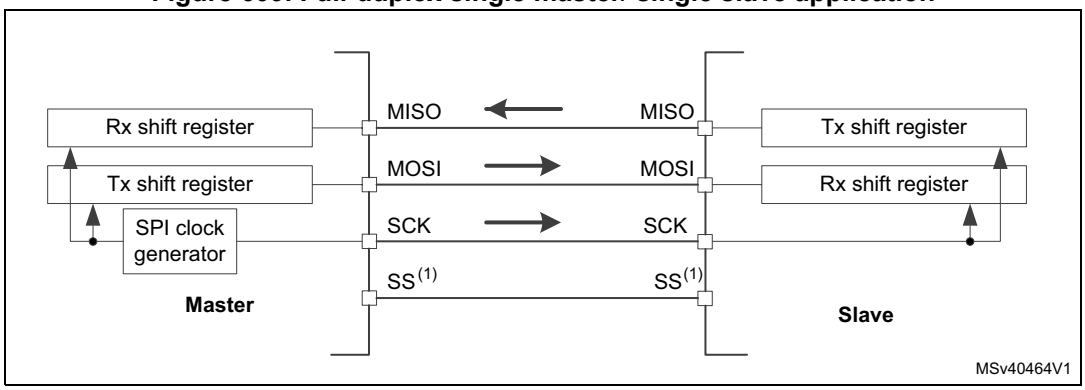
The communication flow may use one of 3 possible modes: full-duplex (3 wires), half-duplex (2 wires) or simplex (2 wires). The SS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the SS

signal can be helpful at this configuration to synchronize the data flow and it is used by default at some specific SPI modes (e.g. TI mode).

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0]=00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 609. Full-duplex single master/ single slave application

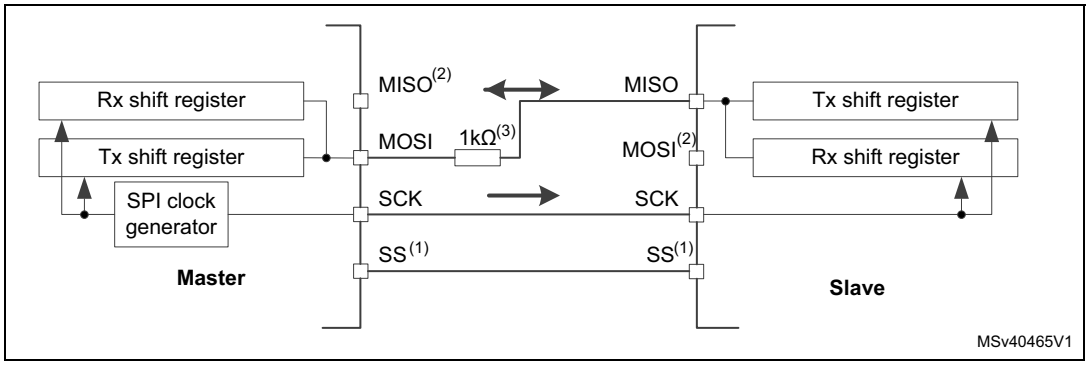


1. The SS pin is configured at output mode at master. The pins can be left unconnected then SS is managed by software internally on both master and slave side.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0]=11 in the SPI_CFG2 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI has to be disabled when changing direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 610. Half-duplex single master/ single slave application



1. The SS pin is configured at output mode at master. The pins can be left unconnected then SS is managed by software internally on both master and slave side.
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectional mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communicated data). Both nodes can fight with opposite outputs levels on the line temporary till next node change its direction setting correspondingly, too. It is suggested to insert serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation,

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pins pair is not used for communication and can be used as standard GPIOs.

- Transmit-only mode: COMM[1:0]=01

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is on-going.

  The slave in transmit only mode sends data as long as it receives a clock on the SCK pin and the SS pin (or SW managed internal signal) is active (see *49.4.6: Multi-master communication*).

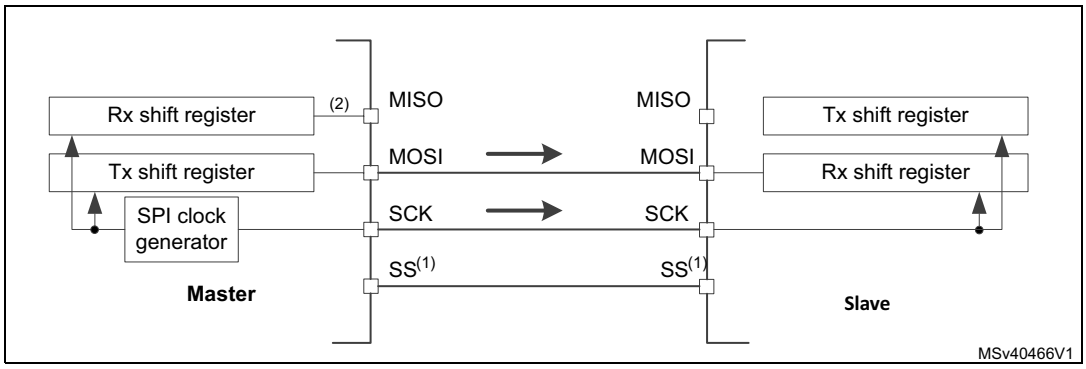
- **Receive-only mode: COMM[1:0]=10**

  In master mode, the MOSI output is disabled and may be used as GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by SW explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *49.4.6: Multi-master communication*). Received data events appear depending on the data buffer configuration.

Note: At whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register. (This bit may only be modified when the SPI is disabled). Any simplex communication can be replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled, while the HDDIR bit is never changed).

Figure 611. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)

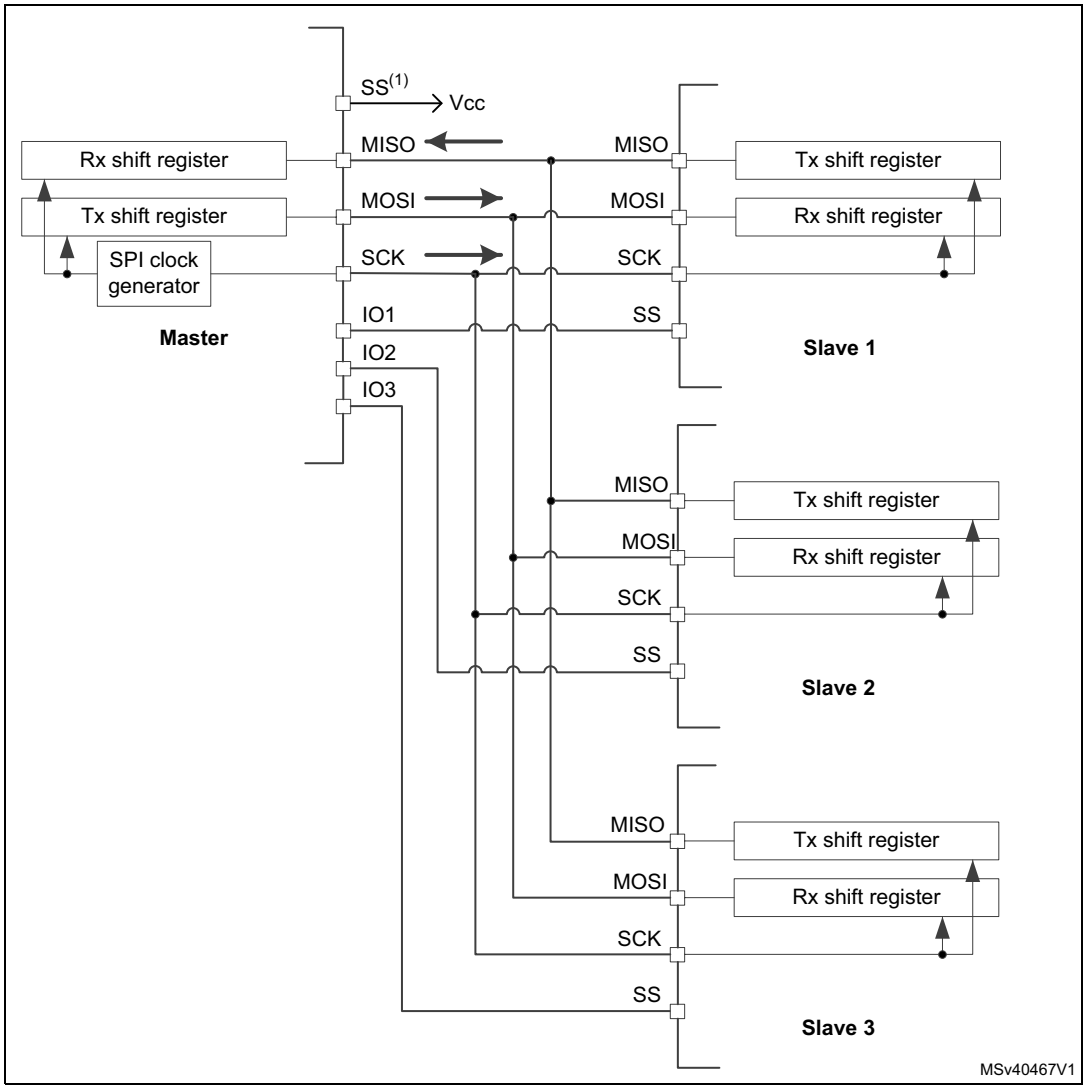


1. The SS pin is configured at output mode at master. The pins can be left unconnected then SS is managed by software internally on both master and slave side.
2. The input information is captured in the shift register and must be ignored in standard transmit only mode (for example, OVF flag)
3. In this configuration, both the MISO pins can be used as GPIOs.

49.4.5 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 612.*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave SS input (only one slave can control data on common MISO line at time). When this is done, a communication between the master and the selected slave is established. Except the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 612. Master and three independent slaves at star topology

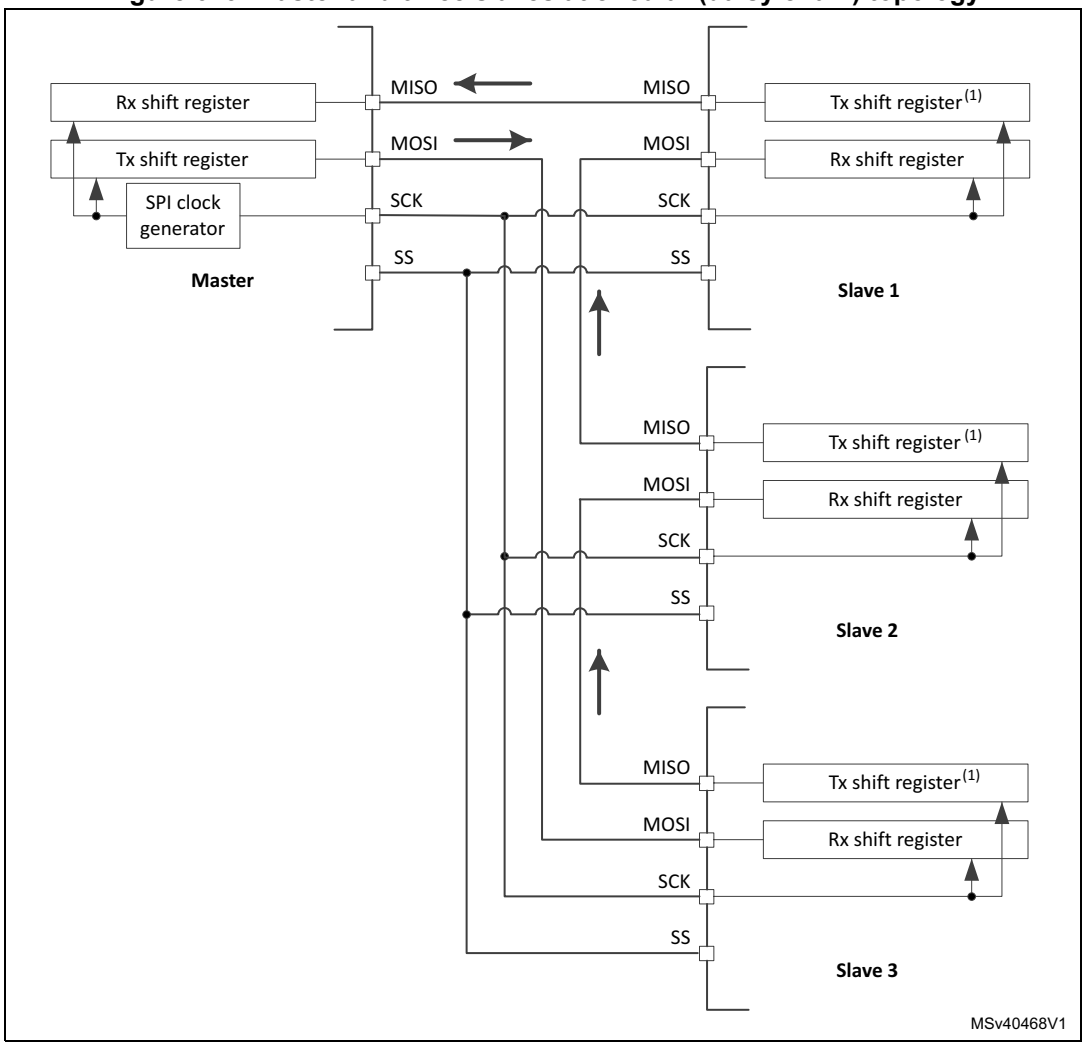


1. SS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.
2. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see *Section 10.3.7: I/O alternate function input/output on page 513*.)

The master can handle the SPI communication with all the slaves in time when a circular topology is applied (see *Figure 613*). All the slaves behave like simple shift registers applied at serial chain under common slave select and clock control. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transacted by the master is equal to the number of slaves. Then when a first data frame is transacted in the chain, the master just sends information dedicated for the last slave node in the chain via the first slave node input while the first information received by the master comes from the last node output at this time. Correspondingly, the lastly transacted data finishing the session is dedicated for the first slave node while its firstly outgoing data just reaches the master input after its circling around the chain passing through all the other slaves during the session. The data format configuration and clock setting has to be the same for all the nodes in the chain at this topology. As the receive and transmit shift registers are separated internally, a trick with

intentional underrun has to be applied at the TxFIFO slaves when information is transacted between the receiver and the transmitter by hardware. In this case, the transmission underrun feature is configured at a mode repeating lastly received data frame (UDRCFG[1:0]=01). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case the underrun happens in fact after this first data frame is transacted (underrun detection has to be set at end of data transaction at slaves UDRDET[1:0]=01). To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user has to disable and enable the SPI between sessions and fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 613. Master and three slaves at circular (daisy chain) topology



1. Underrun feature is used at slaves at this configuration when slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.

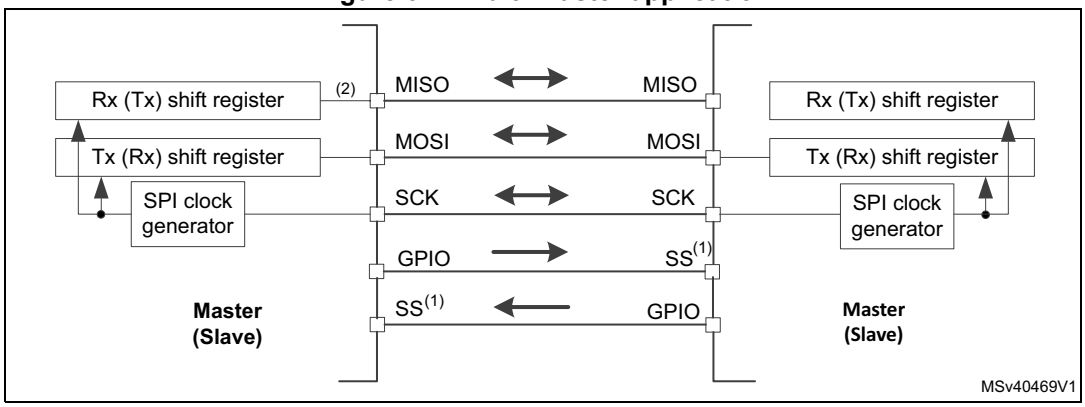
49.4.6 Multi-master communication

Unless the SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the SS pin is used configured at hardware input mode. The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 614. Multi-master application



1. The SS pin is configured at hardware input mode at both nodes. Its active level enable the MISO line output control as passive node is configured as a slave.

49.4.7 Slave select (SS) pin management

In slave mode, the SS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the SS can be used either as an output or an input. As an input it can prevent a multi master bus collision, and as an output it can drive a slave select signal of a single slave. The SS signal can be managed internally (software management of the SS input) or externally when both the SS input and output are associated with the SS pin (hardware SS management). The user can configure which level of this input/output external signal (present on the SS pin) is considered as active one by the SSIOP bit setting. While low level is considered as active internally SSIOP=1 setting can invert this logic for external world.

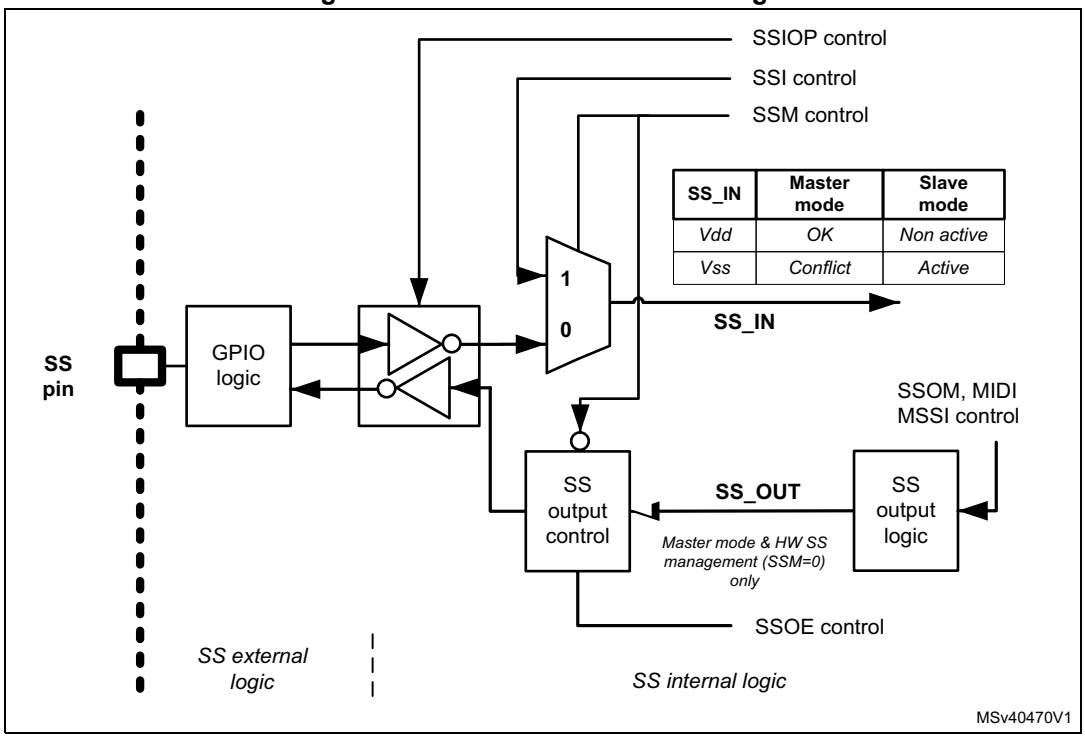
The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software SS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the register SPI_CR1. The external SS pin is free for other application uses (as GPIO or other alternate function).
- **Hardware SS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the SS output configuration (SSOE bit in register SPI_CFG2).
  - **SS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The SS pin is managed by the hardware.
    a) When SSOM = 0 and SP = 000, the SS signal is driven to the active level as soon as the master transfer starts (CSTART=1) and it is kept active until its EOT flag is set or the transmission is suspended.
    b) When SP = 001, a pulse is generated as defined by the TI mode.
    c) When SSOM=1, SP=000 and MIDI>1 the SS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (1 to 14).
  - **SS output disable (SSM=0, SSOE = 0)**:
    a) if the microcontroller is acting as the master on the bus, this configuration allows multi master capability. If the SS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER=0).
    b) In slave mode, the SS pin works as a standard ‘chip select’ input and the slave is selected while the SS line is at its active level.

Note: The purpose of automatic switching into slave mode at mode fault condition is to avoid the possible conflicts on data and clock line. The SPE is not automatically reset, as this would automatically flush both RX and TxFIFOs and current data may be lost. Following the MODF event, the SW must correctly manage the FIFO read/flush and correctly re-program the SPI configuration for taking over the slave role in the system.

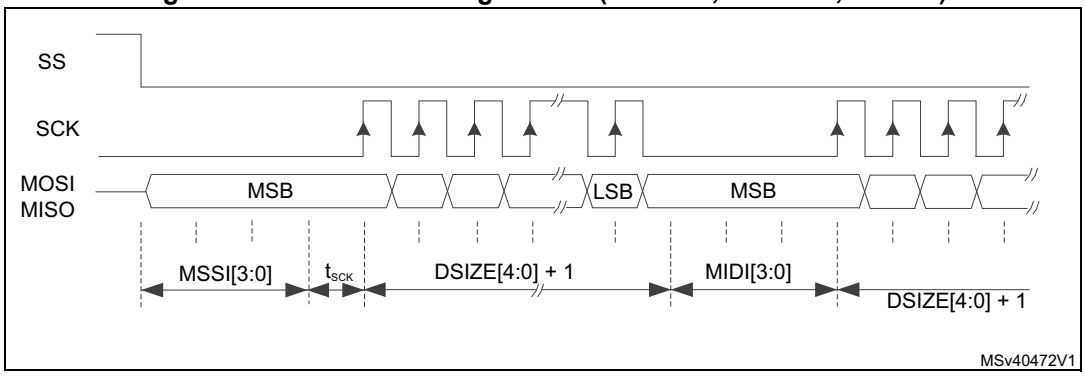
Note: When the SPI slave is enabled at the hardware SS management mode, all the traffics are ignored even in case of the SS is found at active level till the slave detects a start of the SS signal (its transaction from non-active to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external SS pin is fixed. There is no such protection at the SS software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

Figure 615. Scheme of SS control logic



When a hardware output SS control is applied (SSM=0, SSOE=1), by configuration of MIDI[3:0] and MSSI[3:0] bit fields the user can control timing of the SS signal between data frames and insert an extra delay at begin of every transaction (to separate the SS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 616: Data flow timing control (SSOE=1, SSOM=0, SSM=0)*

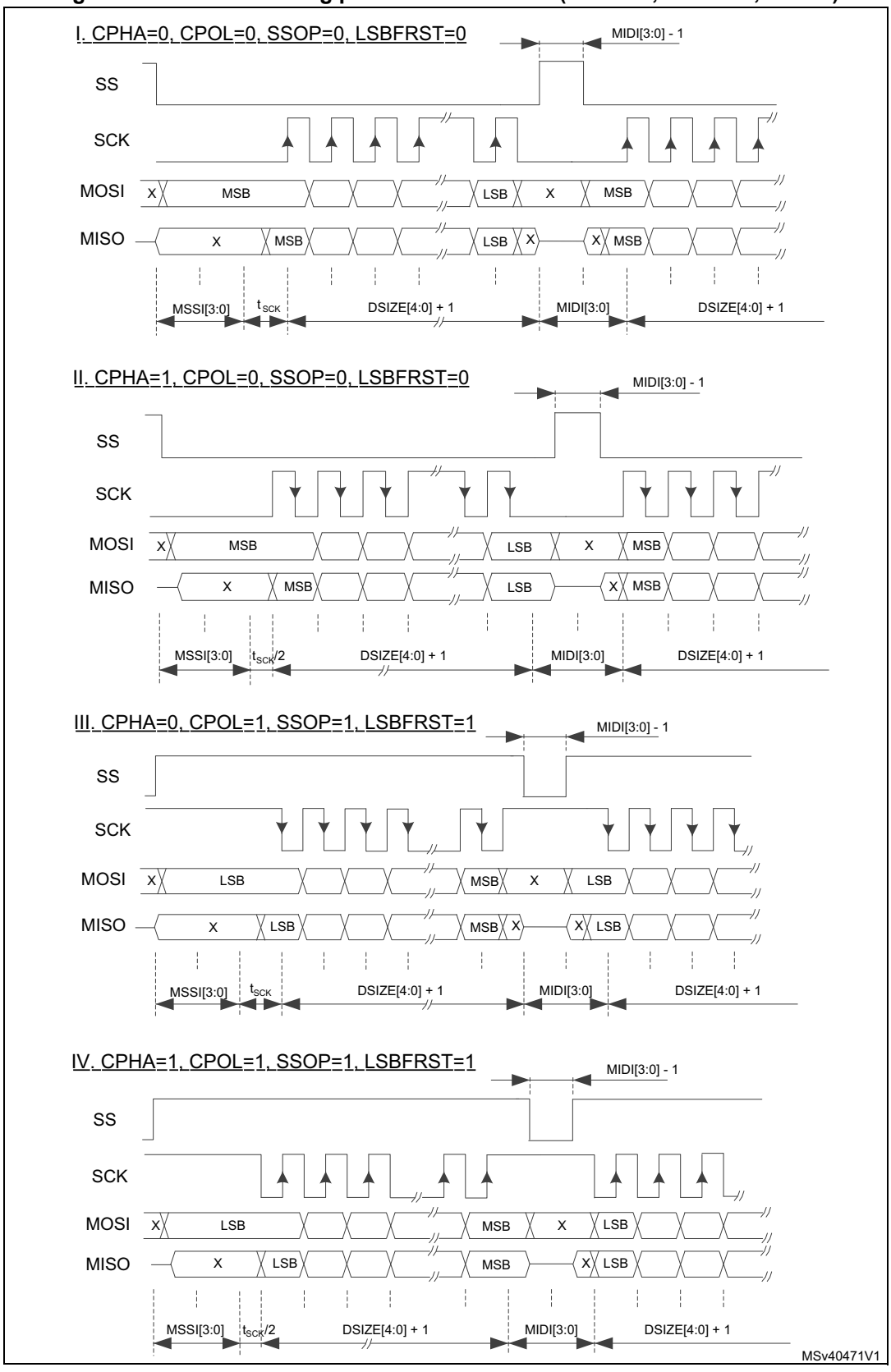
Figure 616. Data flow timing control (SSOE=1, SSOM=0, SSM=0)



1. MSSI[3:0]=0011, MIDI[3:0]=0011 (SCK flow is continuous when MIDI[3;0]=0).
2. CPHA=0, CPOL=0, SSOP=0, LSBFRST=0.

Additionally, bit SSOM=1 setting invokes specific mode which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] has to be set greater then one SPI period). Some configuration examples are shown at *Figure 617: SS interleaving pulses between data (SSOE=1, SSOM=1,SSM=0)*.

Figure 617. SS interleaving pulses between data (SSOE=1, SSOM=1,SSM=0)



1. MSSI[3:0]=0010, MIDI[3:0]=0010.
2. SS interleaves between data when MIDI[3:0]>1.

49.4.8 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data are being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

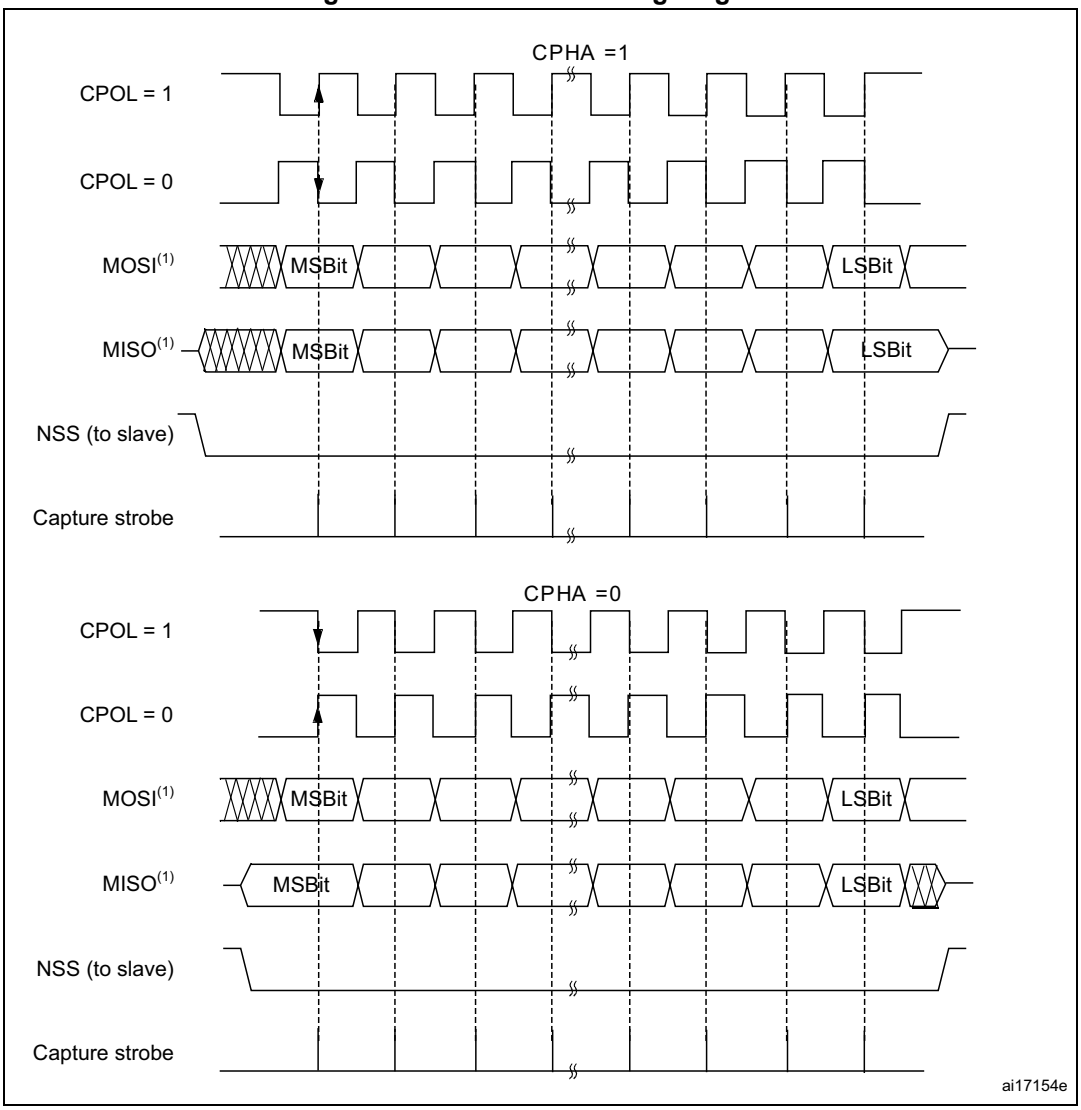
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 618, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

Figure 618. Data clock timing diagram



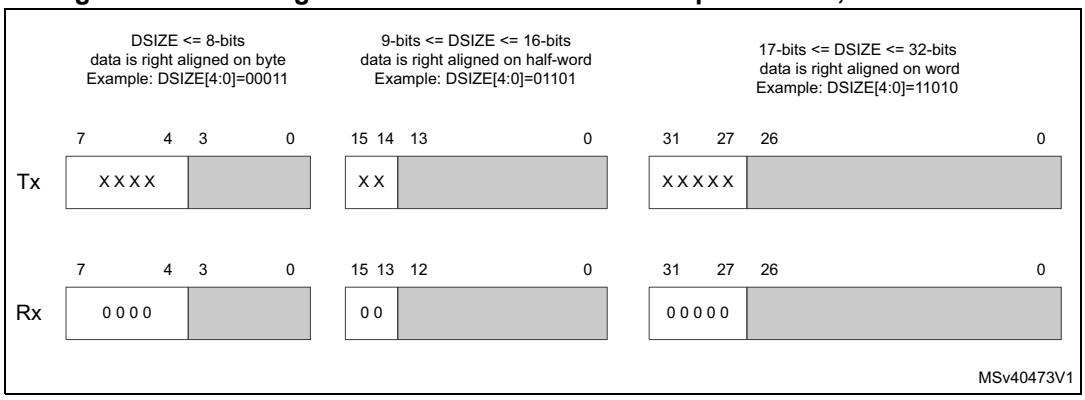
1. The order of data bits depends on LSBFRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit in SPI_CFG2 register. The data frame size is chosen by using the DSIZE[4:0] bits. It can be set from 4-bit up to 32-bit length and the setting applies for both transmission and reception. When the SPI_TXDR/SPI_RXDR registers are accessed, data frames are always right-aligned into either a byte (if the data fit into a byte), a half-word or a word (see *Figure 619*).

If the access is a multiple of the minimum data size needed for a single data frame, 2 or 4 data frames are packed into the register. During communication, only bits within the data frame are clocked and transferred.

Figure 619. Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit



Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 4-bit data frame size.

49.4.9 Configuration of SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication has to be initialized, perform these steps:

7. Write the proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
8. Write to the SPI_CFG1 and SPI_CFG2 registers to set up proper values of all not reserved bits and bit fields included there with next exceptions:
   a) SSOM, SSOE, MBR[2:0], MIDI[3:0] and MSSI[3:0] are required at master mode only, the MSSI bits take effect when SSOE is set, MBR setting is required for slave at TI mode, too
   b) UDRDET[1:0] and UDRCFG[1:0] are required at slave mode only,
   c) CRCSIZE[4:0] is required if CRCEN is set,
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP and SSM are not required at TI mode.
   e) Once the AFCNTR bit is set at SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless the peripheral enable so any later configurations changes of the SPI_CFG1 and SPI_CFG2 registers can affect level of signals at these pins.
   f) The I2SMOD bit at SPI_I2SCFGR register has to be kept cleared to prevent any unexpected influence of occasional I2S configuration.
9. Write to the SPI_CR2 register to select length of the transfer, if it is not known TSIZE has to be programmed to zero.
10. Write to SPI_CRCPOLY and into TCRCINI, RCRCINI and CRC33_17 bits at SPI2S_CR1 register to configure the CRC polynomial and CRC calculation if needed.
11. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see chapter Communication using DMA).
12. Program the IOLOCK bit in the SPI_CFG1 register if the configuration protection is required (for safety).

49.4.10 Procedure for enabling SPI

It is recommended to configure and enable the SPI slave before the master sends the clock but there is no impact if the configuration and enabling procedure is done while a traffic is on going on the bus. The data register of the slave transmitter must contain data to be sent before the master starts its clocking.The SCK signal must be settled to idle state level corresponding to the selected polarity before the SPI slave is selected by SS else following transaction may be desynchronized.

When the SPI slave is enabled at the hardware SS management mode all the traffics are ignored even in case of the SS is found at active level till the slave detects a start of the SS signal (its transaction from non-active to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when external SS pin is fixed. There is no such protection at the SS software management. In this case the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master at full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 49.4.14: Communication using DMA (direct memory addressing)*.

49.4.11 SPI data transmission and reception procedures

RxFIFO and TxFIFO

All SPI data transactions pass through the embedded FIFOs organized by bytes (N x 8-bit). The size of the FIFOs (N) is product and the peripheral instance dependent. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short or the interrupt/DMA latency is too long. Each direction has its own FIFO called TxFIFO and RxFIFO, respectively.

The handling of FIFOs depends on the data exchange mode (duplex, simplex), the data frame format (number of bits in the frame), the access size performed on the FIFO data registers (8-bit, 16-bit or 32-bit), and how data are organized at packets.

A read access to the SPI2S_RXDR register returns the oldest value stored in the RxFIFO that has not been read yet. A write access to the SPI2S_TXDR stores the written data in the TxFIFO at the end of a send queue.

A read access to the SPI2S_RXDR register must be managed by the RXP event. This flag is set by hardware when at least one complete data packet (defined as receiver threshold by FTHLV[3:0] bits at the SPI_CFG1 register) is available at the reception FIFO while reception is active. The RXP is cleared as soon as less data are available in the RxFIFO, when reading SPI2S_RXDR by software or by DMA.

The RXP triggers an interrupt if the RXPIE bit is set or a/o a DMA request if RXDMAEN is set.

Upon setting of the RXP flag, the application software performs the due number of SPI data register reads to download the content of one data packet. Once a complete data packet is downloaded, the application software checks the RXP value to see if other packets are

pending into the receive FIFO and, if so, downloads them packet by packet until the RXP reads 0. RxFIFO can store up to N data frames (for frame size =< 8-bit), N/2 data frames (for 8-bit < frame =< 16-bit), N/3 data frames (for 16-bit < frame =< 24-bit) or N/4 data frames (if data frame >24-bit) where N is the size of the FIFO in bytes.

At the end of a reception, it may happen that some data may still be available in the RxFIFO, without reaching the FTHLV level, thus the RXP is not set. In this case, the number of remaining RX data frames in the FIFO is indicated by RXWNE and RXPLVL fields in the SPI_SR register. It happens when number of the last data received in a transfer cannot fully accomplish the configured packet size in the case transfer size and packet size are not aligned. Nevertheless the application software can still perform the standard number of reads from the RxFIFO used for the previous complete data packets without drawbacks: only the consistent data (completed data frames) are popped from the RxFIFO while redundant reads (or any uncompleted data) are reading 0. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the reception of the last data in a transfer and from calculating the due number of reads to be popped from RxFIFO.

In a similar way, write access of a data frame to be transmitted is managed by the TXP event. This flag is set by hardware when there is enough space for the application software to push at least one complete data packet (defined at FTHLV[3:0] bits at SPI_CFG1 register) into the transmission FIFO while transmission is active. The TXP is cleared as soon as the TxFIFO is filled by software a/o by DMA and space currently available for any next complete data packet is lost. This can lead to oscillations of the TXP signal when data are released out from the TxFIFO while a new packet is stored frame by frame. Any write to the TxFIFO is ignored when there is no sufficient room to store at least a single data frame (TXP event is not respected), when TXTF is set or when the SPI is disabled.

The TXP triggers an interrupt if the TXPIE bit is set or a/o a DMA request if TXDMAEN is set. The TXPIE mask is cleared by hardware when the TXTF flag is set.

Upon setting of the TXP flag application software performs the due number of SPI data register writes to upload the content of one entire data packet. Once new complete data packet is uploaded, the application software checks the TXP value to see if other packets can be pushed into the TxFIFO and, if so, uploads them packet by packet until TXP reads 0 at the end of any packet load.

The number of last data in a transfer can be shorter than the configured packet size in the case when the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of data register writes used for the previous packets without drawbacks: only the consistent data are pushed into the TxFIFO while redundant writes are discarded. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the transmission of the last data in a transfer and from calculating the due number of writes to push the last data into TxFIFO. Just for the last data case, the TXP event is asserted by SPI once there is enough space into TxFIFO to store remaining data to complete current transfer.

Both TXP and RXP events can be polled or handled by interrupts. The DXP bit can be monitored as a common TXP and RXP event at full duplex mode.

Upon setting of the DXP flag the application software performs the due number of writes to the SPI data register to upload the content of one entire data packet for transmission, followed by the same number of reads from the SPI data register to download the content of one data packet. Once one data packet is uploaded and one is downloaded, the application software checks the DXP value to see if other packets can be pushed and popped in sequence and, if so, uploads/downloads them packet by packet until DXP reads 0.

The DXP triggers an interrupt if the DXPIE bit is set or a/o a DMA requests if TXDMAEN and RXDMAEN are set. The DXPIE mask is cleared by hardware when the TXTF flag is set.

The DXP is useful in Full-Duplex communication in order to optimize performance in data uploading/downloading, and reducing the number of interrupts required to support an SPI transfer thus minimizing the request for CPU bandwidth and system power especially when SPI is operated in Stop mode.

Another way to manage the data exchange is to use DMA (see *Communication using DMA (direct memory addressing)*).

If the next data is received when the RxFIFO is full, an overrun event occurs (see description of OVR flag at *Section 49.5.2: SPI error flags*). An overrun event can be polled or handled by an interrupt.

This may happen in slave mode or master mode (full duplex or receive only with MASRX = 0). In master receive only mode, with MASRX = 1, the generated clock stops automatically when the RxFIFO is full, therefore overrun is prevented.

Both RxFIFO and TxFIFO content is kept flushed when SPI is disabled (SPE=0).

Sequence handling

A few data frames can be passed at single sequence to complete a message. The user can handle number of data within a message thanks to values stored into TSIZE and TSER fields. In principle, the transaction of a message starts when the SPI is enabled by setting CSTART bit and finishes when number of required data is transacted. The end of transaction controls the CRC and the hardware SS management when applied. If TSIZE is kept at zero while CSTART is set, an endless transaction is initialized (no data size control is applied). The transaction can be suspended at any time thanks to CSUSP which clears the CSTART bit.

In master mode, the user can extend the number of data within the current session. When the number of data programmed into TSIZE is transacted and if TSER contains a non-zero value, the content of TSER is copied into TSIZE, and TSER value is cleared automatically. The transaction is then extended by a number of data corresponding to the value reloaded into TSIZE. The EOT event is not raised in this case as the transaction continues. After the reload operation, the TSERF flag is set and an interrupt is raised if TSERFIE is set. The user can write the next non-zero value into TSER before the next reload occurs, so an unlimited number of data can be transacted while repeating this process.

When any data extension is applied, it always starts by aligned data packet. That is why it is suggested to keep number of data to be extended always aligned with packet size else the last data packet just before the extension is applied has to be handled as an incomplete one (see data packing chapter). If overall number of data is not aligned, the user must implement the rest not aligned number of data into TSER just at the last extension cycle and then handle the last incomplete packet of data standardly within EOT event handler.

For example, if the user wants to transfer 23 bytes while applies data number extension at configuration of 8-bit data size, data packet set to 4 data and 32-bit access to FIFO is used then whatever next sequence is correct

- TSIZE=16 TSER=7;
- TSIZE=12 TSER=8; last extensionTSER=3;

As the last not aligned MSB byte is ignored just within the last (6th) access of the FIFO.

When a not aligned sequence is applied for data to be extended like at the following cases

- TSIZE=15 TSER=8 or
- TSIZE=8 TSER=7; last extension TSER=8;

The MSB byte is ignored within the 4th access of the FIFO while the other accesses handle always 4 data at the FIFO.

When the transmission is enabled, a sequence begins and continues while any data is present in the TxFIFO of the master. The clock signal is provided permanently by the master until TxFIFO becomes empty, then it stops, waiting for additional data.

In receive-only modes, half duplex (COMM[1:0]=11, HDDIR=0) or simplex (COMM[1:0]=10) the master starts the sequence when SPI is enabled and transaction is released by setting the CSTART bit. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled/suspended by the master. The master receives data frames permanently up to this moment. The reception can be suspended either by SW control, writing 1 to the CSUSP bit in the SPI_CR1 register, or automatically when MASRX=1 and RxFIFO becomes full. The reception is automatically stopped also when the number of frames programmed in TSIZE and TSER fields of the SPI_CR2 register has been completed.

In order to disable the master receive only mode, the SPI must be suspended at first. When the SPI is suspended, the current frame is completed, before changing the configuration.

Caution: If SPE is written to 0 at master, while reception is ongoing without any suspending, the clock is stopped without completing the current frame, and the RxFIFO is flushed.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays by MIDI[3:0] bits setting or provide an initial delay by setting MSSI[1:0] which postpones any transaction start to give slave sufficient room for preparing data. Be aware data from the slave are always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are short, FIFO is accessed by bytes and the SPI bus rate is high.

In order to add some SW control on the SPI communication flow from a slave transmitter node, a specific value written in the SPI_UDRDR (SPI Underrun Data Register) may be used. On slave side, when TxFIFO becomes empty, this value is sent out automatically as next data and may be interpreted by SW on the master receiver side (either simply dropped or interpreted as a XOFF like command, in order to suspend the master receiver by SW).

At multi-slave star topology, a single slave can be only enabled for the output data at a time. The slave just selected for the communication with the master needs to detect a change of its SS input into active level before the communication with the master starts. In a single slave system it is not necessary to control the slave with SS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. The SS can be managed by both software and hardware (*Section 49.4.7: Slave select (SS) pin management*).

49.4.12 Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph.

At the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Otherwise, ongoing transactions may be corrupted in this case.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into STOP mode. Refer to the RCC section for further information.

The master in full duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. TXC flag can be polled (or interrupt enabled with EOTIE=1) in order to wait for the last data frame to be sent.

When the master is in any receive only mode, in order to stop the peripheral, the SPI communication must be first suspended, by setting CSUSP to 1.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user has to ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When the SS signal is managed by software and the master has to provide proper end of SS pulse for slave, or
- When transaction streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure in master mode, except when receive only mode is used, is:

1. Wait until TXC=1 and/or EOT=1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed. TXC/EOT is set when CRC frame is completed in this case. When a transmission is suspended the software has to wait till CSTART bit is cleared.
2. Read all RxFIFO data (until RXWNE=0 and RXPLVL=00)
3. Disable the SPI (SPE=0).

The correct disable procedure for master receive only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP=1)
2. Wait until SUSP=1 (the last data frame is processed) if receive flow is suspended.
3. Read all RxFIFO data (until RXWNE=0 and RXPLVL=00)
4. Disable the SPI (SPE=0).

In slave mode, any on going data are lost when disabling the SPI.

49.4.13 Data packing

From user point of view there are two ways of data packing which can overlay each other:

- Type of access when data are written to TxFIFO or read from RxFIFO
  Multiple data can be pushed or fetched effectively by single access if data size is considerably less than access performed upon SPI2S_TXDR or SPI2S_RXDR registers.
- Number of data to be handled during the single software service
  It is convenient to group data into packets and cumulate the FIFO services overall the data packet content exclusively instead of handling data frame by frame separately. The user can define packets by FIFO threshold settings. Then all the FIFO occupancy events are related to that threshold level while required services are signalized by proper flags with interrupt and/or wake up capabilities.

When the data frame size fits into one byte (less than or equal to 8 bits), the data packing is used automatically when any read or write 16-bit or 32-bit access is performed on the SPI2S_RXDR/SPI2S_TXDR register. The multiple data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other data stored in the MSB. *Figure 620* provides an example of data packing mode sequence handling. While DSIZE[3:0] is configured to 4-bit there, two or four data frames are written in the TxFIFO after the single 16-bit or 32-bit access the SPI2S_TXDR register of the transmitter.

When the data frame size is between 9-bit and 16-bit, data packing is used automatically when a 32-bit access is done. the least significant half-word is used first. (regardless of the LSBFRST value)

This sequence can generate two or four RXP events in the receiver if the RxFIFO threshold is set to 1 frame (and data is read on a frame basis, unpacked), or it can generate a single RXP event if the FTHLV[3:0] field in the SPI_CFG1 register is programmed to a multiple of the frames to be read in a packed mode (16-bit or 32-bit read access).

The data are aligned in accordance with *Figure 619: Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit*. The valid bits are performed on the bus exclusively. Unused bits are not cared at transmitter while padded by zeros at receiver.

When short data frames (<8-bit or < 16-bit) are used together with a larger data access mode (16-bit or 32-bit), the FTHLV value must be programmed as a multiple of the number of frames/data access (i.e. multiple of 4 if 32-bit access is used to up to 8-bit frames or multiple of 2 if 16-bit access is used to up to 8-bit frames or 32-bit access to up to 16-bit frames.).

The RxFIFO threshold setting must always be higher than the following read access size, as spurious extra data would be read otherwise.

The FIFO data access less than the configured data size is forbidden. One complete data frame has to be always accessed at minimum.

A specific problem appears if an incomplete data packet is available at FIFO: less than 4x8-bit frames or one single 16-bit frame is available.

There are two ways of dealing with this problem:

A. without using TSIZE field

On transmitter side, writing the last data frame of any odd sequence with an 8-bit/16-bit access to SPI2S_TXDR is enough.

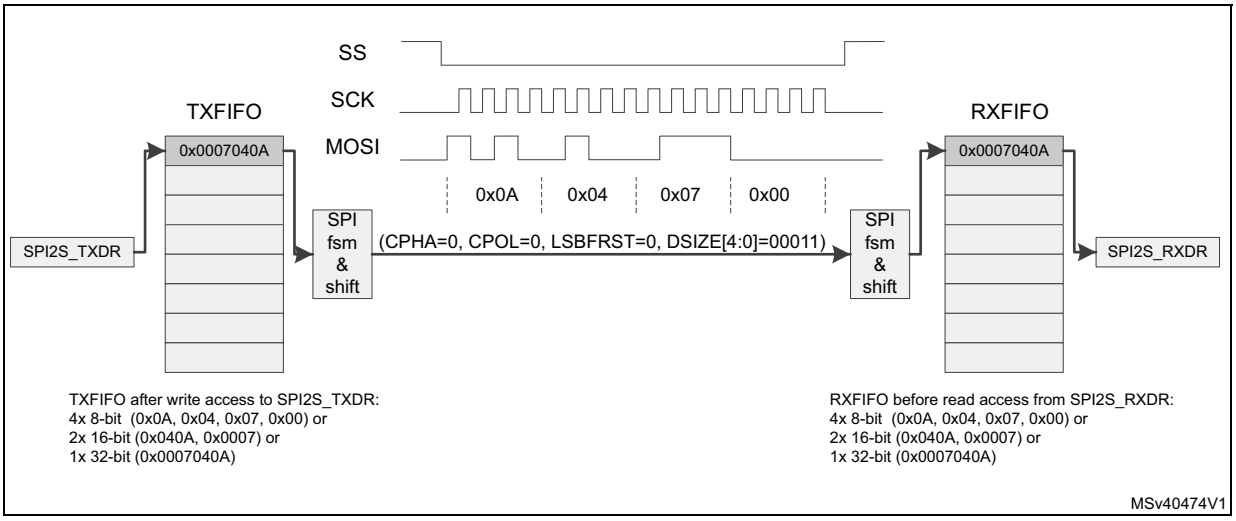
On receiver side, the remaining data may be read by any access. Any extra data read are padded with zeros. Polling the RXWNE and RXPLVL may be used to detect when the RX data are available in the RxFIFO. (a time out may be used at system level in order to detect the polling)

B. using the TSIZE field

On transmitter side, the transaction is stopped by the master when it faces EOT event.

In reception, the RXP flag is not set when EOT is set. In the case when the number of data to be received (TSIZE) is not a multiple of packet size, the number of remaining data is indicated by the RXWNE and RXPLVL fields in the SPI_SR register. The remaining data can be read by any access. Any extra read is padded by zeros.

Figure 620. Packing data in FIFO for transmission and reception



1. DSIZE[3:0] is configured to 4-bit, data is right aligned, valid bits are performed only on the bus, their order depends on LSBFRST, content of LSB byte goes first on the bus.

49.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits in the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a series of DMA requests is triggered each time TXP is set to 1. The DMA then performs series of writes to the SPI2S_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set to 1. The DMA then performs series of reads from the SPI2S_RXDR register. When EOT is set at the end of transaction and last data packet is incomplete then DMA request is activated automatically in according with RXWNE and RXPLVL[1:0] setting to read rest of data.

When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

If the SPI is programmed in receive only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel.

If the SPI is programmed in transmit only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set at SPI2C_SR register), the TXC flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT=1 and/or TXC=1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA request for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

If the DMA channel PSIZE value is equal to 16-bit or 32-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. Similarly, If the DMA channel PSIZE value is equal to 32-bit and SPI data size is less than or equal to 16-bit, then packing mode is enabled.The DMA then automatically manages the write operations to the SPI2S_TXDR register.

Regardless data packing mode is used and the number of data to transfer is not a multiple of the DMA data size (16-bit or 32-bit) while the frame size is smaller, DMA completes the transfer automatically in according with the TSIZE field setting.

Alternatively, last data frames may be written by software, in the single/unpacked mode.

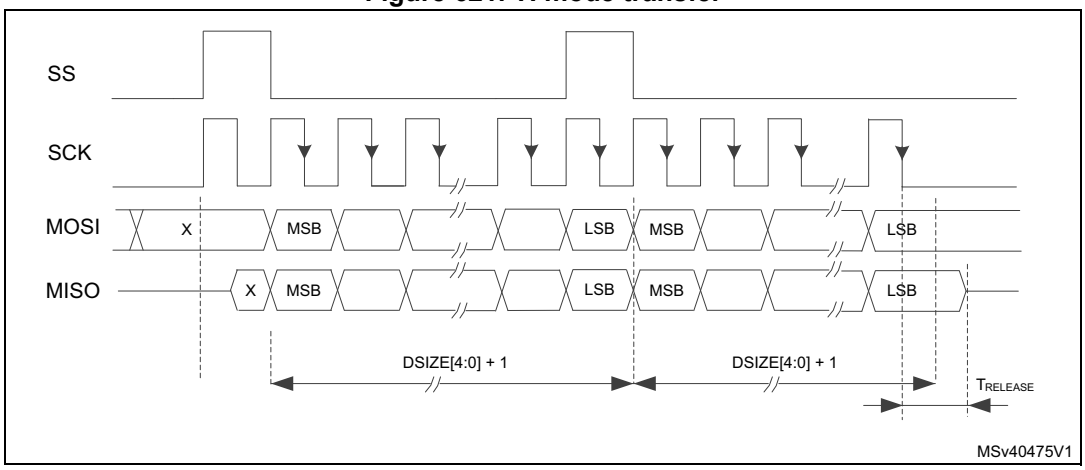
To configure any DMA data access less than the configured data size is forbidden. One complete data frame has to be always accessed at minimum.

49.5 SPI specific modes and control

49.5.1 TI mode

By specific setting of the SP[2:0] bit field at the SPI_CFG2 register the SPI can be configured to be compliant with TI protocol. The SCK and SS signals polarity, phase and flow as well as the bits order are fixed so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP and SSM is not required when the SPI is at TI mode configuration. The SS signal synchronizes the protocol by pulses over the LSB data bit as it is shown at the *Figure 621: TI mode transfer*.

Figure 621. TI mode transfer



In slave mode, the clock generator is used to define time when the slave output at MISO pin becomes to HiZ when the current transaction finishes. The master baud setting is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become HiZ (TRELEASE) depends on internal re-synchronization, too. It is given by formula:

$$\frac{Tbaud}{2} + 2 \times Tcom < Trelease < \frac{Tbaud}{2} + 4 \times Tcom$$

If the slave detects misplaced SS pulse during data transaction the TIFRE flag is set.

49.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the corresponding Interrupt Enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. OVR flag triggers an interrupt if OVRIE bit is set. Clearing the OVR bit is done by a writing 1 to the OVRC bit in the SPI_IFCR. To prevent any

next overrun event the clearing must be done after RxFIFO is emptied by software reads. At master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

At a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO at the moment specified by UDRDET bits. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Once the underrun is captured next provided data for transmission depends on the UDRCFG bits. The slave can provide out either data stored lastly to its TxFIFO or the data received previously from the master or a constant pattern stored by the user at the UDRDR register. The second configuration can be used at circular topography structure (see *Figure 613*). Standard transmission is re-enabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. The user must write some data into TxFIFO prior clearing UDR flag to prevent any next underrun condition occurrence capture.

The data transacted by slave is unpredictable especially when the transaction starts or continues while TxFIFO is empty and underrun condition is either not yet captured or just cleared. Typically, this is the case when UDRDET[1:0]=00 or SPI is just enabled or when a transaction with a defined size just starts. First bits can be corrupted in this case, as well, when slave software writes first data into the empty TxFIFO too close prior the data transaction starts (propagation of the data into TxFIFO takes few APB clock cycles). If the user cannot ensure to write data into the empty TxFIFO in time the UDRDET[1:0]=00 setting must be avoided.

To handle the underrun control feature correctly the user must avoid next critical encroachments especially

- Any fill of empty TxFIFO when master starts clocking (at UDRDET[1:0]=00 especially)
- Any clear of UDR flag while TxFIFO is empty
- Any setting of UDRDET[1:0]=00 together with UDRCFG[1:0]=10
- Any setting of UDRDET[1:0]=10 when underrun must be detected after each data frame while SS signal does not toggle between the frames
- Any setting of UDRDET[1:0]=10 while SS is managed by software

Note: The hardware propagation of an UDR flag change needs additional traffic on the bus. It always takes 3 SPI clock cycles after the event happen (underrun captured by hardware or the UDR flag cleared by software).

Mode fault (MODF)

Mode fault occurs when the master device has its internal SS signal (SS pin in SS hardware mode, or SSI bit in SS software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the MODFIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit in the SPI_IFCR.

To avoid any multiple slave conflicts in a system comprising several MCUs, the SS pin must be pulled to its non-active level before re-enabling the SPI, by setting the SPE bit.

As a security, hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multi master conflict.

A correct SW procedure when master overtakes the bus at multi master system must be the following one:

- Switch into master mode while SSOE=0
  (potential conflict can appear when another master occupies the bus. MODF is raised in this case which prevents any next node switching into master mode)
- Put GPIO pin dedicated for another master SS control into active level
- Perform data transaction
- Put GPIO pin dedicated for another master SS control into non active level
- Switch back to slave mode

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CFG1 register is set. The CRCE flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if CRCIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit in the SPI_IFCR.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an SS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the SS pulse is ignored, and the SPI waits for the next SS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of few data bytes.

The TIFRE flag is cleared by writing 1 to the TIFREC bit in the SPI_IFCR. If the TIFREIE bit is set, an interrupt is generated on the SS error detection. As data consistency is no longer guaranteed, communication must be re-initiated by software between master and slave.

49.5.3 CRC computation

Two separate 33-bit or two separate 17-bit CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers any CRC polynomial length from 5 to 33 bits when maximum data size is 32-bit and from 5 to 17 bits for the peripheral instances where maximum data size is limited to 16-bit. The length of the polynomial is defined by the most significant bit of the value stored at the CRCPOLY register. It has to be set greater than data frame length defined at DSIZE field. When maximum data size is applied, the CRC33_17 bit has to be set additionally to define the most significant bit of the polynomial string while keep its size always greater than data. The CRCSIZE field in the SPI_CFG1 then defines how many the most significant bits from CRC calculation registers are transacted and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit in the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial

defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has to be sent or received in the SPI_DR register.

The length of the transfer has to be defined by TSIZE and TSER. When the desired number of data is transacted, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

TSIZE cannot be set to 0xFFFF value if CRC is enabled. A correct way of sending e.g. 65535 data with CRC is to set:

- TSIZE= 0xFFFE and TSER=1 when data packet is configured to keep one data respective
- TSIZE= 0xFFFC and TSER=3 when data packet keeps 4 data (to ensure the TSIZE value aligned with packet size when its extension is applied).

In transmission, the CRC computation is frozen during CRC transaction and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when desired number of data is transacted. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is completed, an automatic check is performed comparing the received CRC value and the value calculated in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing 1 to the CRCERRC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode in order to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI at SPI2S_CR1 register.

The CRC values are reset when the SPI is disabled.

49.6 Low-power mode management

The SPI has advanced low-power mode functions allowing it to transfer properly data between the FIFOs and the serial interface even when the **spi_pclk** clock is disabled.

In master mode the **spi_ker_ck** kernel clock is needed in order to provide the timings of the serial interface.

In slave mode, the **spi_ker_ck** clock can be removed as well during the transfer of data between the FIFOs and the serial interface. In this mode the clock is provided by the external SPI device.

When the **spi_pclk** clock is gated, (and the **spi_ker_ck** clock as well if the SPI is in slave), the SPI provides a wakeup event signal (**spi_wkup**) if a specific action requiring the activation of the **spi_pclk** clock is needed, such as:

- To fill-up the TxFIFO,
- To empty the RxFIFO,
- Other signaling: end of transfer, errors...

The generation of **spi_ker_ck** and **spi_pclk** clock are controlled by the RCC block according to register settings and the processors modes. Refer to the RCC section for details.

The application shall acknowledge all pending interrupts events before switching the SPI to low-power mode (i.e. removing **spi_pclk**).

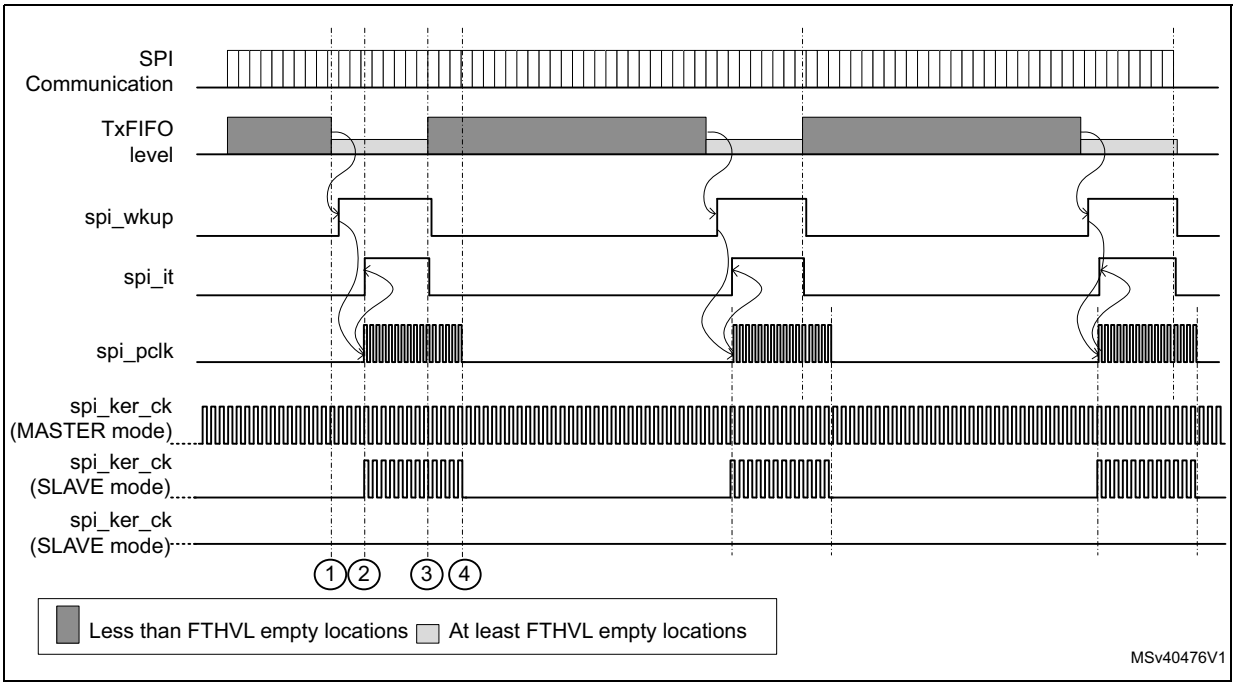
The *Figure 622* shows an example of the clock handling when the SPI2S is working in low-power mode. The example is given for a transmit mode.

In master mode the **spi_ker_ck** clock is required for the timing generation.

The *Figure 622* shows two kinds of supported scenarios for the handling of the **spi_ker_ck** kernel clock in slave mode:

- In most of the slave modes, the **spi_ker_ck** kernel clock can be disabled,
- In some products, the **spi_ker_ck** kernel clock activation may follow the system state.

Figure 622. Low-power mode application example



The figure clearly shows that the **spi_pclk** must be provided to the SPI2S, when data need to be transferred from the memory to the SPI2S TxFIFO. Here is the description of the most important steps:

- **Step 1**
  The TxFIFO level goes below the programmed threshold, this event (TXP) activates the **spi_wkup** signal. This signal is generally used to wake-up the system from low-power mode, and thus to activate the bus clock (**spi_pclk**).
- **Step 2**
  When **spi_pclk** is activated, the **spi_it** is also activated, and the product is ready to fill-up the TxFIFO either by DMA or by software. Note as well that for some product the system wake-up automatically enables the **spi_ker_ck** kernel clock as well.
- **Step 3**
  When the amount of empty locations in the TxFIFO is less than FTHLV, then the **spi_wkup** and **spi_it** signals are deactivated, but the fill-up of the TxFIFO may continue. Note that **spi_wkup** falling edge is aligned with the serial interface clock domain, and the falling edge of the **spi_it** is aligned with the **spi_pclk** clock domain.
- **Step 4**
  The fill-up of the TxFIFO is completed; the software can switch the system back to low-power mode until the next **spi_wkup** occurs.

49.7 SPI wakeup and interrupts

Table 400 gives an overview of the SPI events capable to generate interrupt events (**spi_it**). Some of them feature wake-up from low-power mode capability additionally (**spi_wkup**).

Most of them can be enabled and disabled independently while using specific interrupt enable control bits.

Table 400. SPI wakeup and interrupt requests

Interrupt event	Event flag(1)	Enable Control bit	Event clear method	Interrupt/Wakeup activated	
				spi_it	spi_wkup
TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	YES	YES
Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples		YES
Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared		YES
Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1		NO
Underrun	UDR	UDRIE	Writing UDRC to 1		YES
Overrun	OVR	OVRIE	Writing OVRC to 1		YES
CRC Error	CRCE	CRCEIE	Writing CRCEC to 1		YES
TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1		NO
Mode Fault	MODF	MODFIE	Writing MODFC to 1		NO
End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1		YES
Master mode suspended	SUSP		Writing SUSPC to 1		YES
TxFIFO transmission complete (TxFIFO empty)	TXC		TXC cleared by HW when a transmission activity starts on the bus		NO
TSER value transferred to TSIZE (new value may be loaded to TSER)	TSERF	TSERFIE	Writing TSERFC to 1		NO

1. Refer to SPI2S register description for more details about the event flags.

49.8 I2S main features

- Full duplex communication
- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length may be 16, 24 or 32 bits
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, Overrun and Frame Error
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component. The ratio is fixed at 256 x F_{WS} (where F_{WS} is the audio sampling frequency)

49.9 I2S functional description

49.9.1 I2S general description

The block diagram shown on *Figure 608* also applies for I2S mode.

The SPI/I2S block can work on I2S/PCM mode, when the bit I2SMOD is set to 1. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA and parts of interrupt signaling are shared with SPI function. The low-power mode function is also available in I2S mode, refer to *Section 49.6: Low-power mode management* and *Section 49.10: I2S wakeup and interrupts*.

49.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: Serial Data Output (mapped on the MOSI pin) to transmit the audio samples in master, and to receive the audio sample in slave. Refer to *Section : Serial Data Line swapping on page 2170*.
- SDI: Serial Data Input (mapped on the MISO pin) to receive the audio samples in master, and to transmit the audio sample in slave. Refer to *Section : Serial Data Line swapping on page 2170*.
- WS: Word Select (mapped on the SS pin) is the frame synchronization. It is configured as output in master mode, and as input for slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial bit clock. It is configured as output in master mode, and as input for slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I2S is configured in master mode. The master clock rate is fixed to 256 x F_{WS}, where F_{WS} is the audio sampling frequency.

49.9.3 Bits and fields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = ‘1’), some bit fields are no longer relevant, and must be forced to a specific value in order to guarantee the behavior of the I2S/PCM function. *Table 401* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 401. Bit fields usable in PCM/I2S mode

Register name	Bit fields usable in PCM/I2S Mode	Constraints on other bit fields
SPI/I2S control register 1 (SPI2S_CR1)	IOLOCK, CSUSP, CSTART	Other fields set to their reset values
SPI control register 2 (SPI_CR2)	-	Set to reset value
SPI configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI2S_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI2S_SR)	RXWNE, RXPLVL, SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI2S_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S transmit data register (SPI2S_TXDR)	The complete register	-
SPI/I2S receive data register (SPI2S_RXDR)	The complete register	-
SPI polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI transmitter CRC register (SPI_TXCRC)	-	
SPI receiver CRC register (SPI_RXCRC)	-	
SPI underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

49.9.4 Slave and master modes

The SPI/I2S block supports master and slave mode for both I2S and PCM protocols.
In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK is not used in slave mode.

In order to improve the robustness of the SPI/I2S block in slave mode, the peripheral re-synchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM standard, the shift-in or shift-out of each data is triggered one bit clock after the rising edge WS.

Note: This re-synchronization mechanism is not available for the I2S LSB justified standard.

Note: Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

49.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel shall be considered as the left, and which one is the right.

In I2S master mode, four frames formats are supported:

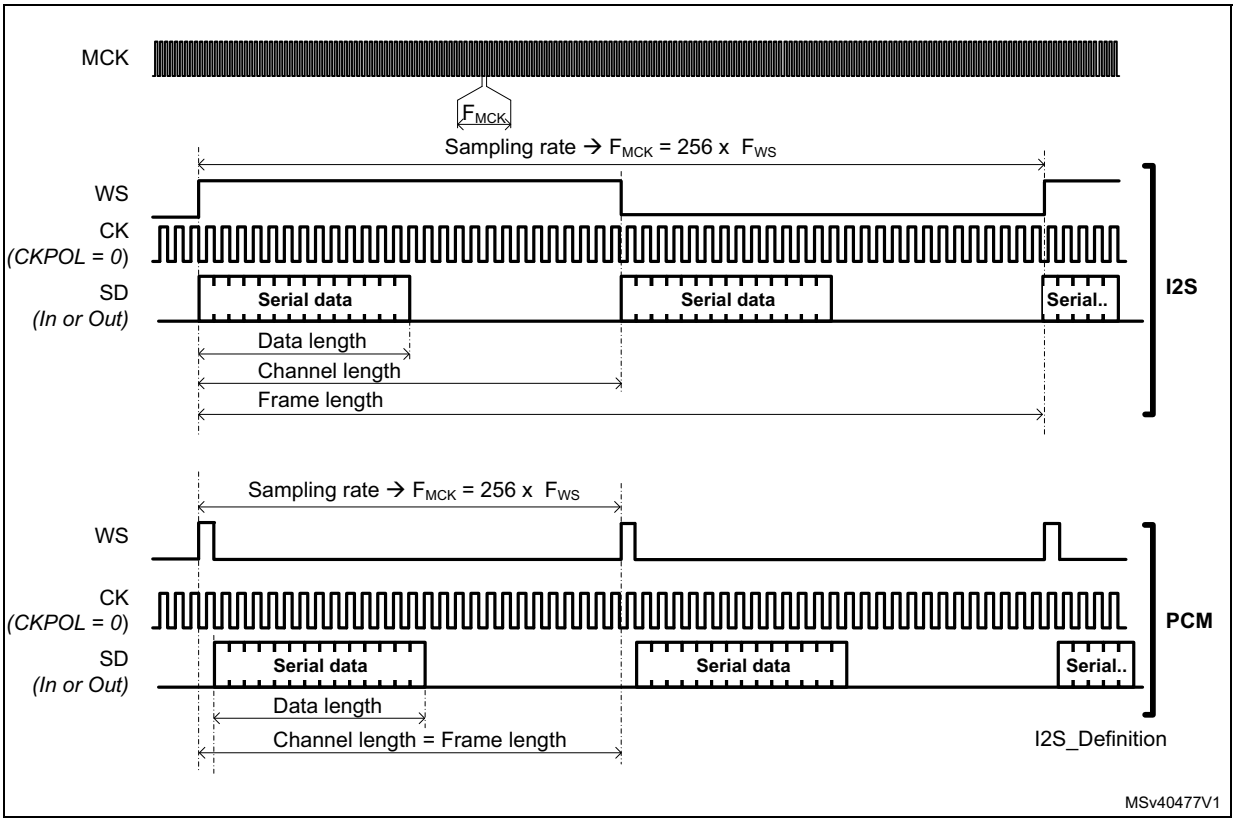
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel
- 32-bit data packed in a 32-bit channel

In PCM master mode, three frames formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 623. Waveform examples

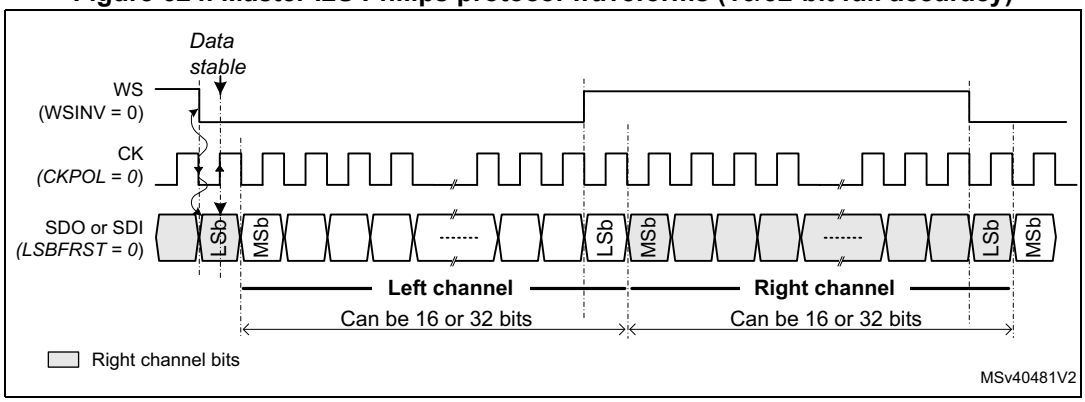


I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transfered is the left channel, and a rising edge transition indicates that the next data transfered is the right channel.

Figure 624. Master I2S Philips protocol waveforms (16/32-bit full accuracy)

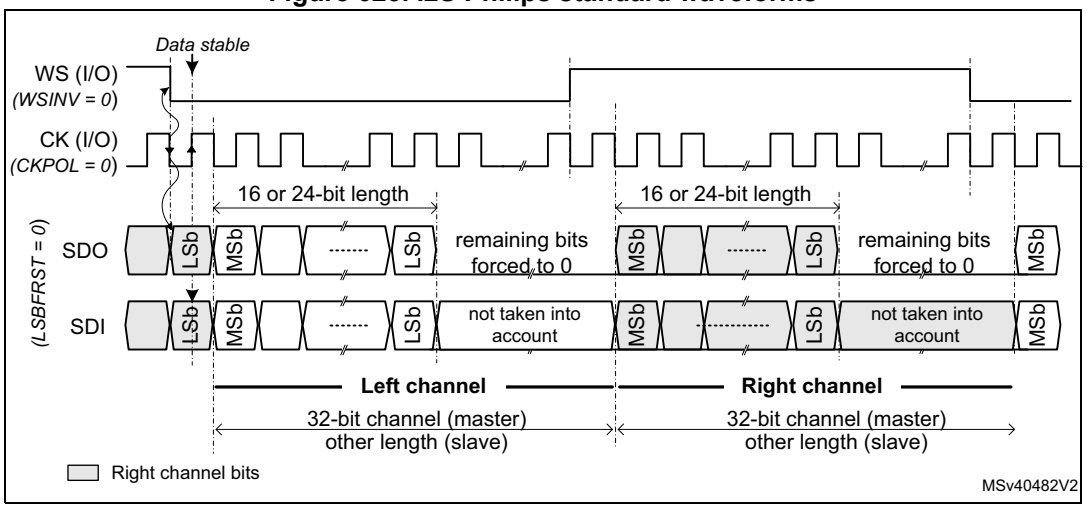


CKPOL is set to 0 in order to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

Figure 624 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS Inversion* for information concerning the handling of WS signal.

Figure 625. I2S Philips standard waveforms

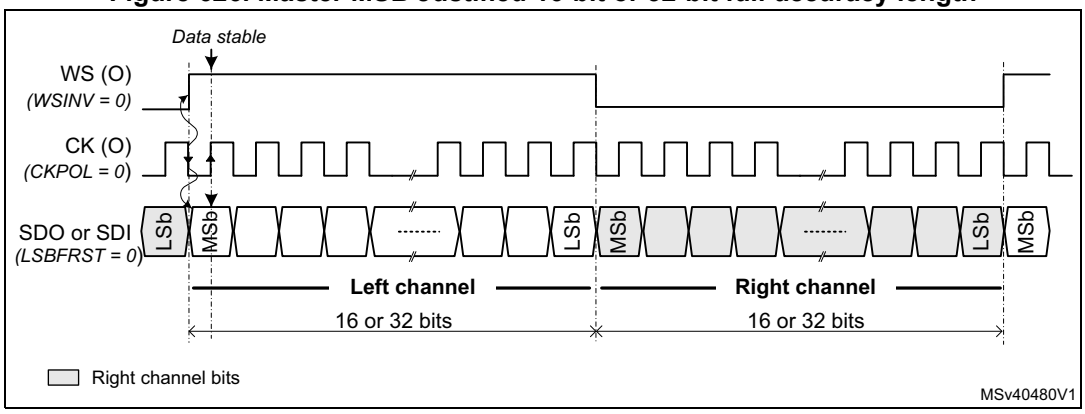


In the case where the channel length is bigger than the data length, the remaining bits are forced to zero when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB justified standard

For this standard, the WS signal toggles when the first data bit, is provided. The data transfered represents the left channel if WS is high, and the right channel if WS is low.

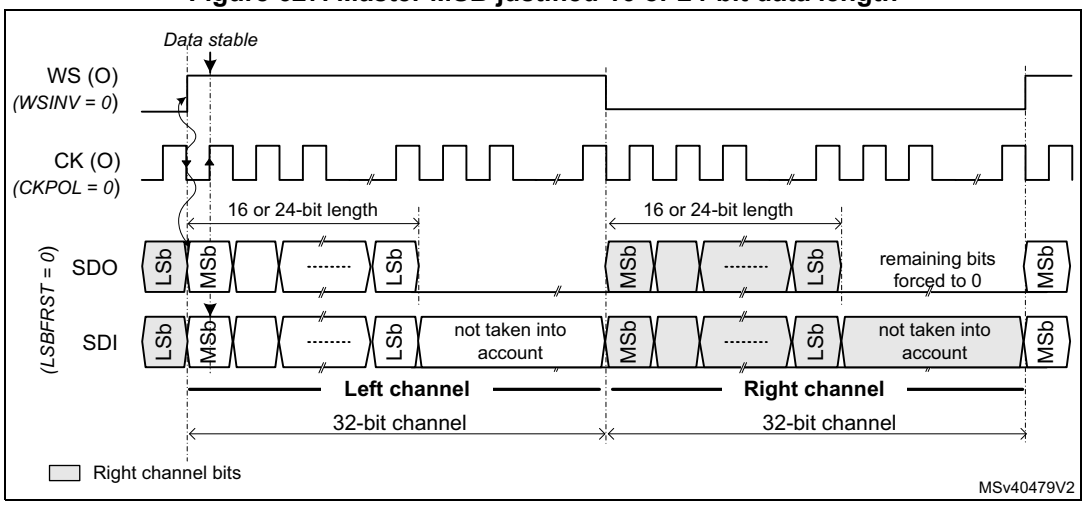
Figure 626. Master MSB Justified 16-bit or 32-bit full-accuracy length



CKPOL is set to 0 in order to match the I2S MSB justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

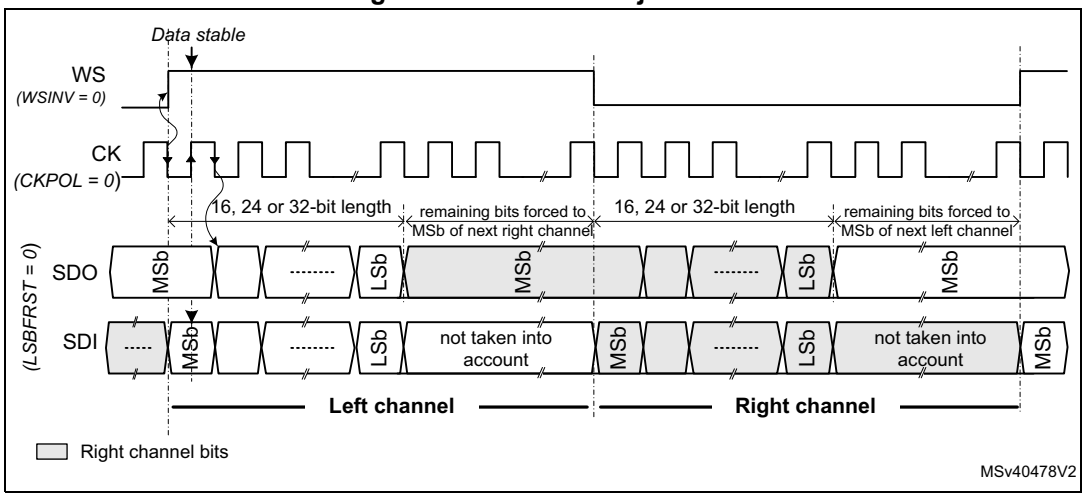
See *Control of the WS Inversion* for information concerning the handling of WS signal.

Figure 627. Master MSB justified 16 or 24-bit data length



In the case where the channel length is bigger than the data length, the remaining bits are forced to zero when the SPI/I2S is configured in master transmit mode. In slave transmit the remaining bits are forced to the value of the first bit of the next data to be generated in order to avoid timing issues (see *Figure 628*).

Figure 628. Slave MSB justified

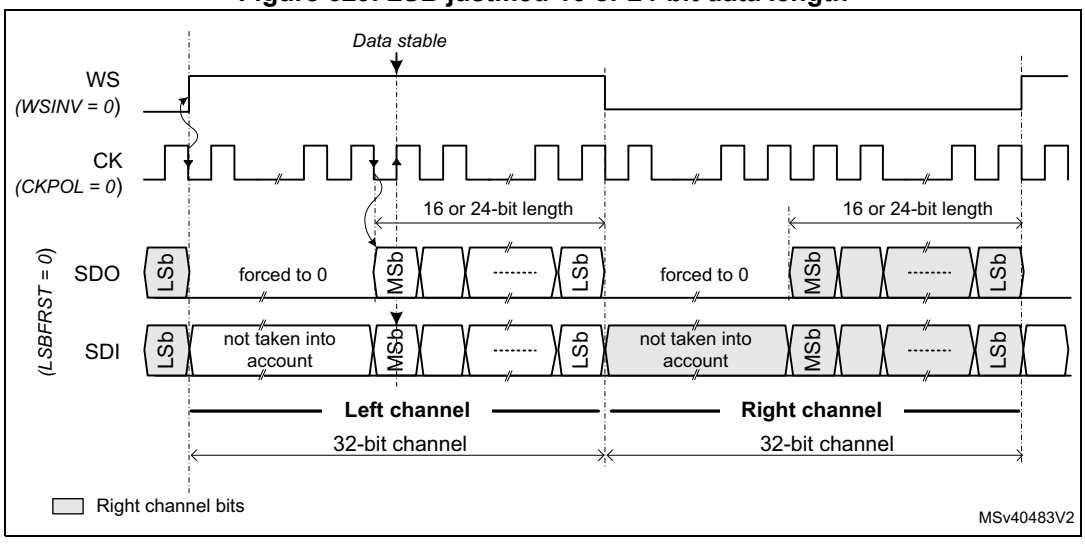


LSB justified standard

This standard is similar to the MSB justified standard in master mode (no difference for the 16 and 32-bit full-accuracy frame formats). The LSB justified 16 or 32-bit full-accuracy format give similar waveforms than MSB justified mode (see *Figure 626*) because the channel and data have the same length.

Note: In the LSB justified format, only 16 and 32-bit channel length are supported in master and slave mode. This is due to the fact that it is not possible to transfer properly the data if the channel length is not known by transmitter and receiver side.

Figure 629. LSB justified 16 or 24-bit data length



CKPOL is set to 0 in order to match the I2S LSB justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

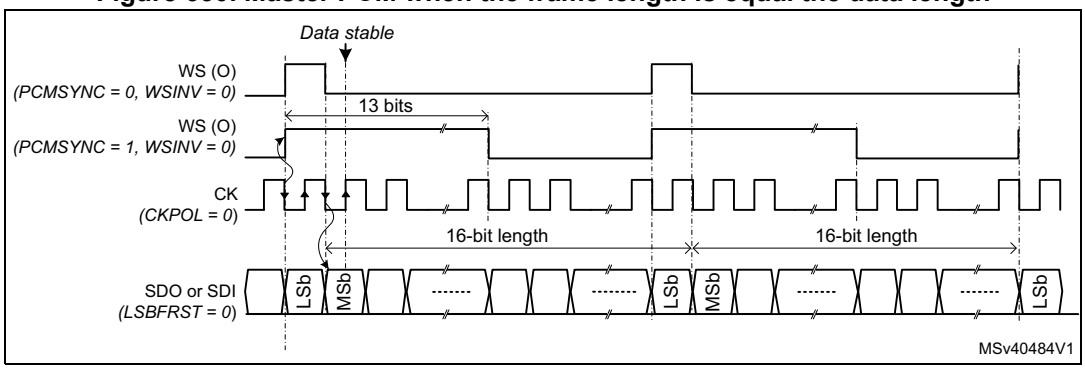
See *Control of the WS Inversion* for information concerning the handling of WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPI_I2SCFGR register.

Note: The difference between the PCM long and short frame, is just the width of the frame synchronization: for both protocols, the active edge of the frame is generated (or is expected for the Slave mode) one CK clock cycle before the first bit.

Figure 630. Master PCM when the frame length is equal the data length



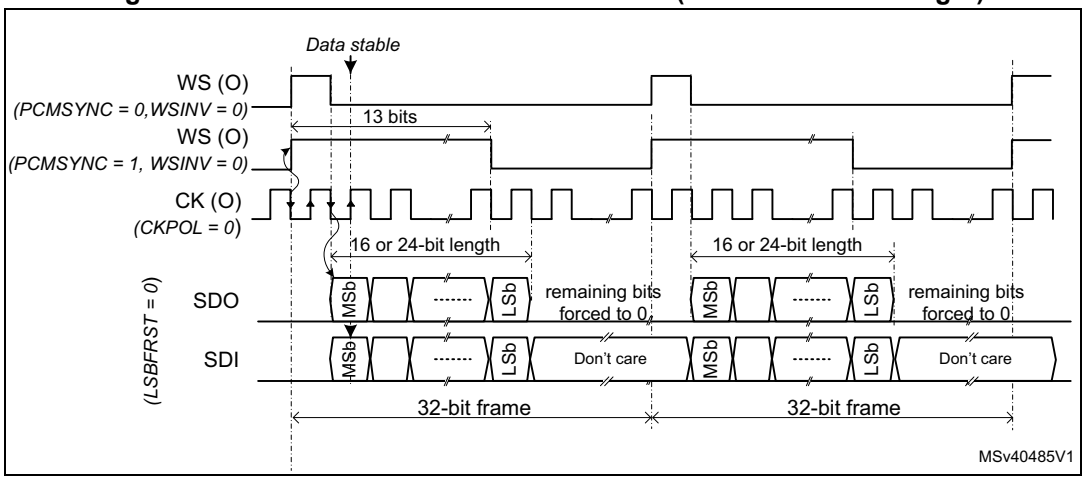
For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

For short frame synchronization, the WS synchronization signal is only one cycle long.

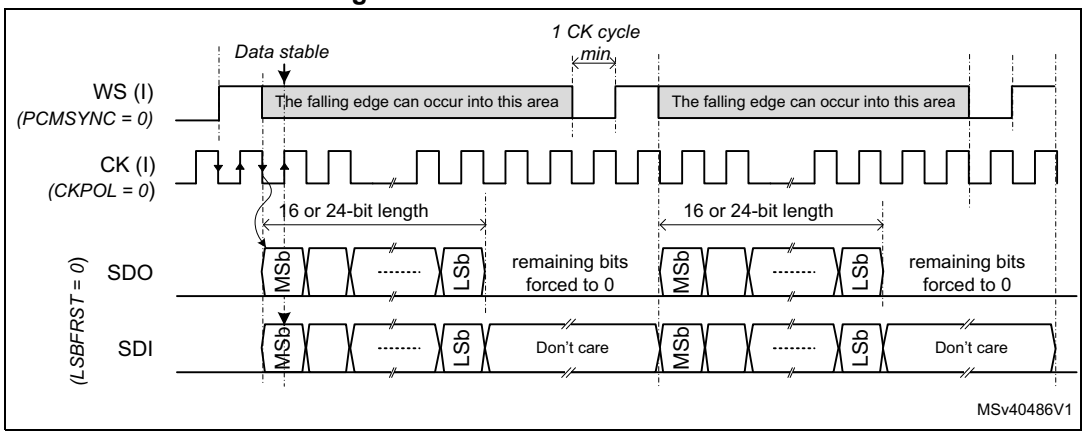
See *Control of the WS Inversion* for information concerning the handling of WS signal.

Figure 631. Master PCM standard waveforms (16 or 24-bit data length)



If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 632*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 632. Slave PCM waveforms



CKPOL is set to 0 in order to match the PCM protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

49.9.6 Additional Serial Interface Flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for I2S LSB justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS Inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV to 1. By default the WS polarity is the following:

- In I2S Philips Standard, WS is low for left channel, and high for right channel
- In MSB/LSB justified mode, WS is high for left channel, and low for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set to 1, the WS polarity is inverted, then:

- In I2S Philips Standard, WS is high for left channel, and low for right channel
- In MSB/LSB justified mode, WS is low for left channel, and high for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located into *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the IOs

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially set to 0

– Set I2SMOD = 1, (In order to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
– Set bits CKPOL and WSINV to the wanted value.
– Set AFCNTR = 1.
Then the inactive level of CK and WS IOs is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
– Then performs the activation sequence of the I2S/PCM

Table 402 shows the level of WS and CK signals, when the AFCNTR bit is set to 1, and before the SPI/I2S block is enabled (i.e. inactive level). Note that the level of WS depends also on the protocol selected.

Table 402. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR shall not be set to 1, when the SPI2S is in slave mode.

Serial Data Line swapping

The SPI/I2S offers the possibility to swap the function of SDI and SDO lines thanks to IOSWP bit located into *SPI configuration register 2 (SPI_CFG2)*. *Table 403* gives details on this feature.

Table 403. Serial data line swapping

Configuration	IOSWP	SDI direction	SDO direction
Master/slave RX	0	IN	-
	1	-	IN

Table 403. Serial data line swapping (continued)

Configuration	IOSWP	SDI direction	SDO direction
Master/slave TX	0	-	OUT
	1	OUT	-
Master/slave Full-duplex	0	IN	OUT
	1	OUT	IN

For simplification, the waveforms shown in the *I2S functional description* section have been done with IOSWP = 0.

49.9.7 Start-up sequence

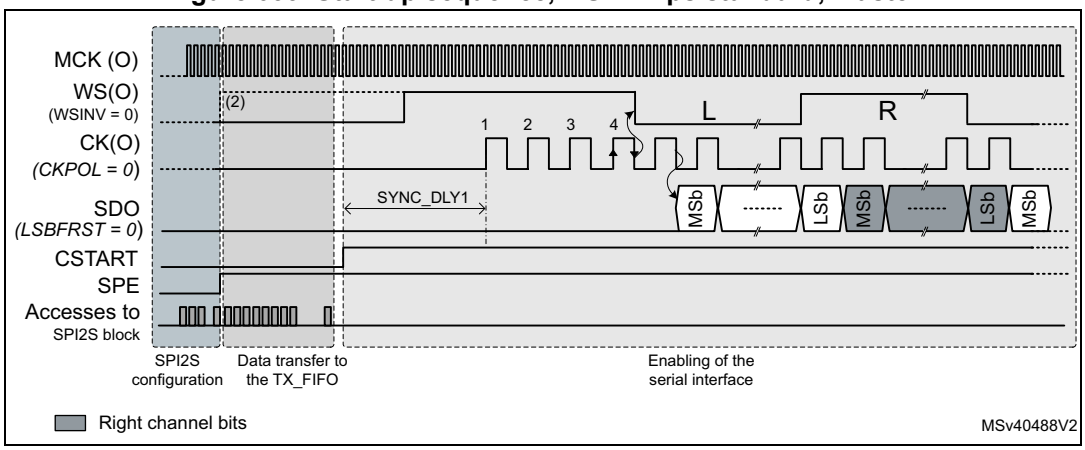
When the bit SPE is set to 0, the user is not allowed to read and write into the SPI2S_RXDR and SPI2S_TXDR registers, but the access to other registers is allowed.

When the application wants to use the SPI/I2S block the user has to proceed as follow:

1. Insure that the SPE is set to 0, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 49.9.16* for detailed programming examples.
3. Set the SPE bit to 1, in order to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO.
4. Set bit CSTART to 1, in order to activate the serial interface.

As shown in *Figure 633*, in I2S Philips standard master TX, the generation of the WS, MCK and CK signals is started as soon as the bit CSTART is set to 1 and the TxFIFO is not empty. Note that the bit clock CK is activated 4 rising edges before the falling edge of WS in order to insure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 633. Start-up sequence, I2S Philips standard, master



1. In this figure, the MCK is enabled before setting the bit SPE to 1. See *MCK Generation* for more information.
2. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set to 1

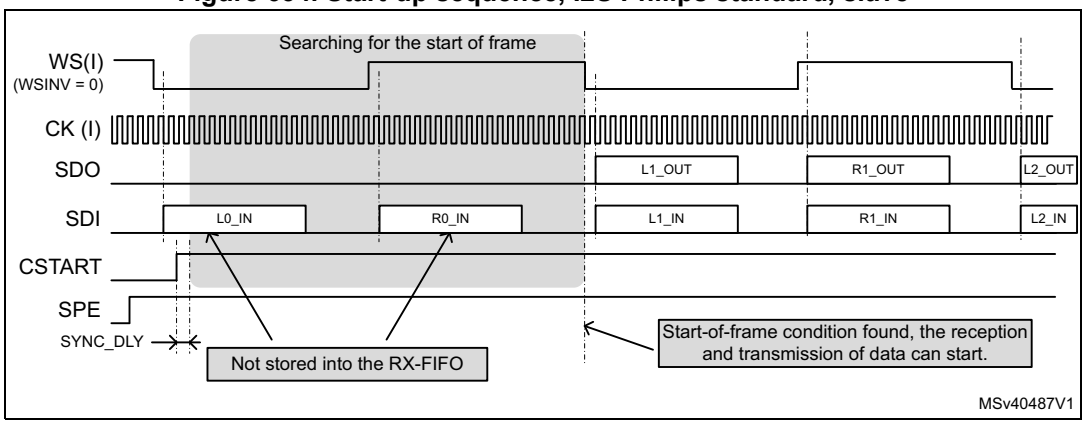
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about 3 periods of CK clock (SYNC_DLY1).

In slave mode, once the bit CSTART is set to 1, the data transfer starts when the start-of-frame condition is met:

- For I2S Philips standard, the start-of-frame condition is a falling edge of WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of WS signal. The transmission/reception starts at rising edge of WS for MSB aligned protocol.The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 634 shows an example of start-up sequence in I2S Philips standard, slave mode.

Figure 634. Start-up sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after 2 periods of CK clock (SYNC_DLY).

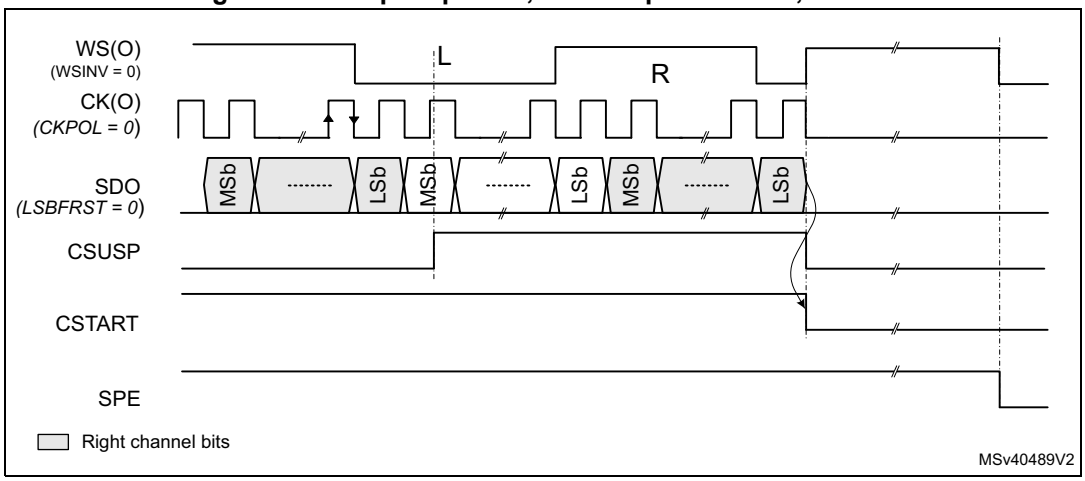
49.9.8 Stop sequence

The application can stop the I2S/PCM transfers by setting the SPE bit to 0. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP to 1, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be set to 0.

The *Figure 635* shows an example of stop sequence in the case of master mode. The CSUSP bit is set to 1, during the transmission of left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can set SPE to 0.

Figure 635. Stop sequence, I2S Philips standard, master

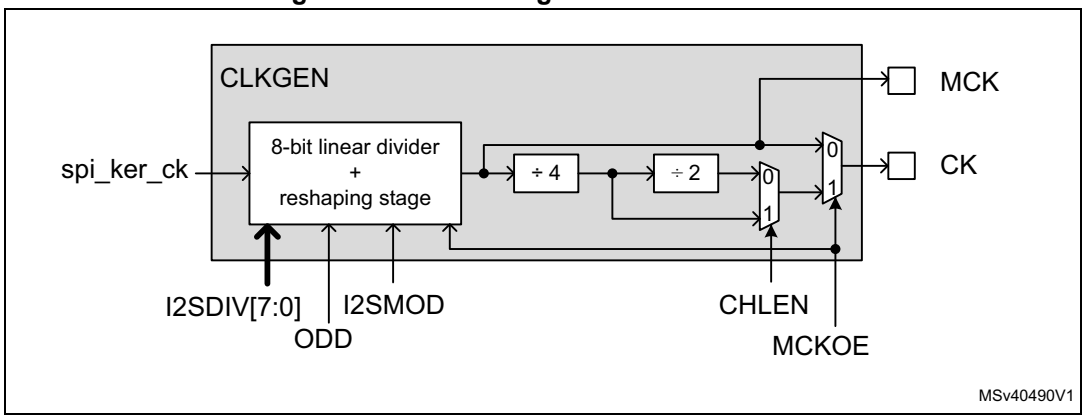


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

49.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the Frame Synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 636. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there is no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

and by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.
I2SDIV can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value I2SDIV = 1 is not allowed.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: When {(2 x I2SDIV) + ODD} is odd, the duty cycle of MCK or the CK signals is not 50%. Care must be taken when odd ratio is used: it can impact margin on setup and hold time. For example if {(2 x I2SDIV) + ODD} = 5, then the duty cycle can be 40%.

Table 404 provides examples of clock generator programming for I2S modes.

MCK Generation

The master clock MCK can be generated regardless to the SPE bit. The MCK generating is controlled by the following bits:

- I2SMOD must equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set to 1

Table 404. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

49.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI2S_TXDR register. The reading of RxFIFO is performed via the SPI2S_RXDR register.

Data alignment and ordering

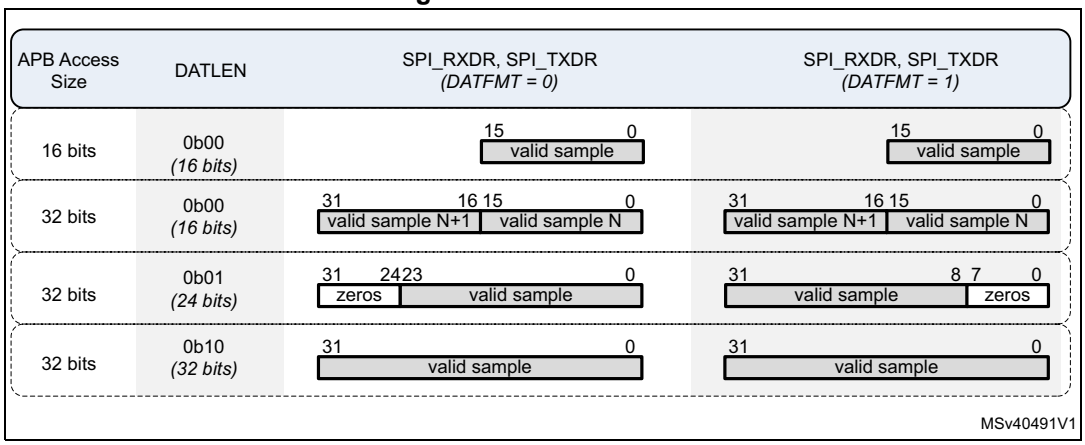
It is possible to select the data alignment into the SPI2S_RXDR and SPI2S_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI2S_RXDR or SPI2S_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 637 shows the allowed settings between APB access sizes, DATFMT and DATLEN.

Note: Caution shall be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 637. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is set to 0.

Refer to *Section 49.9.11* and *Section 49.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes 3 basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits),
- 4 samples, if the DATLEN = 2 (32 bits).

49.9.11 FIFOs status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. thus FTHLV new data to be transmitted can be written into SPI2S_TXDR. The TXP flag is reset when the amount of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI2S_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 49.10* for additional information on interrupt function in I2S mode.

49.9.12 Handling of underrun situation

In transmit mode, the UDR flag is set when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation at least a data is lost.

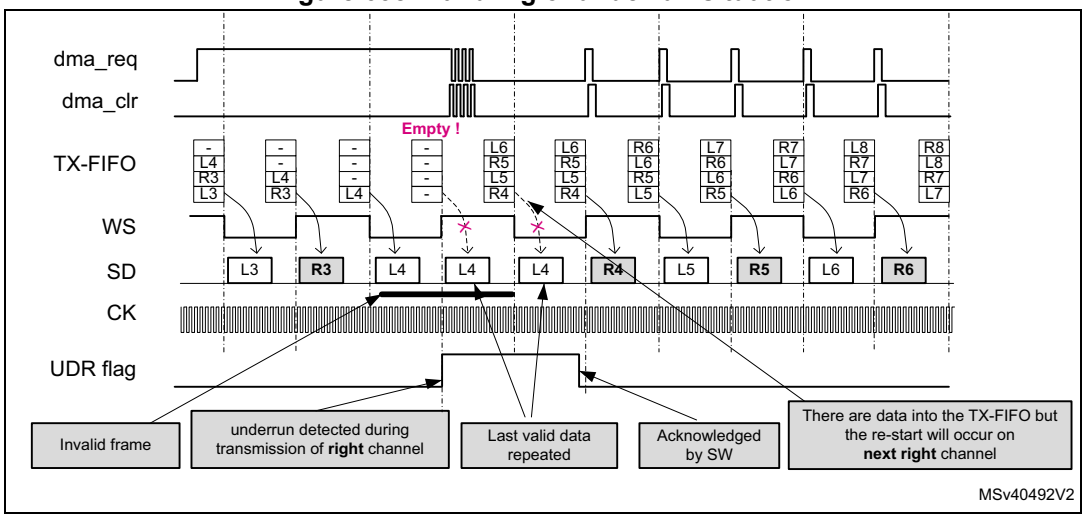
In I2S mode, there is a hardware mechanism in order to prevent misalignment situation (left and right channel swapped). As shown in the following figure, when an underrun occurs, the peripheral re-plays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software,

Then the next data transmitted is:

- A right channel if the underrun occurred when a right channel data needed to be transmitted, or
- A left channel if the underrun occurred when a left channel data needed to be transmitted.

Figure 638. Handling of underrun situation



The UDR flag can trigger an interrupt if the UDRIE bit in the SPI_IER register is set. The UDR bit is cleared by writing UDRC bit of SPI_IFCR register to 1.

When the block is configured in PCM mode, this re-alignment mechanism is not activated.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK and MCK signal are not gated.

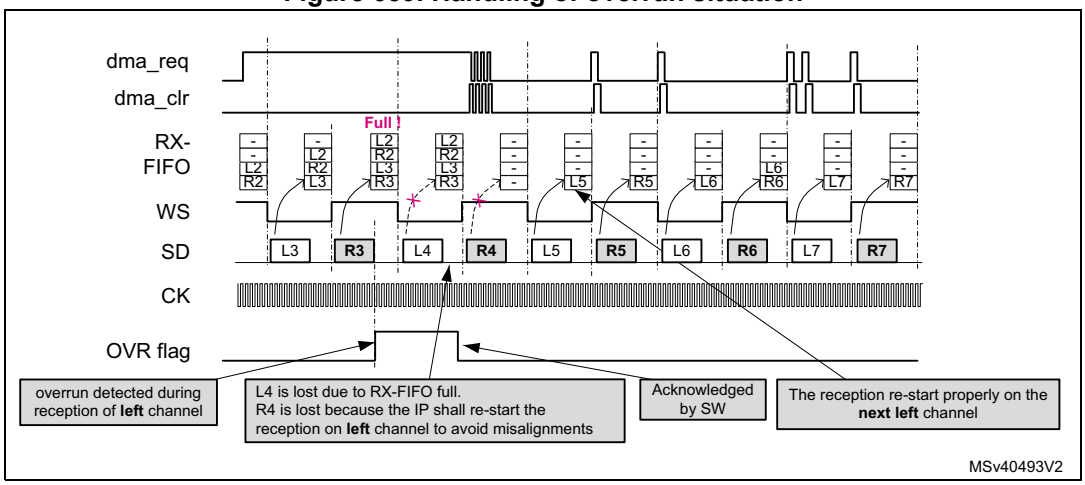
Due to resynchronization, any change on the UDR flag is taken into account by the hardware after at least 2 periods of CK clock.

49.9.13 Handling of overrun situation

The OVR flag is set when received data need to written into the RxFIFO, while the RxFIFO is already full. As a result, some incoming data are lost.

In I2S mode, there is a hardware mechanism in order to prevent misalignment situation (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as conditions of restart are not met. When there is enough room into the RxFIFO, and the OVR flag is cleared, the block starts by writing next the right channel into the RxFIFO if the overrun occurred when a right channel data was received or by writing the next left channel if the overrun occurred when a left channel data was received.

Figure 639. Handling of overrun situation



An interrupt may be generated if the OVRIE bit is set in the SPI_IER register. The OVR bit is cleared by writing OVRC bit of SPI_IFCR register to 1.

When the block is configured in PCM mode, this re-alignment mechanism is not activated

Note: An overrun situation can occur in master or slave mode. In master mode, when an overrun occurs, the WS, CK and MCK signal are not gated.

49.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

- A frame synchronization received while the shift-in or shift-out of the previous data is not completed (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurring at an unexpected position. This mode is selected with FIXCH = 1.

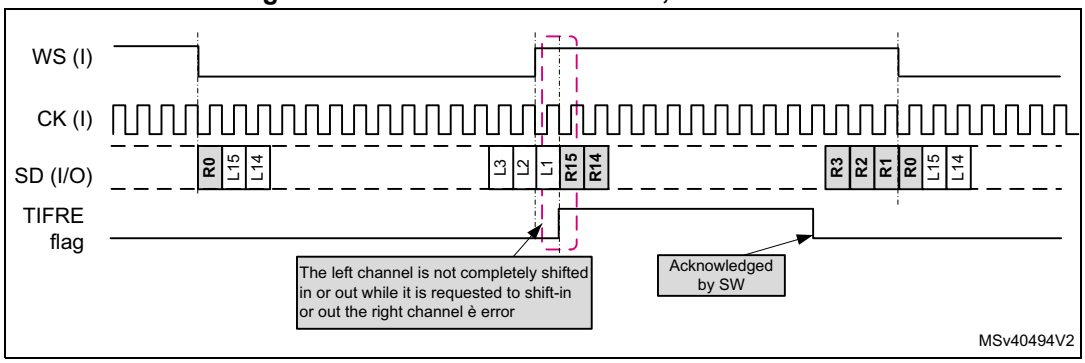
In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to set FIXCH to 0. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH to 1 and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock

between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set to 1.

Figure 640 shows an example of frame error detection. The SPI/I2S block is in slave mode and the amount of bit clock periods for left channel are not enough to shift-in or shift-out the data. The figure shows that the on-going transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 640. Frame error detection, with FIXCH=0

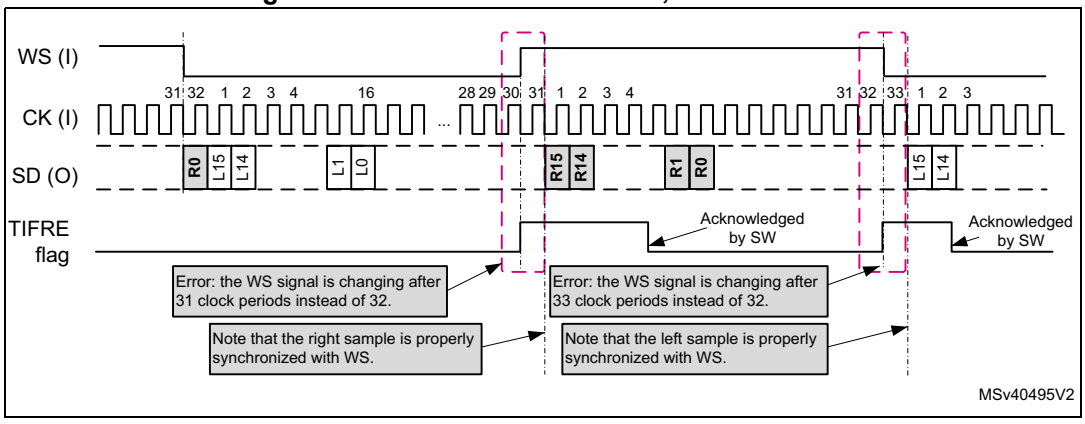


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH to 1. When this bit is set to 1, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 641*, in this mode the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 640* and *Figure 641* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 641. Frame error detection, with FIXCH=1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

49.9.15 DMA Interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of SPI_CFG1 register.

In receive mode, the DMA interface is working as follow:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follow:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

49.9.16 Programing examples

Master I2S Philips standard, transmit

This example shows how to program the interface for supporting the Philips I2S standard in master transmit mode, with a sampling rate of 48 kHz, using the master clock. The assumption taken is that SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit.

Start Procedure

1. Enable the bus interface clock (pclk or hclk), release the reset signal if needed in order to be able to program the SPI/I2S block.
2. Insure that the SPI/I2S block receives properly a kernel frequency (at 61.44 MHz in this example).
3. Insure that SPE is set to 0.
4. Program the clock generator in order to provide the MCK clock and to have a frame synchronization rate at exactly 48 kHz. So I2SDIV = 2, ODD = 1, and MCKOE = 1.
5. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (Philips Standard), I2SCFG = 2 (master transmit), I2SMOD = 1, for I2S/PCM mode. The register SPI_I2SCFGR must be updated before going to next steps.
6. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
7. Clear all status flag registers.
8. Enable the flags who shall generate an interrupt such as UDRIE. Note that TIFRE is not meaningful in master mode.
9. If the data transfer uses DMA:
   – Program the DMA peripheral,
   – Initialize the memory buffer with valid audio samples,
   – Enable the DMA channel,
10. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the TXPIE bit to 1.
11. Set SPE to 1, as soon as this bit is set to one the following actions may happen:
    – If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
    – If the DMA transfer are enabled (TXDMAEN = 1), the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
12. Finally, the user has to insure that the TxFIFO is not empty before enabling the serial interface. This is important in order to avoid an underrun condition when the interface is enabled. Then the SPI/I2S block can be enabled by setting the bit CSTART to 1. CSTART bit is located into SPI_CR1 register.

Stop Procedure in master mode

1. Set the bit CSUSP to 1, in order to stop on-going transfers
2. Check the value of CSTART bit until it goes to 0
3. Stop DMA peripheral, bus clock...
4. Set bit SPE to 0 in order to disable the SPI/I2S block

Master I2S MSB Aligned, full-duplex

This example shows how to program the interface for supporting the I2S MSB aligned protocol in master full-duplex mode, with a sampling rate of 48 kHz, without using the master clock. We took the assumption that the SPI/I2S is receiving a kernel clock (i2s_clk) of 12.288 MHz from the clock controller of the circuit.

Procedure

1. Enable the bus interface clock (pclk or hclk), release the reset signal if needed in order to be able to program the SPI/I2S block.
2. Insure that the SPI/I2S block receives properly a kernel frequency (at 12.288 MHz in this example).
3. Insure that SPE is set to 0.
4. Program the clock generator in order to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. So I2SDIV = 2, ODD = 0, and MCKOE = 0.
5. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 1 (MSB Justified), I2SCFG = 5 (master Full-duplex), I2SMOD = 1, for I2S/PCM mode. The register SPI_I2SCFGR must be updated before going to next steps.
6. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
7. Clear all status flag registers.
8. Enable the flags who shall generate an interrupt such as UDRIE. Note that TIFRE is not meaningful in master mode.
9. If the data transfer uses DMA:
   - Program the DMA peripheral: two channels, one for RX and one for TX
   - Initialize the memory buffer with valid audio samples for TX path
   - Enable the DMA channels,
   - In the SPI/I2S block, enable the DMA by setting the TXDMAEN and RXDMAEN bits to 1. As soon as these bits are set to 1, the SPI/I2S start to fill-up the TxFIFO by sending DMA requests
10. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the TXPIE and RXPIE bits to 1.
11. Set SPE to 1, as soon as this bit is set to one the following actions may happen:
    - If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
    - If the DMA transfer are enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
12. Finally, the user has to insure that the TxFIFO is not empty before enabling the serial interface. This is important in order to avoid an underrun condition when the interface is enabled. Then the SPI/I2S block can be enabled by setting the bit CSTART to 1. CSTART bit is located into SPI_CR1 register.

Refer to *Stop Procedure in master mode* for details on the stop sequence.

49.9.17 Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So we cannot use the capability offered for frame error detection when FIXCH is set to 1.

Procedure

1. Enable the bus interface clock (pclk or hclk), release the reset signal if needed in order to be able to program the SPI/I2S block.
2. Insure that SPE is set to 0.
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode. The register SPI_I2SCFGR must be properly programmed before going to next steps.
4. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags who shall generate an interrupt such as UDRIE and TIFRE.
7. If the data transfer uses DMA:
   – Program the DMA peripheral: one RX channel
   – Enable the DMA channel,
   – In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit to 1.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit to 1.
9. Set SPE to 1.
10. Finally the user can set the bit CSTART to 1 in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop Procedure in slave mode

1. Set bit SPE to 0 in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

49.10 I2S wakeup and interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wakeup event signal (**spi_wkup**) can be generated according to the events described in the *Table 405*.

Interrupt events can be enabled and disabled separately.

Table 405. I2S interrupt requests

Interrupt event	Event flag	Enable control bit	Event clear method	Interrupt/Wakeup activated	
				spi_it	spi_wkup
TxFIFO threshold reached	TXP	TXPIE	TXP flag is cleared when the TxFIFO contains less than FTHLV empty locations	YES	YES
RxFIFO threshold reached	RXP	RXPIE	RXP flag is cleared when the RxFIFO contains less than FTHLV samples		
Overrun error	OVR	OVRIE	OVR is cleared by writing OVRC to 1		
Underrun error	UDR	UDRIE	UDR is cleared by writing UDRC to 1		
Frame error flag	TIFRE	TIFREIE	TIFRE is cleared by writing TIFREC to 1		NO

49.11 SPI/I2S registers

49.11.1 SPI/I2S control register 1 (SPI2S_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IO LOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRC INI	RCRC INI	CRC33 _17	SSI	HDDIR	CSUSP	C START	MAS RX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated IOs
This bit is set by software and cleared by hardware on next device reset
0: AF configuration is not locked
1: AF configuration is locked
When this bit is set, SPI_CFG2 register content cannot be modified any more.
This bit must be configured when the SPI is disabled. When SPE=1, it is write protected.

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter
0: All zero pattern is applied
1: All ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver
0: all zero pattern is applied
1: all ones pattern is applied

Bit 13 **CRC33_17**: 32-bit CRC polynomial configuration
0: full size (33-bit or 17-bit) CRC polynomial is not used
1: full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** internal SS signal input level
This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral SS input and the I/O value of the SS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at Half-duplex mode
In Half-Duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.
0: SPI is Receiver
1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request
This bit reads as zero.
In master mode, when this bit is set by software, CSTART bit is reset at the end of the current frame and SPI communication is suspended. The user has to check SUSP flag to check end of the frame transaction.
The master mode communication must be suspended (using this bit or keeping TXDR empty) before disabling the SPI or going to low-power mode.

Bit 9 **CSTART:** master transfer start

This bit is set by software to start an SPI transfer in master mode. It is cleared by hardware when End Of Transfer (EOT) flag is set or when an CSUSPend request is accepted.

0: master transfer is at idle

1: master transfer is on-going or temporary suspended by automatic suspend

In SPI mode, CSTART can be set only when SPE=1 and MASTER=1.

In SPI mode, In case of master transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

In I2S/PCM mode, CSTART can be set when SPE = 1.

Bit 8 **MASRX:** master automatic SUSP in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when SPI communication is suspended.

When SPI communication is suspended to prevent overrun condition it could happen that few bits of next frame are already clocked out due to internal synchronization delay. Once the RxFIFO is read the communication resumes and continues by subsequent bits transaction without any next constrain.

For the same reason, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

0: serial peripheral disabled.

1: serial peripheral enabled

When SPE=1, SPI data transfer is enabled, Configuration registers and IOLOCK bit in SPI_CR1 are write protected. They can be changed only when SPE=0.

When SPI=0 any SPI operation is stopped and disabled, internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.

SPE cannot be set when MODF error flag is active.

49.11.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSER[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TSER[15:0]**: number of data transfer extension to be reload into TSIZE just when a previous number of data stored at TSIZE is transacted

This register can be set by software when its content is cleared only. It is cleared by hardware once TSIZE reload is done. The TSER value must be programmed in advance before CTSIZE counter reaches zero otherwise the reload is not taken into account and traffic terminates with normal EOT event.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

These bits are changed by software. The value must not be changed while CSTART bit is set. Endless transaction is initialized when CSTART is set while zero value is stored at TSIZE. TSIZE cannot be set to 0xFFFF value when CRC is enabled.

49.11.3 SPI configuration register 1 (SPI_CFG1)

Address offset: 0x08

Reset value: 0x0007 0007

Content of this register is write protected when SPI is enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRC EN	Res.	CRCSIZE[4:0]				
	rw	rw	rw						rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX DMA EN	RX DMA EN	Res.	UDRDET[1:0]		UDRCFG[1:0]		FTHLV[3:0]				DSIZE[4:0]				
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MBR[2:0]:** master baud rate

000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable

0: CRC calculation disabled
1: CRC calculation Enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transacted and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transacted or compared. The length of the polynomial is not affected by this setting.
00000: not used
00001: not used
00010: not used
00011: 4-bits
00100: 5-bits
00101: 6-bits
00110: 7-bits
00111: 8-bits
.....
11101: 30-bits
11110: 31-bits
11111: 32-bits
The value must be equal or multiply of DSIZE length

Note: The most significant bit at CRCSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.

Bit 15 **TXDMAEN:** Tx DMA stream enable
0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable
0: Rx-DMA disabled
1: Rx-DMA enabled

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **UDRDET[1:0]:** detection of underrun condition at slave transmitter
00: underrun is detected at begin of data frame (no protection of 1-st bit)
01: underrun is detected at end of last data frame
10: underrun is detected by begin of active SS signal
11: reserved
The user can define here when and how the underrun condition is detected at slave receiver

Bits 10:9 **UDRCFG[1:0]:** behavior of slave transmitter at underrun condition

00: slave sends a constant pattern defined by the user at SPI_UDRDR register

01: slave repeats lastly received data frame from master

10: slave repeats its lastly transmitted data frame

11: reserved

While TxFIFO is empty shift register for transmission keeps value of SPI_UDRDR register. This register can be either locked to send value defined by the user or refreshed automatically by lastly transacted frame received from master or updated by data stored lastly at slave TxFIFO (by write to TXDR).

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames at single data packet. The size of the packet must not exceed 1/2 of FIFO space.

0000: 1-data

0001: 2-data

0010: 3-data

0011: 4-data

0100: 5-data

0101: 6-data

0110: 7-data

0111: 8-data

1000: 9-data

1001: 10-data

1010: 11-data

1011: 12-data

1100: 13-data

1101: 14-data

1110: 15-data

1111: 16-data

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

– If SPI data register is accessed as a 16-bit register and DSIZE<=8bit, better to select FTHLV=2, 4, 6 etc,

– If SPI data register is accessed as a 32-bit register and DSIZE>8bit, better to select FTHLV=2, 4, 6 etc, while if DSIZE<=8bit, better to select FTHLV=4, 8, 12 etc

Bits 4:0 **DSIZE[4:0]:** number of bits in at single SPI data frame

00000: not used

00001: not used

00010: not used

00011: 4-bits

00100: 5-bits

00101: 6-bits

00110: 7-bits

00111: 8-bits

.....

11101: 30-bits

11110: 31-bits

11111: 32-bits

Note: The most significant bit at DSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.

49.11.4 SPI configuration register 2 (SPI_CFG2)

Address offset: 0x0C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled or IOLOCK bit is set at SPI2S_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AF CNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSB FRST	MAS TER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE=0 only

0: the peripheral takes no control of GPIOs while it is disabled

1: the peripheral keeps always control of all associated GPIOs

When SPI master has to be disabled temporary for a specific configuration reason (e.g. CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration. This bit must be never used at slave mode as any slave transmitter must not force its MISO output once the SPI is disabled.

Note: This bit can be also used in PCM and I2S modes.

Bit 30 **SSOM**: SS output management in master mode

This bit is used in master mode when SSOE is enabled. It allows to configure SS output between two consecutive data transfers.

0: SS is kept at active level till data transfer is completed, it becomes inactive with EOT flag

1: SPI data frames are interleaved with SS non active pulses when MIDI[3:0]>1

Bit 29 **SSOE:** SS output enable

This bit is taken into account at master mode only

0: SS output is disabled and the SPI can work in multi-master configuration

1: SS output is enabled. The SPI cannot work in a multi-master environment. It forces the SS pin at inactive level after the transfer in according with SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP:** SS input/output polarity

0: low level is active for SS signal

1: high level is active for SS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** software management of SS signal input

0: SS input value is determined by the SS PAD

1: SS input value is determined by the SSI bit

SS signal input has to be managed by software (SSM=1, SSI=1) when SS output mode is enabled (SSOE=1) at master mode.

Bit 25 **CPOL:** clock polarity

0: SCK signal is at 0 when idle

1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master
0: SPI Slave
1: SPI Master

Bits 21:19 **SP[2:0]:** Serial protocol
000: SPI Motorola
001: SPI TI
others: Reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI communication mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: salf-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
0: no swap
1: MOSI and MISO are swapped
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted.
Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
Note: This bit can be also used in PCM and I2S modes.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: master Inter-Data Idleness
Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.
0000: no delay
0001: 1 clock cycle period delay
...
1111: 15 clock cycle periods delay
Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: master SS idleness
Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of SS and first data transaction start in master mode when SSOE is enabled.
0000: no extra delay
0001: 1 clock cycle period delay added
...
1111: 15 clock cycle periods delay added
Note: This feature is not supported in TI mode.

49.11.5 SPI/I2S interrupt enable register (SPI2S_IER)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TSERFIE	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
					rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **TSERFIE**: additional number of transactions reload interrupt enable
0: TSERF interrupt disabled
1: TSERF interrupt enabled

Bit 9 **MODFIE**: mode fault interrupt enable
0: MODF interrupt disabled
1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable
0: TIFRE interrupt disabled
1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable
0: CRC interrupt disabled
1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable
0: OVR interrupt disabled
1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTFIE interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP Interrupt Enable
0: RXP interrupt disabled
1: RXP interrupt enabled

49.11.6 SPI/I2S status register (SPI2S_SR)

Address offset: 0x14

Reset value: 0x0000 1002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	TSERF	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus

Bit 15 **RXWNE:** RxFIFO word not empty
RxFIFO contains more than one 32-bit data
0: less than 32-bit data frame available in RxFIFO
1: at least one 32-bit data is available in the RxFIFO

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing leveL
Number of packed frames available in the last 32-bit word of the RxFIFO. This number of frames is read from the RxFIFO when RXWNE=0.
When frame size is smaller than or equal to 8-bit
00: the number of frames in RxFIFO is 0 or a multiple of 4.
01: 1 frame available when RXWNE=0
10: 2 frames available when RXWNE=0
11: 3 frames available when RXWNE=0
When the frame size is smaller than or equal to 16-bit, but larger than 8-bit, RXPLV[1:0] can take the values 0 or 1.
00: the number of frames in RxFIFO is 0 or a multiple of 2.
01: the number of frame in RxFIFO is odd. One 16-bit data is available when RXWNE=0
Other values forbidden.
When the frame size is greater than 16-bit, these bits read as 00.
00: -
Other values forbidden.

Bit 12 **TXC:** TxFIFO transmission complete

This flag is changed by hardware.

When TSIZE=0 the TXC raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE <>0 the TXC raises at the end of transfer no matter on TxFIFO occupancy.

When this flag is set, master transmission is completed. The flag is not reliable to consider end of master reception.When CRC mode is enabled, TXC is set only after the CRC transmission. TXC generates an interrupt when EOTIE is set.

0: Current data transaction is still ongoing, data is available in TxFIFO or last frame transmission is on going (including CRC).

1: Last TxFIFO or CRC frame transmission completed

Bit 11 **SUSP:** suspend

In Master mode, SUSP is set by hardware when a CSUSP request is done, as soon as the current frame is completed or at master automatic suspend receive mode (MASRX bit is set at SPI2S_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit is cleared by write 1 to SUSPC bit at SPI2S_IFCR

0: SPI not suspended (master mode active or other mode).

1: Master mode SUSPended (Current Frame completed)

Bit 10 **TSERF**: additional number of SPI data to be transacted was reload

This bit is cleared by write 1 to TSERFC bit at SPI2S_IFCR or by writing the TSER[15:0] (SPI_CR2) register

0: no acceptation

1: additional number of data accepted, current transaction continues

Bit 9 **MODF**: mode fault

0: no mode fault

1: mode fault detected

This bit is cleared by write 1 to MODFC bit at SPI2S_IFCR

Bit 8 **TIFRE**: TI frame format error

0: no TI Frame Error

1: TI Frame Error detected

This bit is cleared by write 1 to TIFREC bit at SPI2S_IFCR

Bit 7 **CRCE:** CRC error

0: no CRC error

1: CRC error detected

This bit is cleared by write 1 to CRCEC bit at SPI2S_IFCR

Bit 6 **OVR:** overrun

0: no overrun

1: overrun detected

This bit is cleared by write 1 to OVRC bit at SPI2S_IFCR

Bit 5 **UDR:** underrun at slave transmission mode

0: no underrun

1: underrun detected

This bit is cleared by write 1 to UDRC bit at SPI2S_IFCR

Note: UDR flag applies to Slave mode only

Bit 4 **TXTF:** transmission transfer filled

0: upload of TxFIFO is on-going or not started

1: TxFIFO upload is finished

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to TXTFC bit at SPI2S_IFCR

TXTF flag triggers an interrupt if TXTFIE bit is set.

TXTF setting clears the TXIE and DPXIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared by software write 1 to EOTC bit at SPI2S_IFCR.

EOT flag triggers an interrupt if EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

0: transfer is on-going or not started

1: transfer complete

In master, EOT event terminates the data transaction and handles SS output optionally. In both master and slave, the event handles CRC transaction.

Bit 2 **DXP**: duplex packet

0: TxFIFO is Full and/or RxFIFO is Empty

1: Both TxFIFO has space for write and RxFIFO contains for read a single packet at least

DXP flag is set when both TXP and RXP flags are set.

Bit 1 **TXP:** Tx-packet space available

0: there is not enough space to locate next data packet at TxFIFO

1: TxFIFO has enough free location to host 1 data packet

TXP flag is changed by hardware. It monitors overall space currently available at TxFIFO if SPI is enabled. It has to be checked once a complete data packet is stored at TxFIFO.

Bit 0 **RXP:** Rx-packet available

0: RxFIFO is empty or a not complete data packet is received

1: RxFIFO contains at least 1 data packet

RXP flag is changed by hardware. It monitors number of overall data currently available at RxFIFO if SPI is enabled. It has to be checked once a data packet is completely read out from RxFIFO.

49.11.7 SPI/I2S interrupt/status flags clear register (SPI2S_IFCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	TSERFC	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w	w	w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** SUSPend flag clear
Writing a 1 into this bit clears SUSP flag in the SPI2S_SR register

Bit 10 **TSERFC**: TSERFC flag clear
Writing a 1 into this bit clears TSERF flag in the SPI2S_SR register
Note: *TSERF is also reset by writing the TSER[15:0] (SPI_CR2) register*

Bit 9 **MODFC**: mode fault flag clear
Writing a 1 into this bit clears MODF flag in the SPI2S_SR register

Bit 8 **TIFREC**: TI frame format error flag clear
Writing a 1 into this bit clears TIFRE flag in the SPI2S_SR register

Bit 7 **CRCEC:** CRC error flag clear
Writing a 1 into this bit clears CRCE flag in the SPI2S_SR register

Bit 6 **OVRC:** overrun flag clear
Writing a 1 into this bit clears OVR flag in the SPI2S_SR register

Bit 5 **UDRC:** underrun flag clear
Writing a 1 into this bit clears UDR flag in the SPI2S_SR register

Bit 4 **TXTFC:** transmission Transfer Filled flag clear
Writing a 1 into this bit clears TXTF flag in the SPI2S_SR register

Bit 3 **EOTC:** end of transfer flag clear
Writing a 1 into this bit clears EOT flag in the SPI2S_SR register

Bits 2:0 Reserved, must be kept at reset value.

49.11.8 SPI/I2S transmit data register (SPI2S_TXDR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TXDR[31:0]:** transmit data register
The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.
halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be written by single access.
word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be written by single access.

49.11.9 SPI/I2S receive data register (SPI2S_RXDR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: Data is always right-aligned. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be read by single access

49.11.10 SPI polynomial register (SPI_CRCPOLY)

Address offset: 0x40

Reset value: 0x0000 0107

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCPOLY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used at some other ST products with fixed length of the polynomial string where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored at this register. It has to be set greater than DSIZE. CRC33_17 bit has to be set additionally with CRCPOLY register when DSIZE is configured to maximum 32-bit or 16-bit size and CRC is enabled (to keep polynomial length grater than data size).

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

49.11.11 SPI transmitter CRC register (SPI_TXCRC)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: A read to this register when the communication is ongoing could return an incorrect value.

Not used for the I^2S mode.

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

49.11.12 SPI receiver CRC register (SPI_RXCRC)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.
The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: A read to this register when the communication is ongoing could return an incorrect value.

Not used for the I^2S mode.

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

49.11.13 SPI underrun data register (SPI_UDRDR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDRDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account at slave mode and at underrun condition only. The number of bits considered depends on DSIZE bit settings at SPI_CFG1 register. Underrun condition handling depends on setting if UDRDET and UDRCFG bits at SPI_CFG1 register.
Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

49.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x50

Reset value: 0x0000 0000

Note: All documented bits in this register must be configured when the I2S is disabled (SPE = 0). These bits are not used in SPI mode except for I2SMOD which needs to be set to 0 in SPI mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCK OE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DAT FMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCM SYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2S MOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MCKOE**: master clock output enable
0: Master clock output is disabled
1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler
0: Real divider value is = I2SDIV *2
1: Real divider value is = (I2SDIV * 2) + 1
Refer to *Section 49.9.9: Clock generator* for details

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
I2SDIV can take any values except the value 1, when ODD is also equal to 1.
Refer to *Section 49.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: the data inside the SPI2S_RXDR or SPI2S_TXDR are right aligned
1: the data inside the SPI2S_RXDR or SPI2S_TXDR are left aligned.

Bit 13 **WSINV**: Word select inversion
This bit is used to invert the default polarity of WS signal.
0: in I2S Philips standard, Left channel is transfered when WS is low, and right channel when WS is high.
In MSB or LSB justified mode, Left channel is transfered when WS is high, and right channel when WS is low.
In PCM mode the start of frame is indicated by a rising edge.
1: in I2S Philips standard, Left channel is transfered when WS is high, and right channel when WS is low.
In MSB or LSB justified mode, Left channel is transfered when WS is low, and right channel when WS is high.
In PCM mode the start of frame is indicated by a falling edge.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN not taken into account)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide
The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.

Bits 9:8 **DATLEN[1:0]**: data length to be transferred
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: not allowed

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I^2S standard selection
00: I^2S Philips standard.
01: MSB justified standard (left justified)
10: LSB justified standard (right justified)
11: PCM standard
For more details on I^2S standards, refer to *Section 49.9.5: Supported audio protocols*

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - full duplex
101: master - full duplex
others, not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

49.12 SPI register map and reset values

Table 406. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPI2S_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x04	**SPI_CR2**	TSER[15:0]																TSIZE[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**SPI_CFG1**	Res.	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]					TXDMAEN	RXDMAEN	Res.	UDRDET[1:0]		UDRCFG[1:0]		FTHLV[3:0]				DSIZE[4:0]				
	Reset value		0	0	0						0		1	1	1	1	1	0	0		0	0	0	0	0	0	0	0	0	0	1	1	1
0x0C	**SPI_CFG2**	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]				
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0								0	0	0	0	0	0	0	0
0x10	**SPI2S_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSERFIE	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x14	**SPI2S_SR**	CTSIZE[15:0]																RXWNE	RXPLVL[1:0]		TXC	SUSP	TSERF	MODF	TIFRE	CRCERR	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
0x18	**SPI2S_IFCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	TSERFC	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0	0	0	0	0	0	0	0	0			
0x20	**SPI2S_TXDR**	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24 - 0x2C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x30	**SPI2S_RXDR**	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34 - 0x3C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x40	**SPI_CRCPOLY**	CRCPOLY[31:16](1)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x44	**SPI_TXCRC**	TXCRC[31:16](1)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**SPI_RXCRC**	RXCRC[31:16](1)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**SPI_UDRDR**	UDRDR[31:16](1)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 406. SPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	**SPI_I2SCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	Reset value							0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0

1. Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored. *Refer to Table register boundary addresses.*

50 Serial audio interface (SAI)

50.1 Introduction

The SAI interface (Serial Audio Interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many stereo or mono audio applications may be targeted. I2S standards, LSB or MSB-justified, PCM/DSP, TDM, and AC'97 protocols may be addressed for example. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio sub-blocks. Each block has it own clock generator and I/O line controller.

The SAI can work in master or slave configuration. The audio sub-blocks can be either receiver or transmitter and can work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIs to work synchronously.

50.2 SAI main features

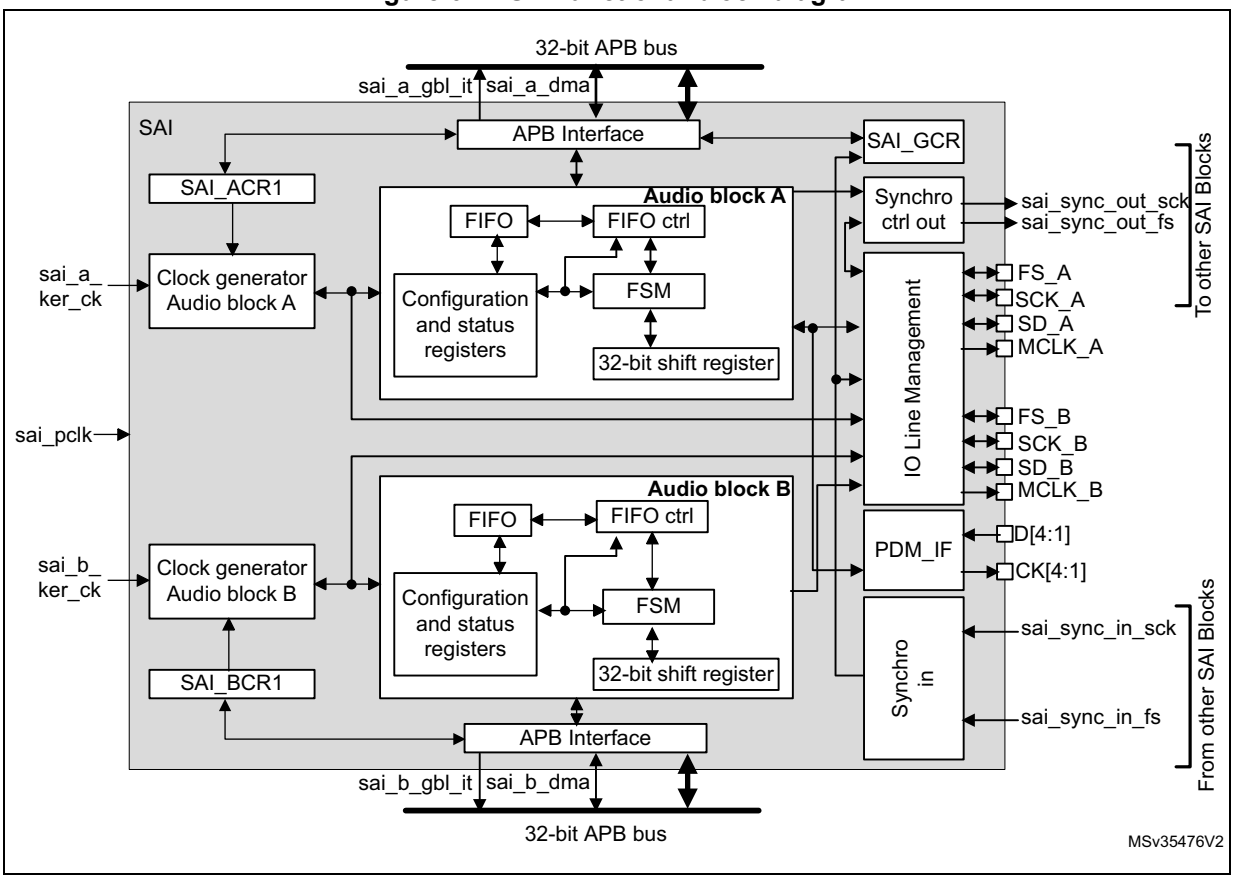
- Two independent audio sub-blocks which can be transmitters or receivers with their respective FIFO.
- 8-word integrated FIFOs for each audio sub-block.
- Synchronous or asynchronous mode between the audio sub-blocks.
- Possible synchronization between multiple SAIs.
- Master or slave configuration independent for both audio sub-blocks.
- Clock generator for each audio block to target independent audio frequency sampling when both audio sub-blocks are configured in master mode.
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit.
- Audio protocol: I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required.
- Up to 16 slots available with configurable size.
- Number of bits by frame can be configurable.
- Frame synchronization active level configurable (offset, bit length, level).
- First active bit position in the slot is configurable.
- LSB first or MSB first for data transfer.
- Mute mode.
- Stereo/Mono audio frame capability.
- Communication clock strobing edge configurable (SCK).
- Error flags with associated interrupts if enabled respectively.
  - Overrun and underrun detection,
  - Anticipated frame synchronization signal detection in slave mode,
  - Late frame synchronization signal detection in slave mode,
  - Codec not ready for the AC’97 mode in reception.
- Interruption sources when enabled:
  - Errors,
  - FIFO requests.
- 2-channel DMA interface.

50.3 SAI functional description

50.3.1 SAI block diagram

Figure 642 shows the SAI block diagram while *Table 407* and *Table 408* list SAI internal and external signals.

Figure 642. SAI functional block diagram



The SAI is mainly composed of two audio sub-blocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA in order to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two sub-blocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set-up the desired protocols (audio frame waveform generator).

The audio sub-block can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC'97 protocol, it will be an SAI output even if the SAI (link controller) is set-up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B sub-block.

50.3.2 SAI pins and internal signals

Table 407. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts.
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests.
sai_sync_out_sck, sai_sync_out_fs	Output	Internal clock and frame synchronization output signals exchanged with other SAI blocks.
sai_sync_in_sck, sai_sync_in_fs	Input	Internal clock and frame synchronization input signals exchanged with other SAI blocks.
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock.
sai_pclk	Input	APB clock.

Table 408. SAI input/output pins

Name	Signal type	Comments
SAI_SCK_A/B	Input/output	Audio block A/B bit clock.
SAI_MCLK_A/B	Output	Audio block A/B master clock.
SAI_SD_A/B	Input/output	Data line for block A/B.
SAI_FS_A/B	Input/output	Frame synchronization line for audio block A/B.
SAI_CK[4:1]	Output	PDM bitstream clock.
SAI_D[4:1]	Input	PDM bitstream data.

50.3.3 Main SAI modes

Each audio sub-block of the SAI can be configured to be master or slave via MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI sub-block is configured in asynchronous mode, then SCK_x and FS_x pins are configured as inputs.
- If the SAI sub-block is configured to operate synchronously with another SAI interface or with the second audio sub-block, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio sub-block can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, SAI_SD_x pin will be respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI are enabled by SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock line, data line and synchronization line in slave mode.

In master TX mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO, However FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave TX mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold *(*FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

50.3.4 SAI synchronization mode

There are two levels of synchronization, either at audio sub-block level or at SAI level.

Internal synchronization

An audio sub-block can be configured to operate synchronously with the second audio sub-block in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

External synchronization

The audio sub-blocks can also be configured to operate synchronously with another SAI. This can be done as follow:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio sub-block is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.
2. The SAI which shall receive the synchronization signals has to select which SAI will provide the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio sub-blocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

Note: SYNCIN[1:0] and SYNCOUT[1:0] bits are located into the SAI_GCR register, and SYNCEN bits into SAI_xCR1 register.

If both audio sub-blocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as ‘Res.’ shall not be used.

Table 409. External synchronization selection

Block instance	SYNCIN= 3	SYNCIN= 2	SYNCIN= 1	SYNCIN= 0
SAI1	SAI4 sync	SAI3 sync	SAI2 sync	Res.
SAI2	SAI4 sync	SAI3 sync	Res.	SAI1 sync
SAI3	SAI4 sync	Res.	SAI2 sync	SAI1 sync
SAI4	Res.	SAI3 sync	SAI2 sync	SAI1 sync

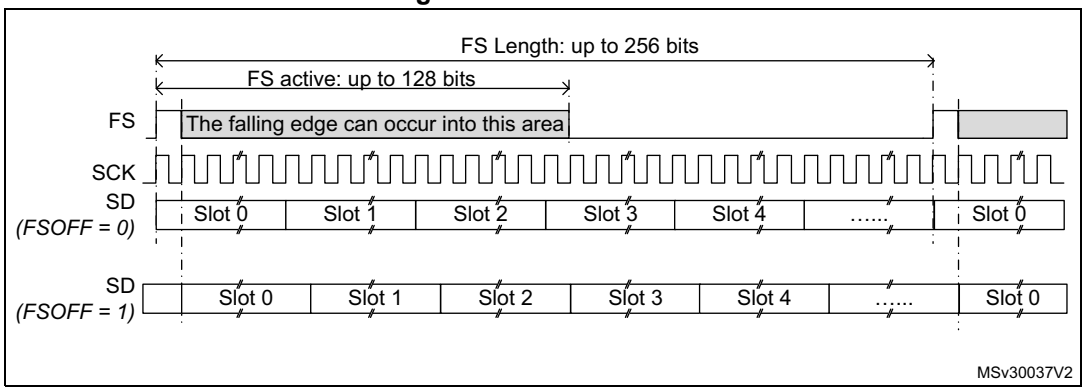
50.3.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24 or 32 bits. During the transfer, either the MSB or the LSB of the data are sent first, depending on the configuration of bit LSBFIRST in the SAI_xCR1 register.

50.3.6 Frame synchronization

The FS signal acts as the Frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable in order to target the different audio protocols with their own specificities concerning this Frame synchronization behavior. This reconfigurability is done using register SAI_xFRCR. *Figure 643* illustrates this flexibility.

Figure 643. Audio frame



In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC'97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256 bit clock cycles, by setting FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit will be extended to 0 or the SD line will be released to HI-z depending the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

  If bit NOMCK is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

  If bit NOMCK is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 50.3.8: SAI clock generator*".

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the Frame synchronization signal during an on-going audio frame. In this case an error will be generated. For more details refer to *Section 50.3.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. Start of frame is synchronized to this signal. It is effective only if the start of frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 50.3.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition will be managed as described in *Section 50.3.14: Error flags*), but the audio communication flow will not be interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register allow configuring the length of the active level of the Frame synchronization signal. The length can be set from 1 to 128 bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM mode.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the Frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). FSOFF bit in the SAI_xFRCR register allows to choose one of the two configurations.

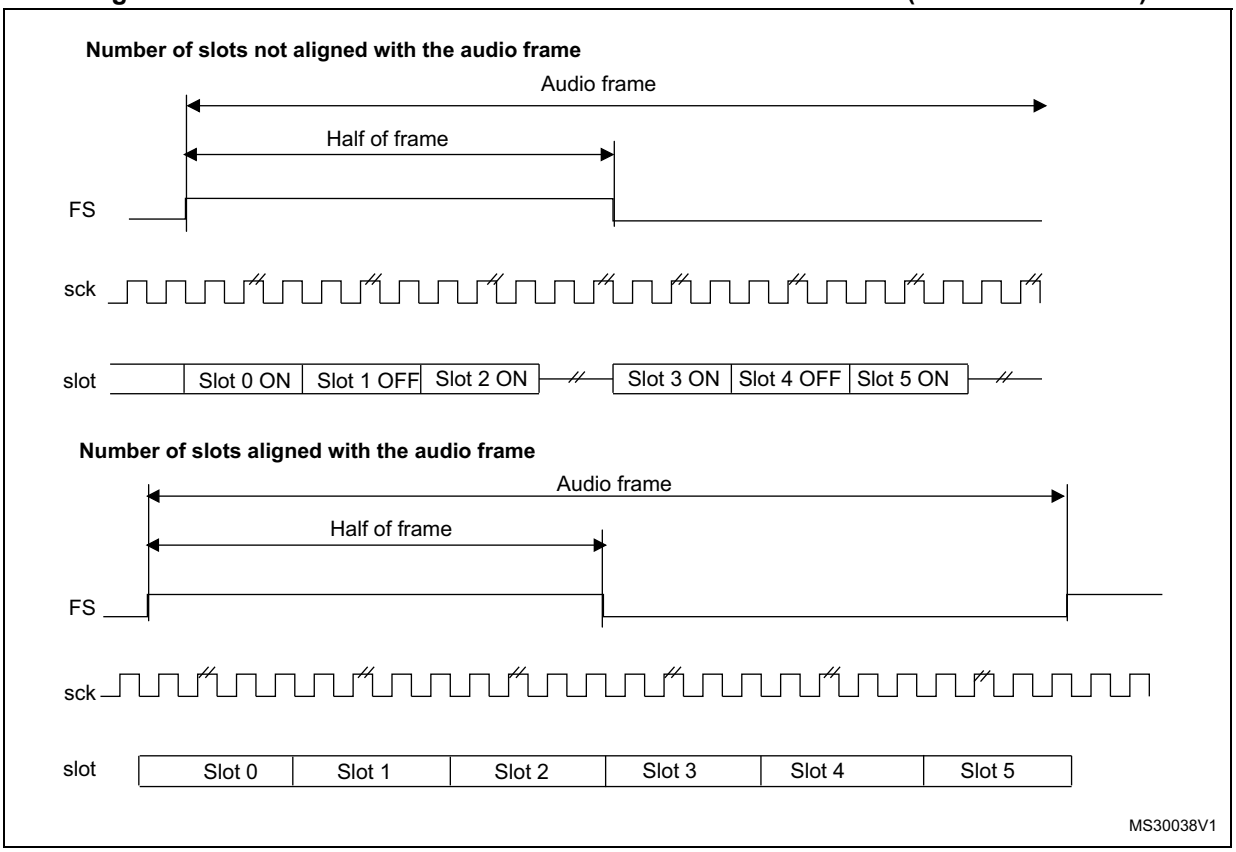
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it will have:

- 0: start of frame, like for instance the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S, the MSB or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise if TRIS = 1, the SD line is released to HI-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 644. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

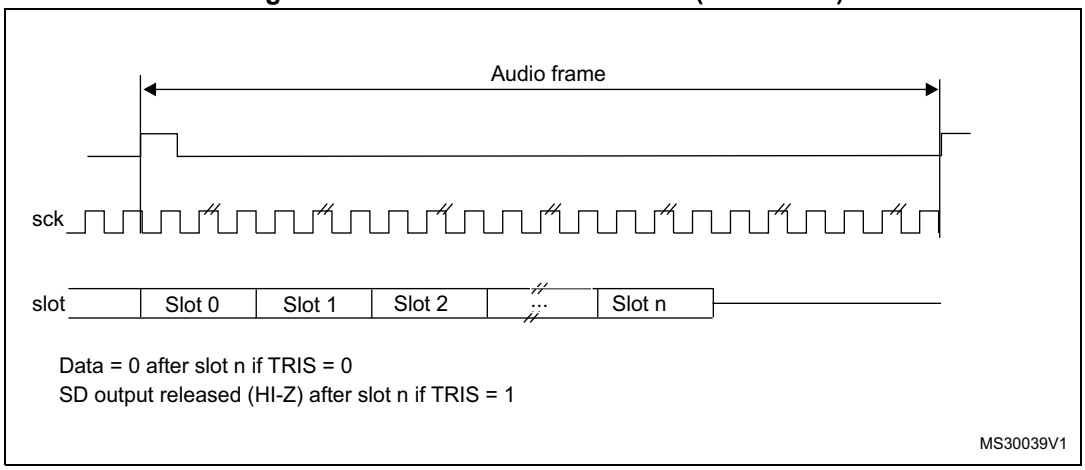


1. The frame length should be even.

If FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- if TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot will be forced to 0 until the end of frame in case of transmitter,
- if TRIS = 1, the line will be released to HI-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 645. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O will be released and left free for other purposes.

50.3.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

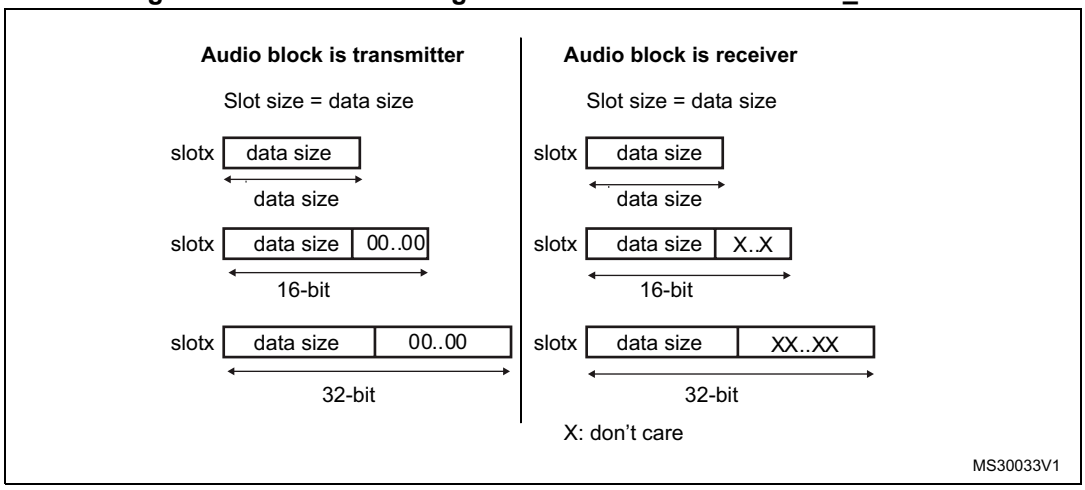
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When a invalid slot is transferred, the SD data line is either forced to 0 or released to HI-z depending on TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there will be no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

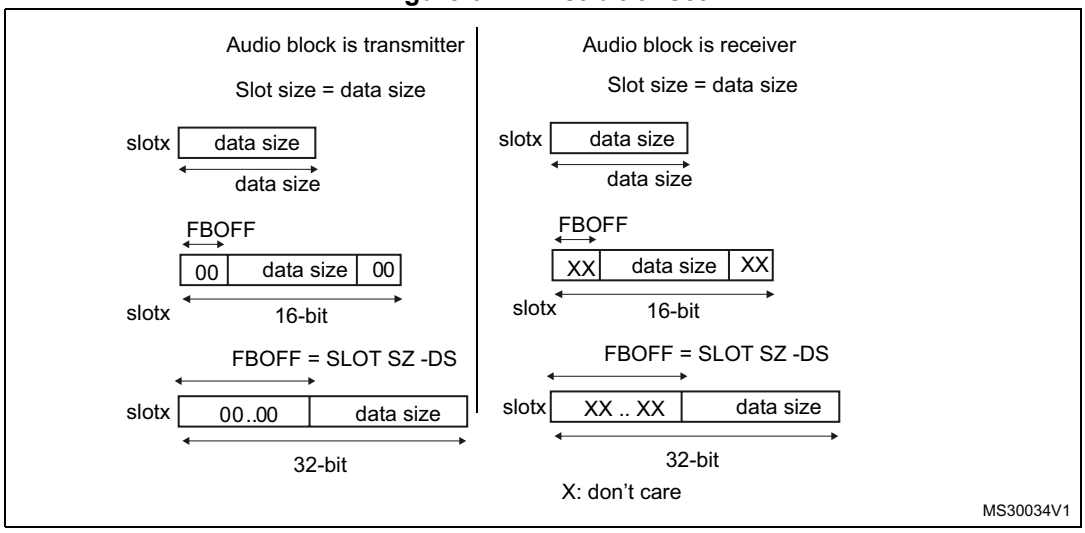
The slot size is also configurable as shown in *Figure 646*. The size of the slots is selected by setting SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 646. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values will be injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 647. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

FBOFF ≤(SLOTSZ - DS),

DS ≤SLOTSZ,

NBSLOT x SLOTSZ ≤FRL (frame length),

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 50.3.11: AC'97 link controller*.

50.3.8 SAI clock generator

Each audio sub-block has its own clock generator that makes these two blocks completely independent. There is no difference in terms of functionality between these two clock generators.

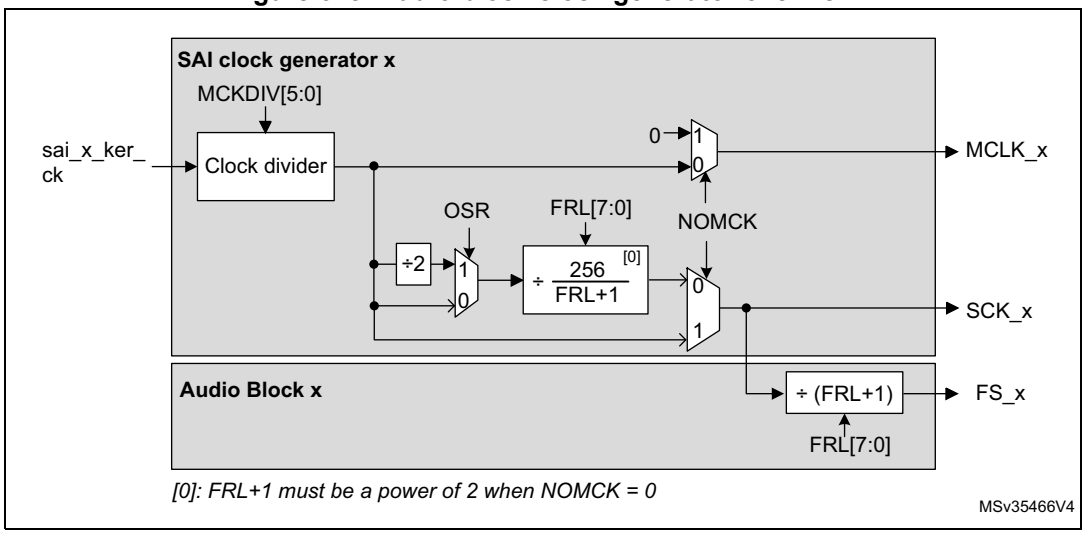
When the audio block is configured as master, the clock generator provides the bit clock (SCK_x) and the master clock (MCLK_x) for external decoders. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator. The clock source for the SAI clock generator (sai_x_ker_ck) is delivered by the product clock controller (RCC).

When the audio block is defined as slave, the clock generator is OFF. The value of NOMCK, MCKDIV and OSR bits are ignored. In addition, MCLK_x I/O pin is released and can be used as a general purpose I/O.

The bit clock strobing edge of (SCK_x) can be configured through CKSTR bit in the SAI_xCR1 register. This bit is functional in master and slave mode.

Figure 648 illustrates the architecture of the audio block clock generator.

Figure 648. Audio block clock generator overview



The NOMCK bit of the SAI_xCR1 register is used to define whether the master clock is generated or not.

When the SAI is used in master mode, the clock generator configuration differs depending on whether a master clock (MCLK_x) needs to be provided or not.

If NOMCK is set to 1, the master clock is not generated, and the user has more flexibility to select the frame length and frame synchronization frequency. In addition, MCLK_x signal is driven Low if this pin is configured as the SAI pin in GPIO peripherals. MCKDIV can still be used to adjust the SCK_x clock to the required frequency.

If NOMCK is set to 0, the master clock is generated, and can be used as reference clock for external decoders. In this case, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, and the frame length must be a power of 2. More details are given hereafter.

Clock generator programming with MCLK (NOMCK = 0)

In that case, MCLK_x frequency will be:

$$F_{MCLK_x} = 256 \times F_{FS_x} \text{ if OSR=0}$$

$$F_{MCLK_x} = 512 \times F_{FS_x} \text{ if OSR=1}$$

When MCKDIV is different from 0, MCLK_x frequency is given by the equation below:

$$F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS_x} = \frac{F_{sia_x_ker_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The frequency of the bit clock (SCK_x) is given by the following expression:

$$F_{SCK_x} = \frac{F_{MCLK_x} \times (FRL + 1)}{(OSR + 1) \times 256}$$

Note: If NOMCK = 0, (FRL+1) must be a power of two. In addition (FRL+1) must be between 8 and 256 (see Section : FS signal role).

When MCKDIV division ratio is odd, the duty cycle of MCLK will not be 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, and if OSR is equal to 0, and if (FRL+1) = 2^8.

It is recommended to configure MCKDIV to an even or big values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming without MCLK (NOMCK = 1)

When MCKDIV is different from 0, SCK_x frequency is given in the equation below:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following equation:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NOMCK = 0, (FRL+1) can take any values from 8 to 256.

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 410 shows some programming examples for 48, 96 and 192 kHz.

Table 410. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/ F_{FS}	FRL (1)	OSR	NOMCK	MCKDIV[5:0]	Audio Sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	0 or 1	192 kHz
		512	2^N-1	1	0	2	96 kHz
		512	2^N-1	1	0	4	48 kHz
		256	2^N-1	0	0	2	192 kHz
		256	2^N-1	0	0	4	96 kHz
		256	2^N-1	0	0	8	48 kHz
	N	-	63	-	1	8	192 kHz
		-	63	-	1	16	96 kHz
		-	63	-	1	32	48 kHz

1. N is an integer value between 3 and 8.

50.3.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. There is therefore only one FIFO request linked to FREQ bit in the SAI_xSR register.

An interrupt is generated if FREQIE bit is enabled in the SAI_xIM register. This depends on:

- FIFO threshold setting (FLVL bits in SAI_xCR2)
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if no data are available in SAI_xDR register (FLVL[2:0] bits in SAI_xSR is less than 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is no more empty (FLVL[2:0] bits in SAI_xSR are different from 0b000) i.e one or more data are stored in the FIFO.
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 0b010).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in

SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 011b).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 011b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if less than three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are higher or equal to 0b100).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR is less than 101b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b value).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one data is available in SAI_xDR register(FLVL[2:0] bits in SAI_xSR is higher or equal to 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in SAI_xSR is equal to 0b000) i.e no data are stored in FIFO.
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter fully (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 0b010).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half fully (FTH[2:0] set to 0b010 value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 011b).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter full(FTH[2:0] set to 011b value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b100). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations avalable(FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full(FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR is less than 101b).

Like interrupt generation, the SAI can use the DMA if DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interruption generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data should be right aligned when it is written in the SAI_xDR.

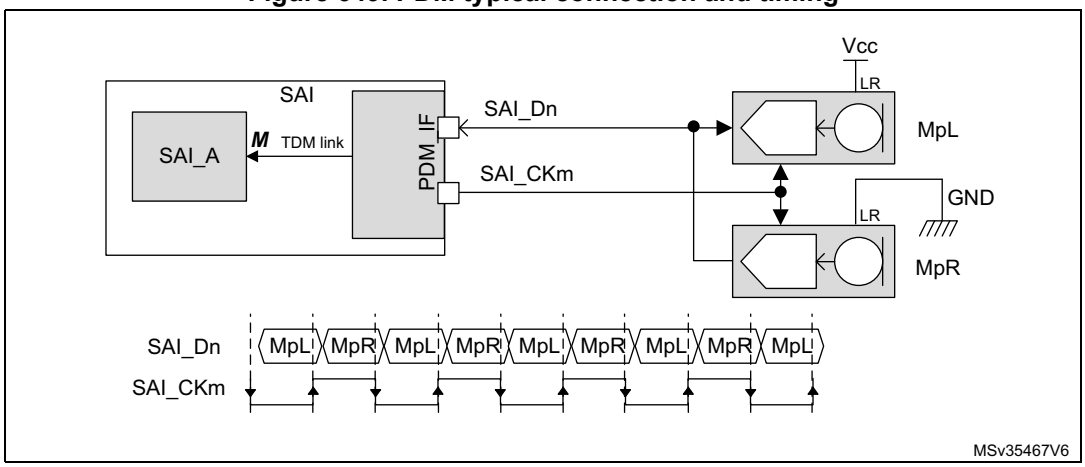
Data received will be right aligned in the SAI_xDR.

The FIFO pointers can be reinitialized when the SAI is disabled by setting bit FFLUSH in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled the data present in the FIFO will be lost automatically.

50.3.10 PDM Interface

The PDM (Pulse Density Modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. *Figure 649* shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 649. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A sub-block configured in TDM MASTER mode. It cannot be used with SAI_B sub-block. The PDM interface uses the timing signals provided by the TDM interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

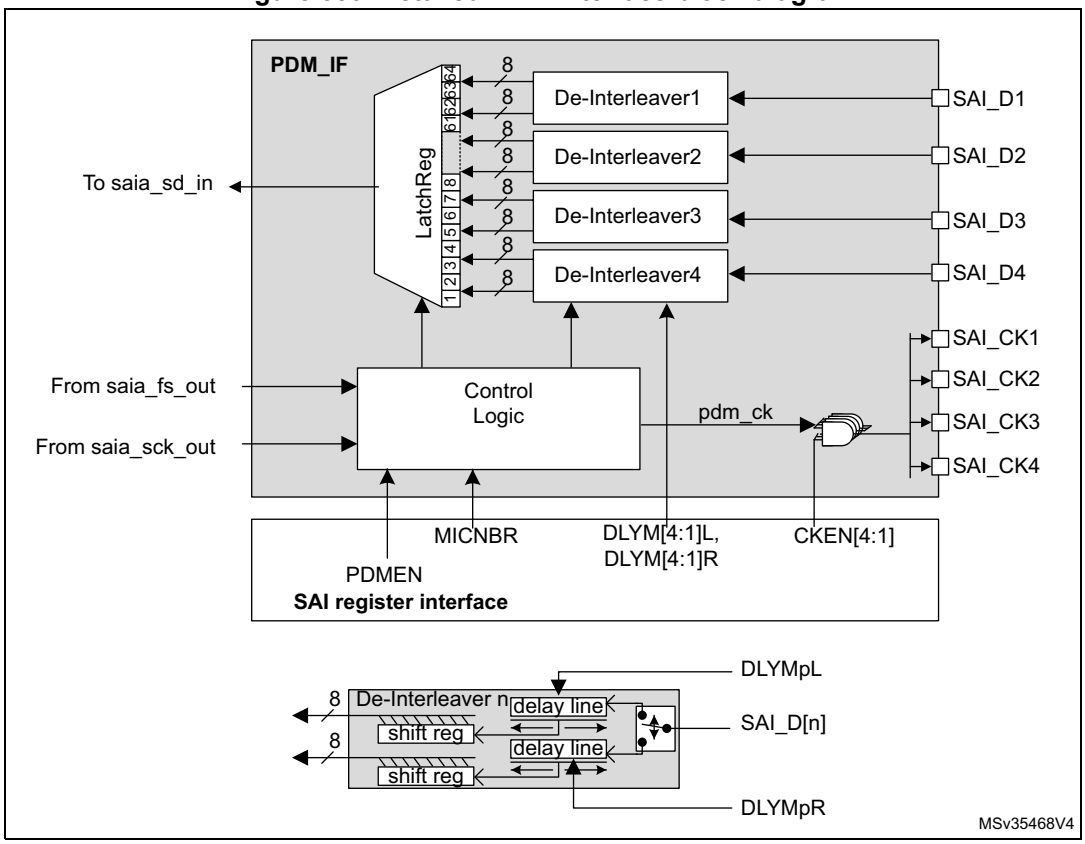
The data processing sequence into the PDM is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the TDM interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line in order to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A via the serial data line of the TDM interface.

Figure 650 hereafter shows the block diagram of PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface *does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.*

Figure 650. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM interface can be enabled through the PDMEN bit in SAI_PDMCR register. However the PDM interface must be enabled prior to enabling SAI_A block.

To reduce the memory footprint, the user can select the amount of microphones the application needs. This can be done through MICNBR[1:0] bits. It is possible to select between 2,4,6 or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

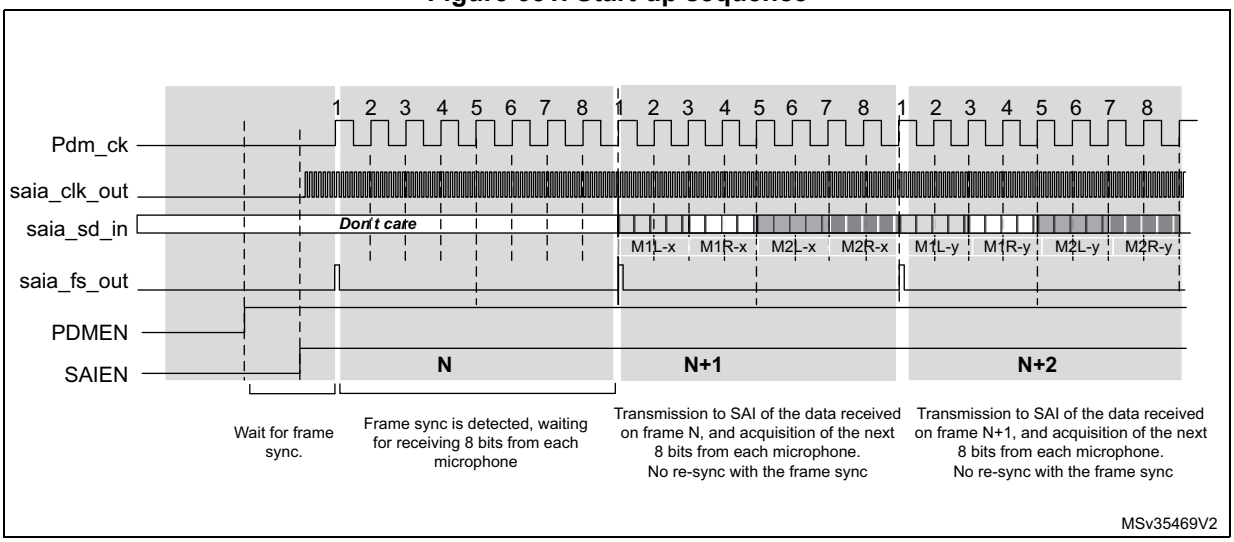
1. Configure SAI_A in TDM MASTER mode (see *Table 411*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first 2 TDMA frames received on SAI_ADR are invalid and shall be dropped.

Start-up sequence

Figure 651 shows the start-up sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.

Figure 651. Start-up sequence



SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- The amount of microphones
- The slot width selected
- LSBFIRST bit.

The slot width defines the amount of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR will contain 32 useful bits. This reduces the amount of words stored into the memory. However the

counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

In the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR will contain 8 useful bits. This increases the amount of words stored into the memory. However, it offers the advantage to avoid extra processing since each word contains information from one microphone.

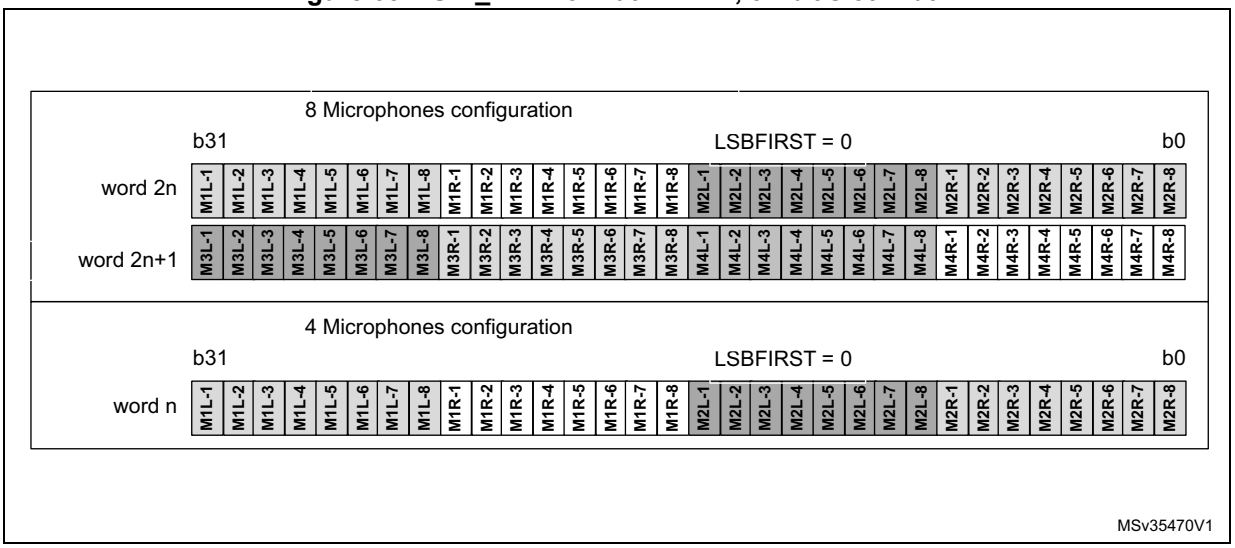
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 652*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 652. SAI_ADR format in TDM, 32-bit slot width

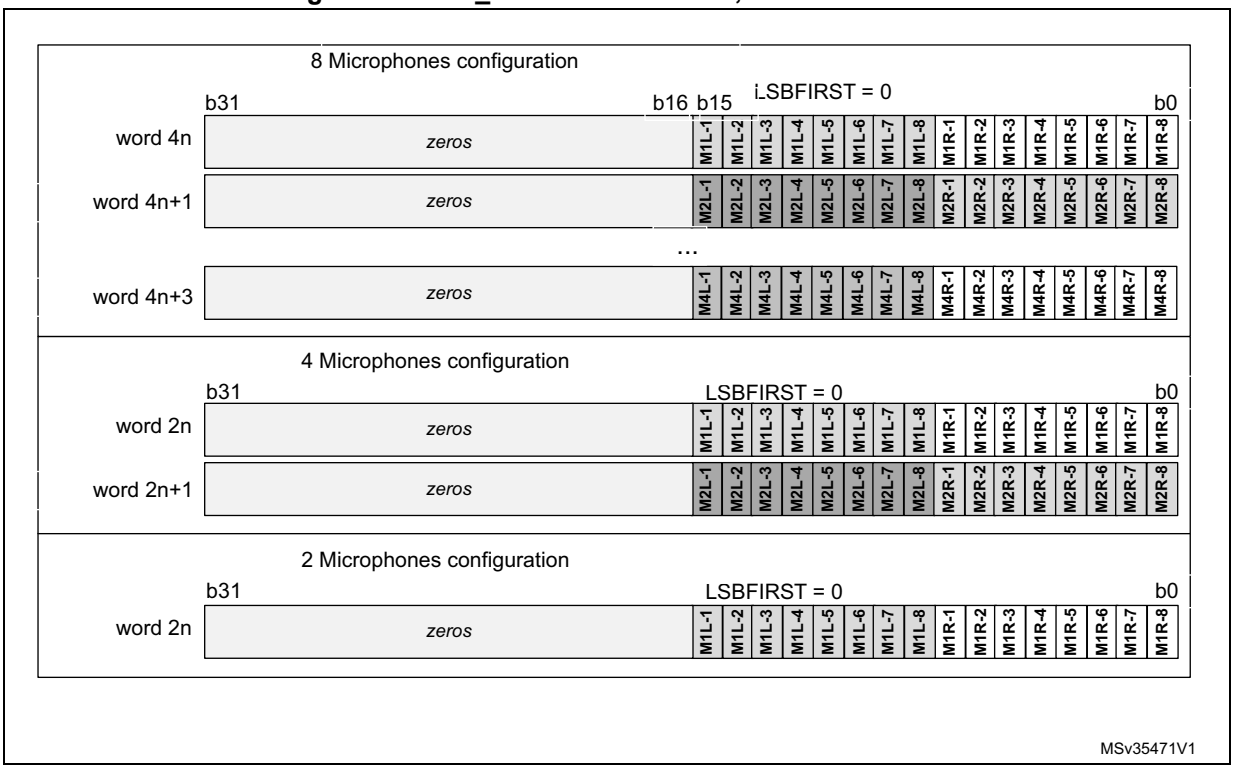


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 653*.

  For an 8 microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right aligned.

  For 4 or 2 microphone configuration, the SAI behavior is similar to 8-microphone configurations. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 653. SAI_ADR format in TDM, 16-bit slot width

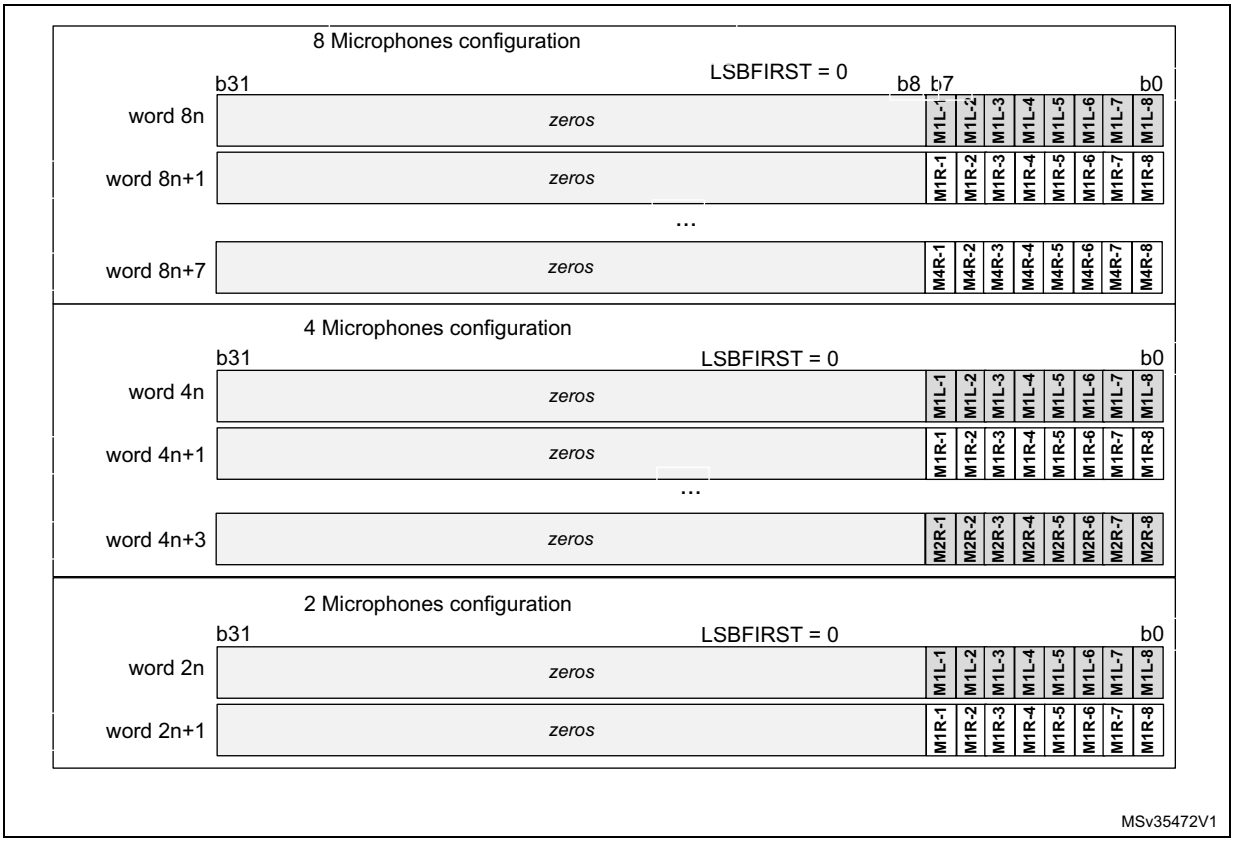


- **Using a 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 654*.

  For an 8 microphone configuration, 8 consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right aligned.

  For 4 or 2 microphone configuration, the SAI behavior is similar to 8 microphone configurations. Up to 4 words of 8 bits are required to acquire a byte from 4 microphones and 2 words from 2 microphones.

Figure 654. SAI_ADR format in TDM, 8-bit slot width



TDM configuration for PDM interface

SAI_A TDM interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the PDM interface as shown in *Table 411* to ensure a good connection with the PDM interface.

Table 411. TDM settings

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver
PRTCFG	0b00	Free protocol for TDM
DS	X	To be adjusted according to the required data format, in accordance to the frame length and the number of slots (FRL and NBSLOT). See *Table 412*.
LSBFIRST	X	This parameter can be used according to the wanted data format
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 412*.
FSALL	0	Pulse width is one bit clock cycle
FSDEF	0	FS signal is a start of frame

Table 411. TDM settings (continued)

Bit Fields	Values	Comments
FSPOL	1	FS is active High
FSOFF	0	FS is asserted on the first bit of slot 0
FBOFF	0	No offset on slot
SLOTSZ	0	Slot size = data size
NBSLOT	X	To be adjusted according to the required data format, in accordance to the slot size, and the frame length (FRL and DS). See *Table 412*.
SLOTEN	X	To be adjusted according to NBSLOT
NOMCK	1	No need to generate a master clock MCLK
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter shall be adjusted to generate the proper bitstream clock frequency. See *Table 412*.

Adjusting the bitstream clock rate

To properly program the SAI TDM interface, the user application must take into account the settings given in *Table 411*, and follow the below rules:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2$$

   MICNBR can be 0,1,2 or 3 (0 = 2 microphones., see *Section 50.5.18*)

2. Set the frame length (FRL) using the following formula

$$FRL = (16 \times (MICNBR + 1)) - 1$$

3. Configure the slot size (DS) to a multiple of (FRL+1).

Table 412. Allowed TDM frame configuration(1)

Microphone Sample rate	Nber of Mic	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 411: TDM settings* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI should be a multiple of the SCK_A frequency, and MCKDIV should be programmed accordingly.
2. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via SAI_PDMDLY register.

The new delays values will become effective after two TDM frames.

50.3.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

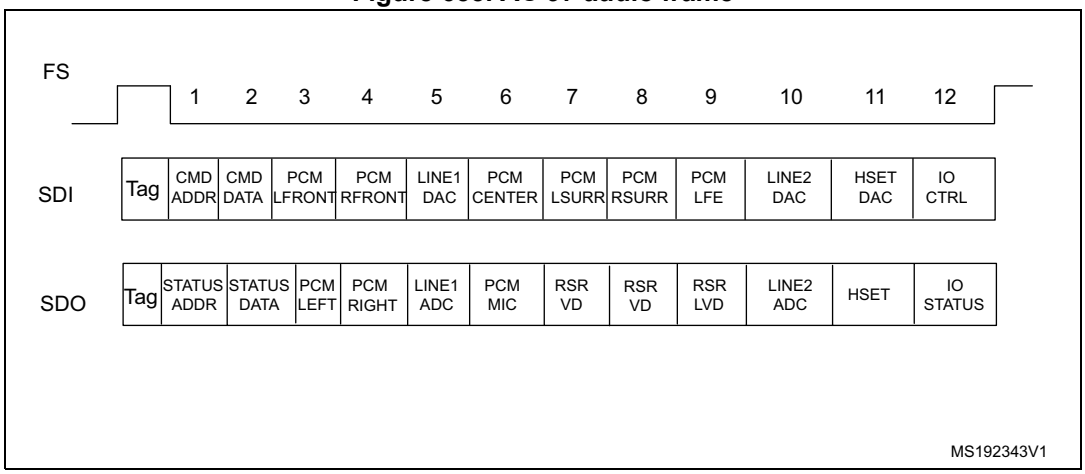
To select this protocol, set PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed to 13 slots. The first one is 16-bit wide and all the others are 20-bit wide (data slots).
- FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 655 shows an AC’97 audio frame structure.

Figure 655. AC’97 audio frame



Note: In AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

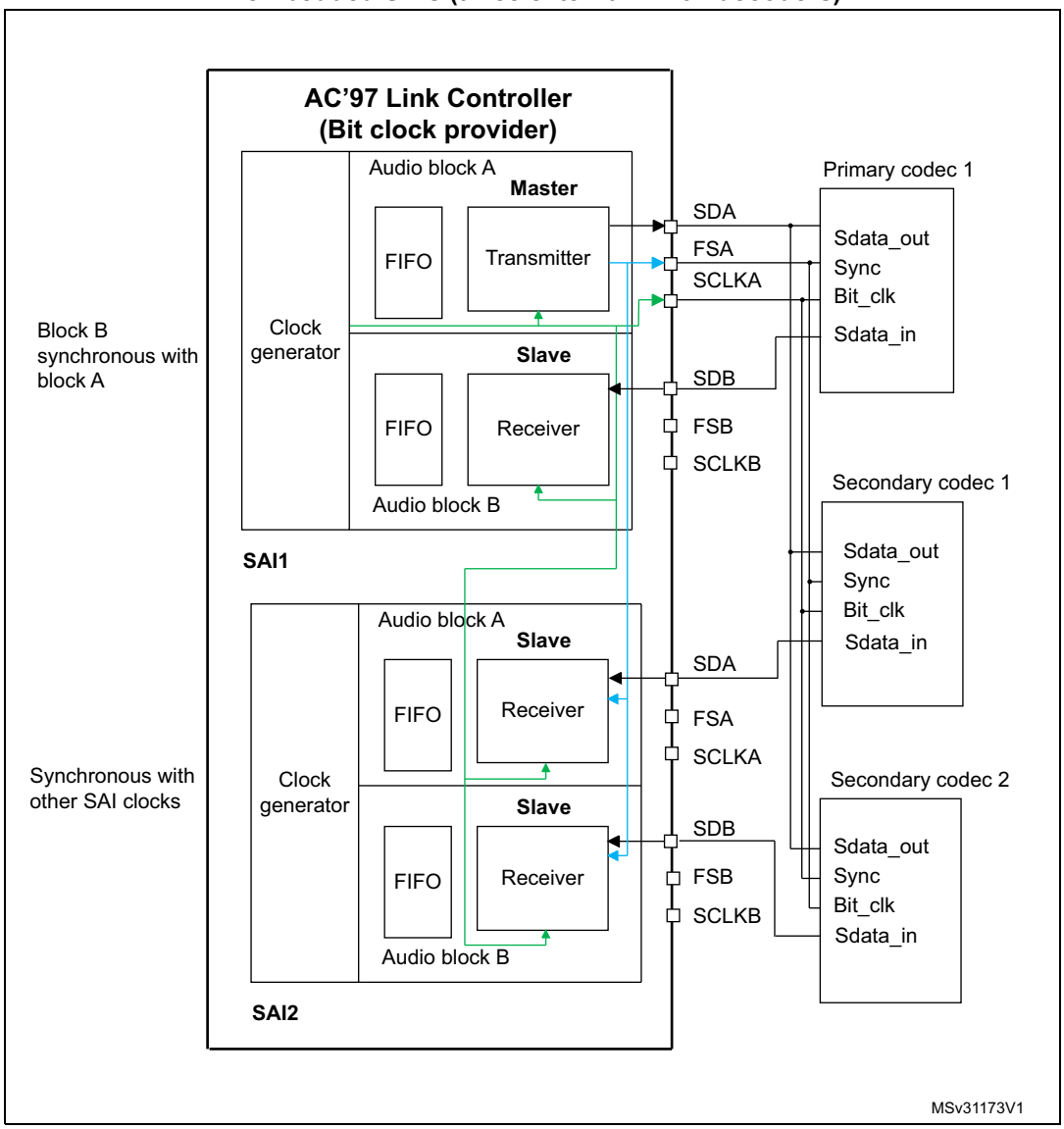
One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) allows controlling three external AC’97 decoders as illustrated in *Figure 656*.

In SAI1, the audio block A must be declared as asynchronous master transmitter whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.

Figure 656. Example of typical AC'97 configuration on devices featuring at least 2 embedded SAIs (three external AC'97 decoders)



In receiver mode, the SAI acting as an AC'97 link controller requires no FIFO request and so no data storage in the FIFO when the Codec ready bit in the slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY will be set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC'97 protocol.

Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency shall be set to 48 kHz. The formulas given in *Section 50.3.8: SAI clock generator* shall be used with FRL = 255, in order to generate the proper frame rate (F_{FS_x}).

50.3.12 SPDIF output

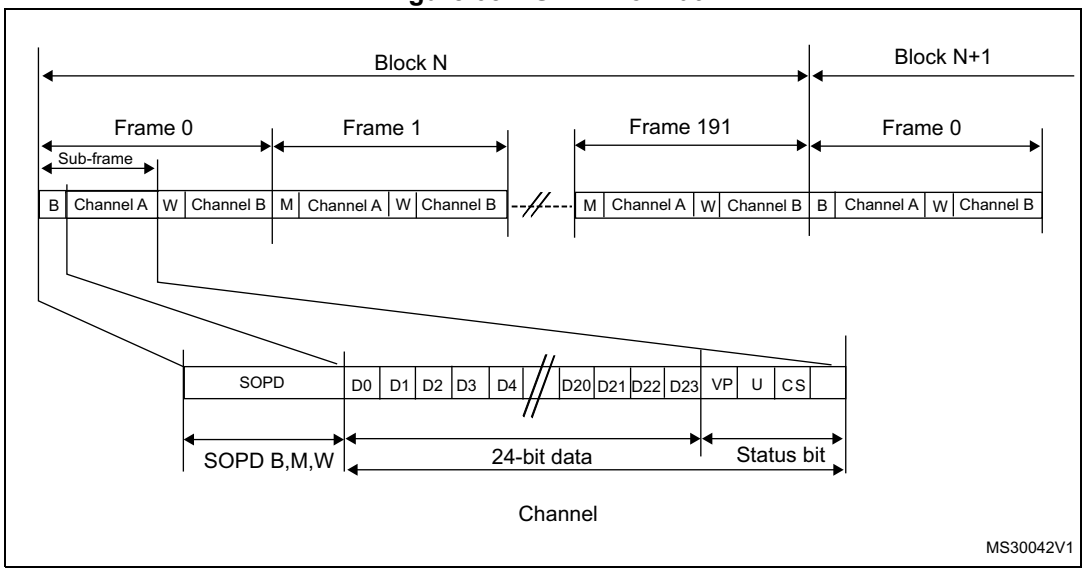
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set PRTCFG[1:0] bit to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only SD data line is enabled.
- FS, SCK, MCLK I/Os pins are left free.
- MODE[1] bit is forced to 0 to select the master mode in order to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock should be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 657*.

Figure 657. SPDIF format



A SPDIF block contains 192 frames. Each frame is composed of two 32-bit sub-frames, generally one for the left channel and one for the right channel. Each sub-frame is composed of a SOPD pattern (4-bit) to specify if the sub-frame is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 413*). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.

Table 413. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

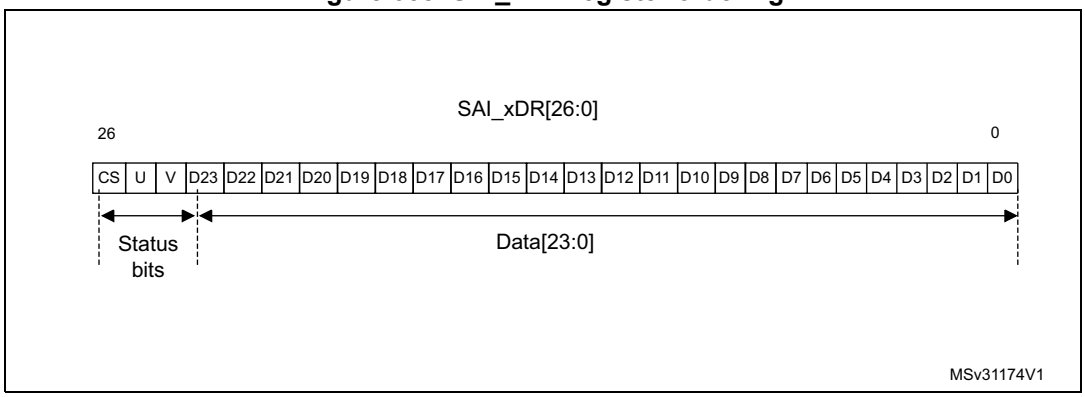
- SAI_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, then data shall be mapped on SAI_xDR[23:4].

If the data size is 16 bits, then data shall be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 658. SAI_xDR register ordering



Note: The transfer is performed always with LSB first.

The SAI first sends the adequate preamble for each sub-frame in a block. The SAI_xDR is then sent on the SD line (manchester coded). The SAI ends the sub-frame by transferring the Parity bit calculated as described in *Table 414*.

Table 414. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence should be

executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to a start of new block (data for preamble B). If the DMA is used, the DMA source base address pointer should be updated accordingly.
5. Enable again the DMA stream (DMA peripheral) if the DMA used to manage data transfers according to the new source base address.
6. Enable again the SAI by setting SAIEN bit in SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI shall provide a bit clock twice faster as the symbol-rate. The table hereafter shows usual examples of symbol rates with respect to the audio sampling rate.

Table 415. Audio sampling frequency versus symbol rates

Audio Sampling Frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_X}) is given by the formula:

$$F_S = \frac{F_{SCK_x}}{128}$$

And the bit clock rate is obtained as follow:

$$F_{SCK_x} = \frac{F_{SAI_CK_x}}{MCKDIV}$$

50.3.13 Specific features

The SAI interface embeds specific features which can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio sub-block is a transmitter or a receiver.

Audio sub-block in transmission mode

In transmitter mode, the mute mode can be selected at anytime. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is set during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame will still be a mute frame.

If the number of slots set through NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, MUTEVAL bit in the SAI_xCR2 is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio sub-block in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio sub-block is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed, without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO will be reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode,

1. Set MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2 as in transmitter mode. When it is set, only slot 0 data will be stored in the FIFO. The data belonging to slot 1 will be discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require to process the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when TDM mode is selected), the application software can choose to process or not the data before sending it on SD serial output line (compression) or to expand the data after the reception on SD serial input line (expansion) as illustrated in *Figure 659*. The two companding modes supported are the µ-Law and the A-Law log which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

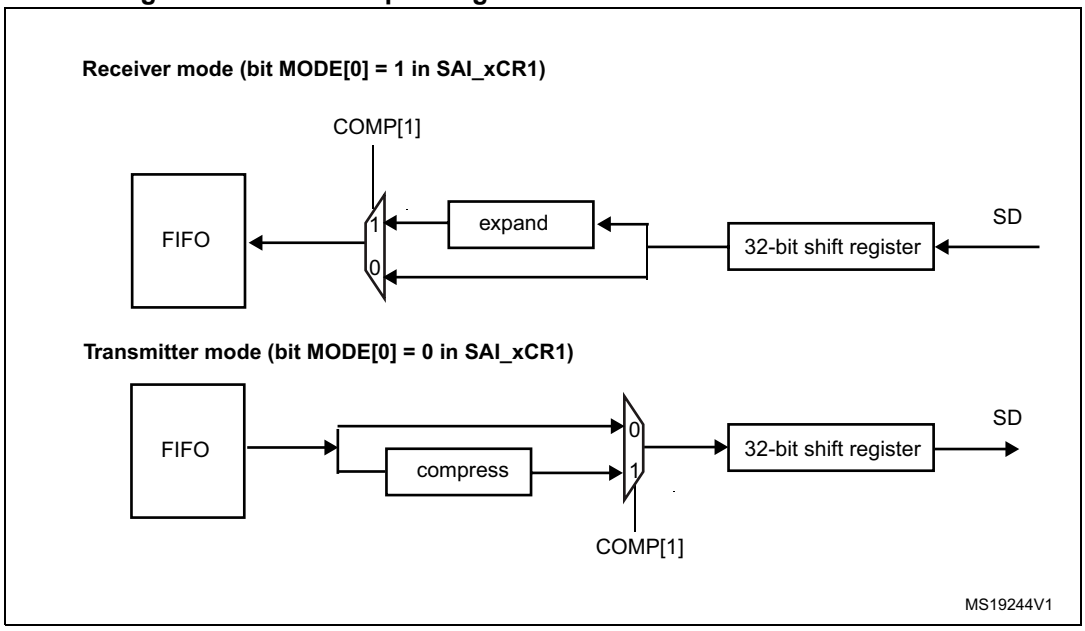
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law or A-Law companding standard can be computed based on 1's complement or 2's complement representation depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8-bit wide. For this reason, DS[2:0] bits in the SAI_xCR1 register will be forced to 010 when the SAI audio block is enabled (SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes selected through the COMP[1:0] bits.

If no companding processing is required, COMP[1:0] bits should be kept clear.

Figure 659. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through the SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode will be applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm will be applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via TRIS bit).

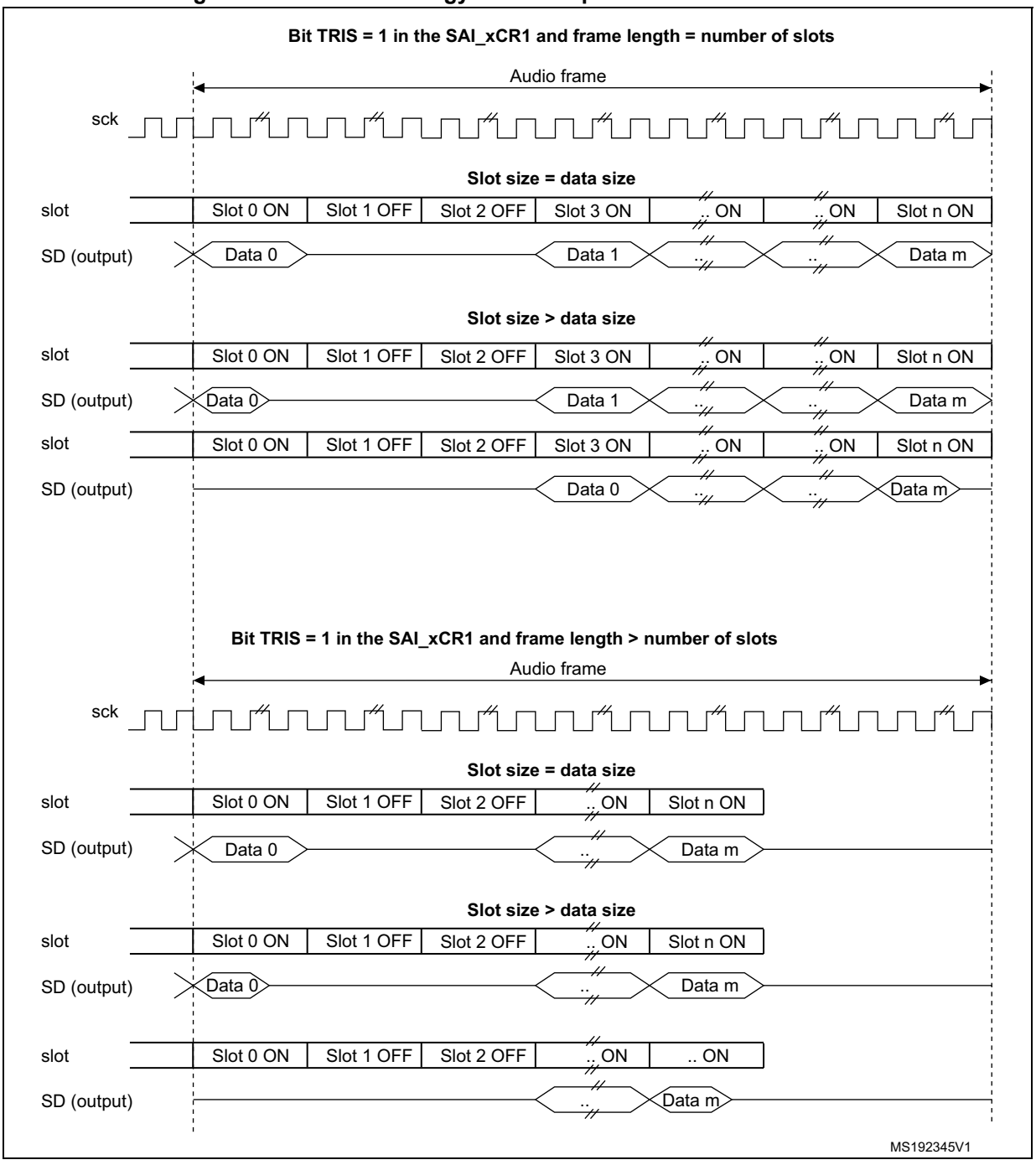
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which could result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin will then be tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line will be tri-stated when the padding to 0 is done to complete the audio frame.

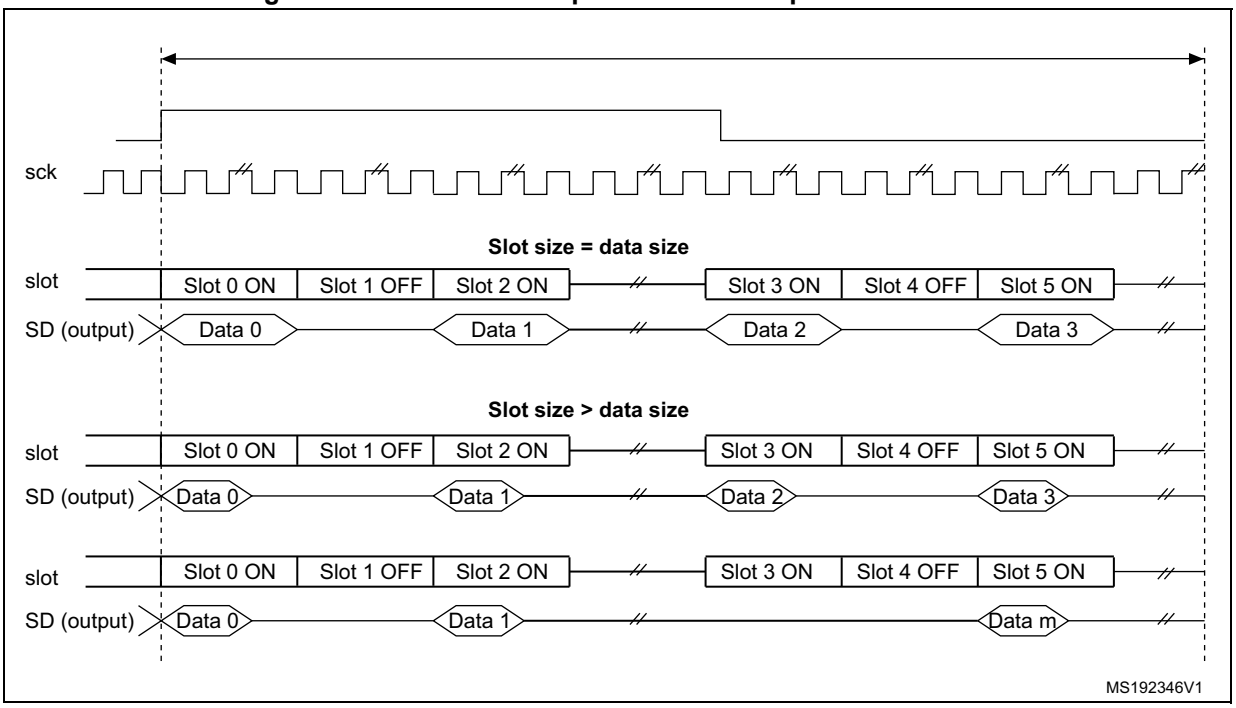
Figure 660 illustrates these behaviors.

Figure 660. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 661* (where bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 661. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the High impedance states on the SD output line on *Figure 660* and *Figure 661* are replaced by a drive with a value of 0.

50.3.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun
- Anticipated frame synchronization detection
- Late frame synchronization detection
- Codec not ready (AC'97 exclusively)
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

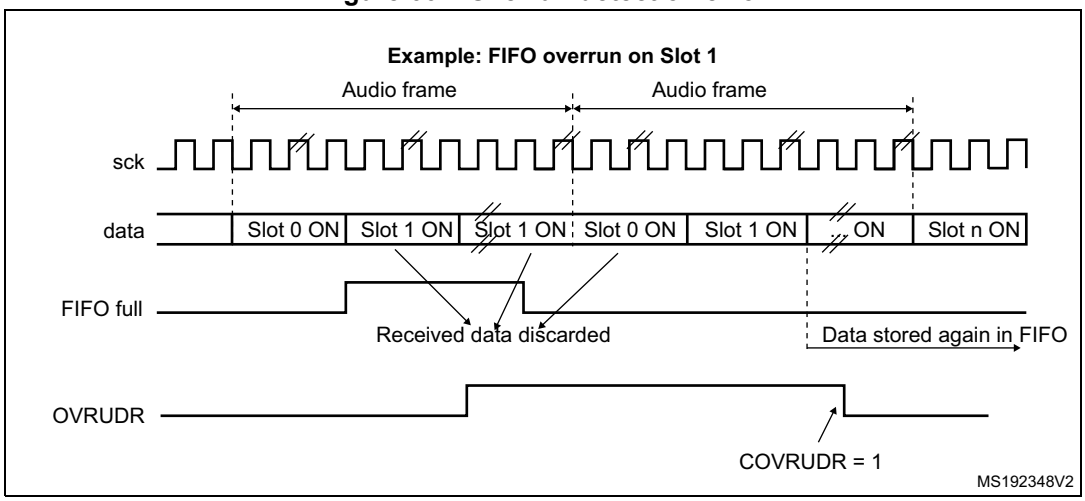
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the flag OVRUDR in the SAI_xSR register is set and an interrupt is generated if OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data will be stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver will store new data (from new audio frame) from the slot number which was stored internally when the overrun condition was detected. This avoids data slot de-alignment in the destination memory (refer to *Figure 662*).

The OVRUDR flag is cleared when COVRUDR bit is set in the SAI_xCLRFR register.

Figure 662. Overrun detection error



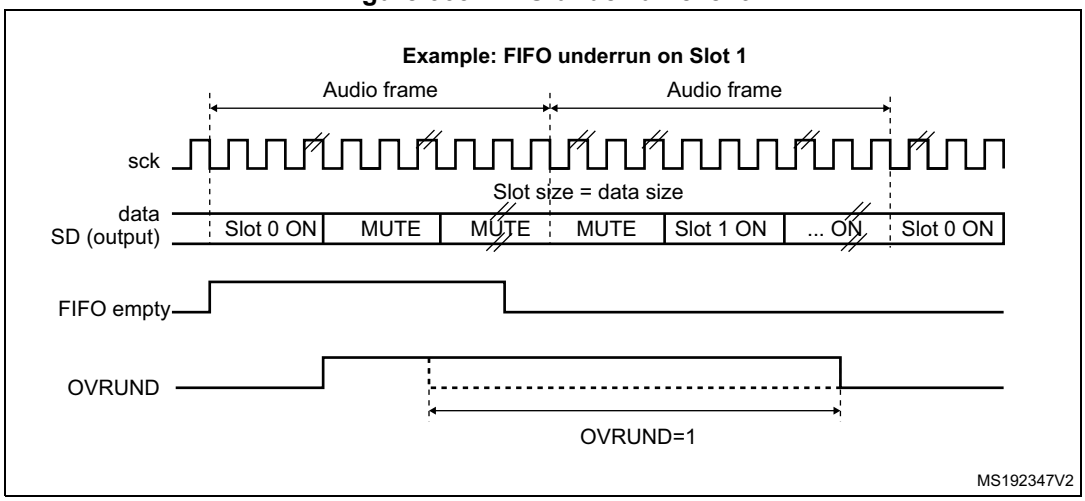
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 663*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio sub-block is configured as master or slave.

Figure 663. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting SAIEN bit in SAI_xCR1 register. To make sure the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via FFLUS bit in SAI_xCR2 register.
3. Enable again the SAI peripheral (SAIEN bit set to 1).
4. The SAI block will wait for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity and the frame offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expecting time thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which should not be the case in most applications. In this case, the LFSDET flag will be set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the Codec is not ready to communicate during the reception of the TAG 0 (slot0) of the AC'97 audio frame. In this case, no data will be automatically stored into the FIFO since the Codec is not ready, until the TAG 0 indicates that the Codec is ready. All the active slots defined in the SAI_xSLOTR register will be captured when the Codec is ready.

To clear CNRDY flag, CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NOMCK = 0)

When the audio block operates as a master (MODE[1] = 0) and NOMCK bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

MODE, NOMCK, and SAIEN bits belong to SAI_xCR1 register and FRL to SAI_xFRCR register.

If WCKCFGIE bit is set, an interrupt is generated when WCKCFG flag is set in the SAI_xSR register. To clear this flag, set CWCKCFG bit in the SAI_xCLRFR register.

When WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of SAIEN bit.

50.3.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI is stops working. SAIEN bit remains High until the SAI is completely switched-off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one which is the master must be disabled first.

50.3.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio sub-block supporting basic DMA request/acknowledge protocol.

To configure the audio sub-block for DMA transfer, set DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 50.3.9: Internal FIFOs*). DMA transfer direction is linked to the SAI audio sub-block configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block is operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure SAI and FIFO threshold levels to specify when the DMA request will be launched.
2. Configure SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

Note: Before configuring the SAI block, the SAI DMA channel must be disabled.

50.4 SAI interrupts

The SAI supports 7 interrupt sources as shown in *Table 416*.

Table 416. SAI interrupt sources

Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 50.3.9: Internal FIFOs*
OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
WCKCFG	ERROR	Master with NOMCK = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

50.5 SAI registers

50.5.1 Global configuration register (SAI_GCR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs
These bits are set and cleared by software.
00: No synchronization output signals. SYNCOUT[1:0] should be configured as “No synchronization output signals” when audio block is configured as SPDIF
01: Block A used for further synchronization for others SAI
10: Block B used for further synchronization for others SAI
11: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs
These bits are set and cleared by software.
Refer to for information on how to program this field.
These bits must be set when both audio blocks (A and B) are disabled.
They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

50.5.2 Configuration register 1 (SAI_ACR1)

Address offset: 0x004

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	OSR	MCKDIV[5:0]						NOMCK	Res.	DMAEN	SAIEN
					rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **OSR:** Oversampling ratio for master clock
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 50.3.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NOMCK:** No divider
This bit is set and cleared by software.
0: Master clock generator is enabled
1: Master clock generator is disabled. The clock divider controlled by MCKDIV can still be used to generate the bit clock.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command will not be taken into account.
This bit allows controlling the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio sub-block is disabled.

00: audio sub-block in asynchronous mode.

01: audio sub-block is synchronous with the other internal audio sub-block. In this case, the audio sub-block must be configured in slave mode

10: audio sub-block is synchronous with an external SAI embedded peripheral. In this case the audio sub-block should be configured in Slave mode.

11: Reserved

Note: The audio sub-block should be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol allows to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

50.5.3 Configuration register 1 (SAI_BCR1)

Address offset: 0x024

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	OSR	MCKDIV[5:0]						NOMCK	Res.	DMAEN	SAIEN
					rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **OSR:** Oversampling ratio for master clock

0: Master clock frequency = F_{FS} x 256

1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider

These bits are set and cleared by software.

000000: Divides by 1 the kernel clock input (sai_x_ker_ck).

Otherwise, The master clock frequency is calculated according to the formula given in *Section 50.3.8: SAI clock generator*.

These bits have no meaning when the audio block is slave.

They have to be configured when the audio block is disabled.

Bit 19 **NOMCK:** No divider

This bit is set and cleared by software.

0: Master clock generator is enabled

1: Master clock generator is disabled. The clock divider controlled by MCKDIV can still be used to generate the bit clock.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable

This bit is set and cleared by software.

0: DMA disabled

1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command will not be taken into account.

This bit allows controlling the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled.

Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio sub-block is disabled.

00: audio sub-block in asynchronous mode.

01: audio sub-block is synchronous with the other internal audio sub-block. In this case, the audio sub-block must be configured in slave mode

10: audio sub-block is synchronous with an external SAI embedded peripheral. In this case the audio sub-block should be configured in Slave mode.

11: Reserved

Note: The audio sub-block should be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol allows to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

50.5.4 Configuration register 2 (SAI_ACR2)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that will be used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when TDM is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1’s complement representation.

1: 2’s complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET will be set and an interrupt will be generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software.It must be written before enabling the audio block: SAIEN. This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interruption must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

50.5.5 Configuration register 2 (SAI_BCR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that will be used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when TDM is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1’s complement representation.

1: 2’s complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET will be set and an interrupt will be generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software.It must be written before enabling the audio block: SAIEN. This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and should not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It should be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit should be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interruption must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

50.5.6 Frame configuration register (SAI_AFRCR)

Address offset: 0x00C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.

This bit must be configured when the audio block is disabled.

0: FS is active low (falling edge)

1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.

0: FS signal is a start frame signal

1: FS signal is a start of frame signal + channel side identification

When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots will be dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).

This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block will behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length should be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NOMCK = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

50.5.7 Frame configuration register (SAI_BFRCR)

Address offset: 0x02C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.

This bit must be configured when the audio block is disabled.

0: FS is active low (falling edge)

1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots will be dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block will behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length should be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NOMCK = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

50.5.8 Slot register (SAI_ASLOTR)

Address offset: 0x010

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots should be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI will be undetermined.
Refer to *Section : Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

50.5.9 Slot register (SAI_BSLOTR)

Address offset: 0x030

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots should be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI will be undetermined.
Refer to *Section : Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

50.5.10 Interrupt mask register (SAI_AIM)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET IE	AFSDET IE	CNRDY IE	FREQ IE	WCKCFG IE	MUTEDET IE	OVRUDR IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt will be generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt will be generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interruption i generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interruption in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NOMCK = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in TDM mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

50.5.11 Interrupt mask register (SAI_BIM)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt will be generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt will be generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interruption i generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interruption in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NOMCK = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in TDM mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

50.5.12 Status register (SAI_ASR)

Address offset: 0x018

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
If the SAI block is configured as transmitter:
000: FIFO empty
001: FIFO <= ¼ but not empty
010: ¼ < FIFO <= ½
011: ½ < FIFO <= ¾
100: ¾ < FIFO but not full
101: FIFO full
If SAI block is configured as receiver:
000: FIFO empty
001: FIFO < ¼ but not empty
010: ¼ <= FIFO < ½
011: ½ =< FIFO < ¾
100: ¾ =< FIFO but not full
101: FIFO full

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.

0: No FIFO request.

1: FIFO request to read or to write the SAI_xDR.

The request depends on the audio block configuration:

– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.

– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.

This bit is read only.

0: Clock configuration is correct

1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 50.3.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)

This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NOMCK = 0.

It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.

This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

50.5.13 Status register (SAI_BSR)

Address offset: 0x038

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
If the SAI block is configured as transmitter:
000: FIFO empty
001: FIFO <= ¼ but not empty
010: ¼ < FIFO <= ½
011: ½ < FIFO <= ¾
100: ¾ < FIFO but not full
101: FIFO full
If SAI block is configured as receiver:
000: FIFO empty
001: FIFO < ¼ but not empty
010: ¼ <= FIFO < ½
011: ½ =< FIFO < ¾
100: ¾ =< FIFO but not full
101: FIFO full

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.

0: No FIFO request.

1: FIFO request to read or to write the SAI_xDR.

The request depends on the audio block configuration:

– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.

– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.

This bit is read only.

0: Clock configuration is correct

1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 50.3.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)

This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NOMCK = 0.

It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.

This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

50.5.14 Clear flag register (SAI_ACLRFR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NOMCK = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

50.5.15 Clear flag register (SAI_BCLRFR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NOMCK = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

50.5.16 Data register (SAI_ADR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

50.5.17 Data register (SAI_BDR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

50.5.18 PDM control register (SAI_PDMCR)

Address offset: 0x0044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
				rw	rw	rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **CKEN4**: Clock enable of bitstream clock number 4
This bit is set and cleared by software.
0: SAI_CK4 clock disabled
1: SAI_CK4 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.

Bit 10 **CKEN3**: Clock enable of bitstream clock number 3
This bit is set and cleared by software.
0: SAI_CK3 clock disabled
1: SAI_CK3 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2
This bit is set and cleared by software.
0: SAI_CK2 clock disabled
1: SAI_CK2 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1
This bit is set and cleared by software.
0: SAI_CK1 clock disabled
1: SAI_CK1 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones
This bit is set and cleared by software.
00: Configuration with 2 microphones
01: Configuration with 4 microphones
10: Configuration with 6 microphones
11: Configuration with 8 microphones
Note: It is not recommended to configure this field when PDMEN = 1.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable
This bit is set and cleared by software. This bit allows to control the state of the PDM interface block. Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.
0: PDM interface disabled
1: PDM interface enabled

50.5.19 PDM delay register (SAI_PDMDLY)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 of T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bit is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This field can be changed on-the-fly.0

50.5.20 SAI register map

The following table summarizes the SAI registers.

Table 417. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	SAI_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
	Reset value																											0	0			0	0
0x0004 or 0x0024	SAI_xCR1	Res.	Res.	Res.	Res.	Res.	OSR	MCKDIV[3:0]						NOMCK	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]		Res.		PRTCFG[1:0]		MODE[1:0]	
	Reset value						0	0	0	0	0	0	0	0	-	0	0			0	0	0	0	0	0	0	1	0		0	0	0	0
0x0008 or 0x0028	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]							MUTE VAL	MUTE	TRIS	FFLUS	FTH	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C or 0x002C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0010 or 0x0030	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0
0x0014 or 0x0034	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDETIE	CNRDYIE	FREQIE	WCKCFG	MUTEDET	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x0018 or 0x0038	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	0	1	0	0
0x001C or 0x003C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	CAFSDET	CNRDY	Res.	WCKCFG	MUTEDET	OVRUDR
	Reset value																										0	0	0		0	0	0
0x0020 or 0x0040	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 417. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0044	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																					0	0	0	0			0	0				0
0x0048	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0	-	0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0	

Refer to *Section 2.3 on page 124* for the register boundary addresses.

51 SPDIF receiver interface (SPDIFRX)

51.1 SPDIFRX interface introduction

The SPDIFRX interface handles S/PDIF audio protocol.

51.2 SPDIFRX main features

- Up to 4 inputs available
- Automatic symbol rate detection
- Maximum symbol rate: 12.288 MHz
- Stereo stream from 8 to 192 kHz supported
- Supports Audio IEC-60958 and IEC-61937, consumer applications
- SOPDs B, M and W insertion inside S/PDIF flow
- Parity bit management
- Communication using DMA for audio samples
- Communication using DMA for control and user channel information
- Interrupt capabilities

51.3 SPDIFRX functional description

The SPDIFRX peripheral, is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. These standards support simple stereo streams up to high sample rate, and compressed multi-channel surround sound, such as those defined by Dolby or DTS.

The receiver provides all the necessary features to detect the symbol rate, and decode the incoming data. It is possible to use a dedicated path for the user and channel information in order to ease the interface handling. *Figure 664* shows a simplified block diagram.

The SPDIFRX_DC block is responsible of the decoding of the S/PDIF stream received from SPDIFRX_IN[4:1] inputs. This block re-sample the incoming signal, decode the manchester stream, recognize frames, sub-frames and blocks elements. It delivers to the REG_IF part, decoded data, and associated status flags.

This peripheral can be fully controlled via the APB1 bus, and can handle two DMA channels:

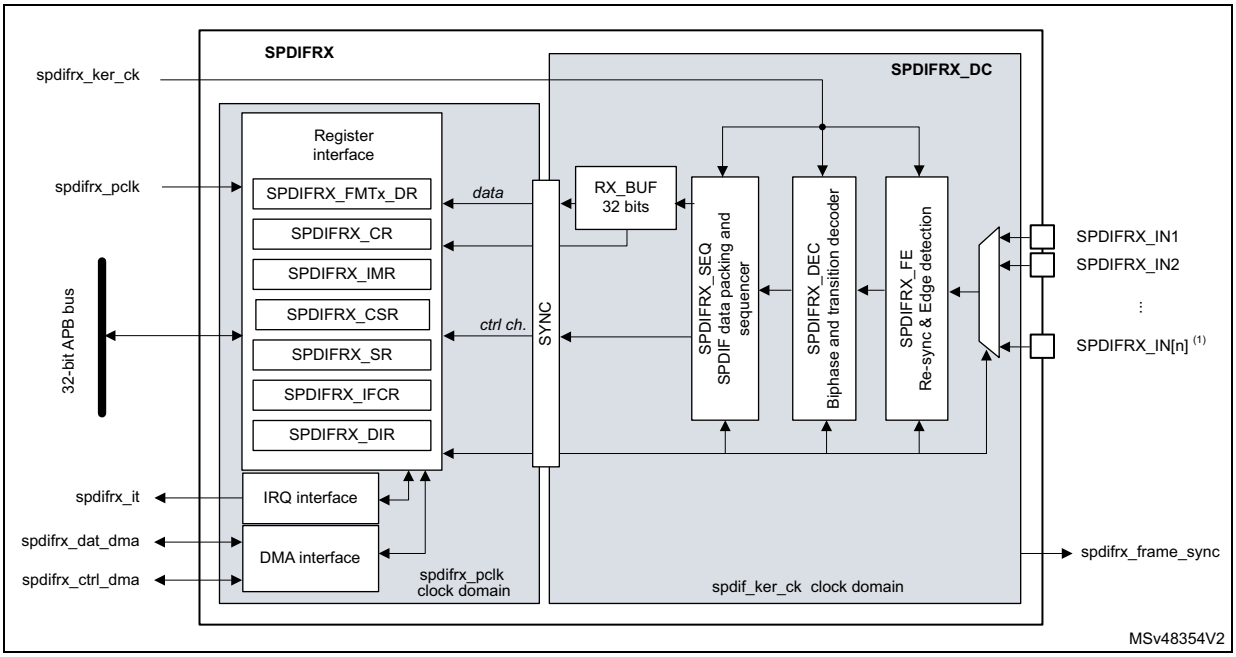
- A DMA channel dedicated to the transfer of audio samples
- A DMA channel dedicated to the transfer of IEC60958 channel status and user information

Interrupt services are also available either as an alternative function to the DMA, or for signaling error or key status of the peripheral.

The SPDIFRX also offers a signal named **spdifrx_frame_sync**, which toggles every time that a sub-frame's preamble is detected. So the duty cycle will be 50%, and the frequency equal to the frame rate.

This signal can be connected to timer events, in order to compute frequency drift.

Figure 664. SPDIFRX block diagram



1. 'n' is fixed to 4.

51.3.1 SPDIFRX pins and internal signals

Table 418 lists the SPDIFRX internal input/output signals, *Table 419* the SPDIFRX pins (alternate functions).

Table 418. SPDIFRX internal input/output signals

Signal name	Signal type	Description
spdifrx_ker_ck	Digital input	SPDIFRX kernel clock
spdifrx_pclk	Digital input	SPDIFRX register interface clock
spdifrx_it	Digital output	SPDIFRX global interrupt
spdifrx_dat_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for data transfer
spdifrx_ctrl_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for channel status and user information transfer
spdifrx_frame_sync	Digital output	SPDIFRX frame rate synchronization signal

Table 419. SPDIFRX pins

Signal name	Signal type	Description
SPDIFRX_IN1	Digital input	Input 1 for S/PDIF signal
SPDIFRX_IN2	Digital input	Input 2 for S/PDIF signal
SPDIFRX_IN3	Digital input	Input 3 for S/PDIF signal
SPDIFRX_IN4	Digital input	Input 4 for S/PDIF signal

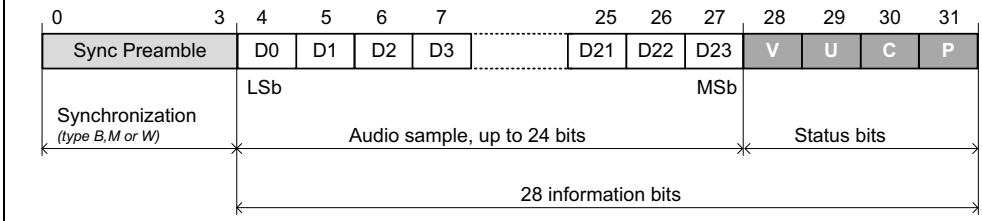
51.3.2 S/PDIF protocol (IEC-60958)

S/PDIF block

A S/PDIF frame is composed of two sub-frames (see *Figure 666*). Each sub-frame contains 32 bits (or time slots):

- Bits 0 to 3 carry one of the synchronization preambles
- Bits 4 to 27 carry the audio sample word in linear 2's complement representation. The most significant bit (MSB) is carried by bit 27. When a 20-bit coding range is used, bits 8 to 27 carry the audio sample word with the LSB in bit 8.
- Bit 28 (validity bit “V”) indicates if the data is valid (for converting it to analog for example)
- Bit 29 (user data bit “U”) carries the user data information like the number of tracks of a Compact Disk.
- Bit 30 (channel status bit “C”) carries the channel status information like sample rate and protection against copy.
- Bit 31 (parity bit “P”) carries a parity bit such that bits 4 to 31 inclusive carry an even number of ones and an even number of zeroes (even parity).

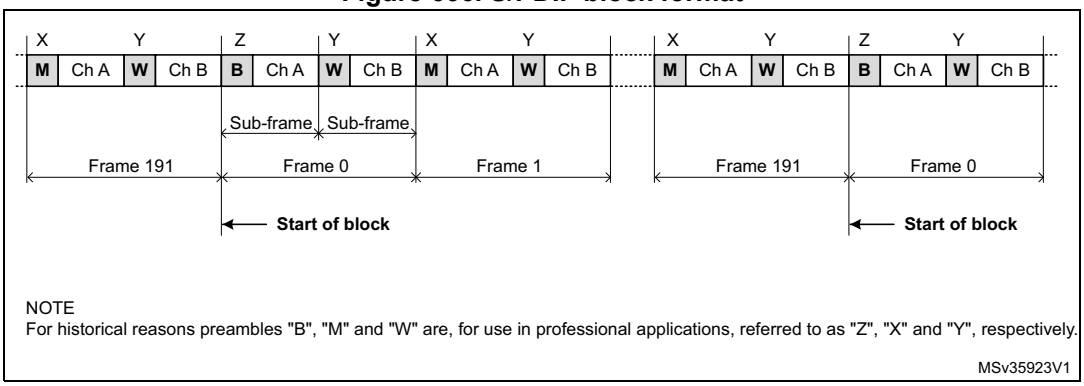
Figure 665. S/PDIF Sub-Frame Format



For linear coded audio applications, the first sub-frame (left or “A” channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble “M”. However, the preamble changes to preamble “B” once every 192 frames to identify the start of the block structure used to organize the channel status and user information. The second sub-frame (right or “B” channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble “W”.

A S/PDIF block contains 192 pairs of sub-frames of 32 bits.

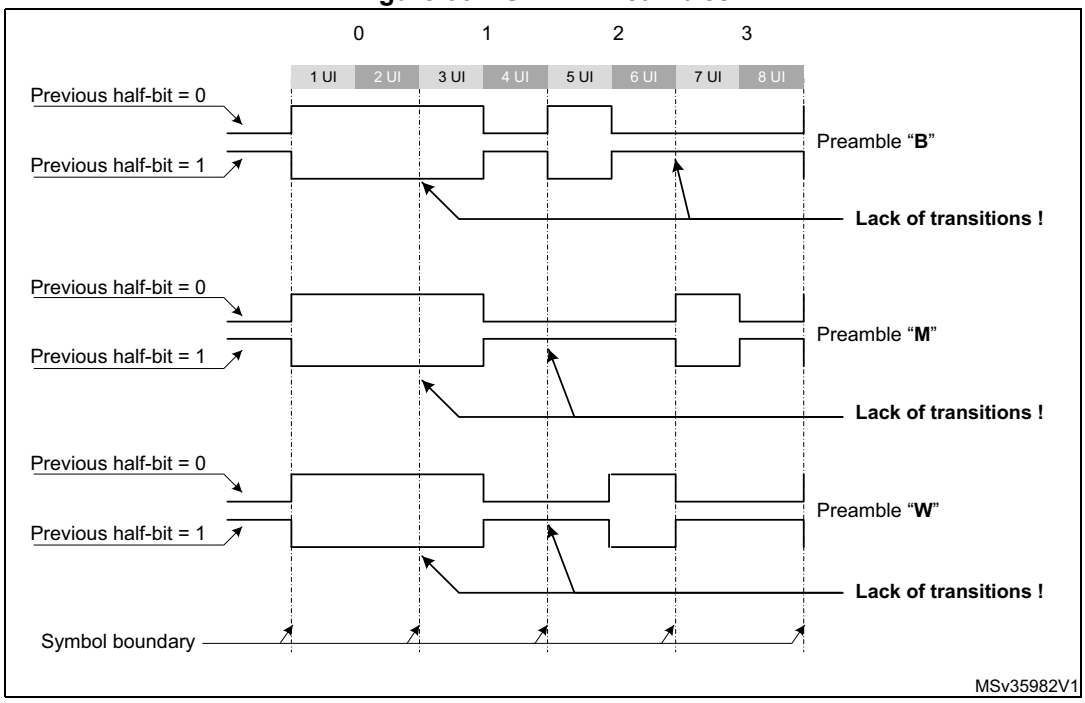
Figure 666. S/PDIF block format



Synchronization preambles

The preambles patterns are inverted or not according to the previous half-bit value. This previous half-bit value is the level of the line before enabling a transfer for the first “B” preamble of the first frame. For the others preambles, this previous half-bit value is the second half-bit of the parity bit of the previous sub-frame. The preambles patterns B, M and W are described in the *Figure 667*.

Figure 667. S/PDIF Preambles



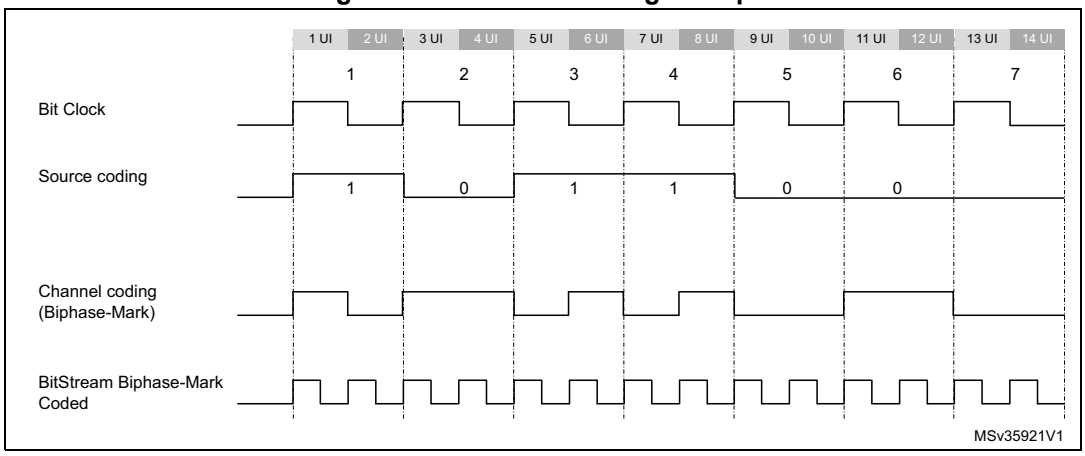
Coding of information bits

In order to minimize the DC component value on the transmission line, and to facilitate clock recovery from the data stream, bits 4 to 31 are encoded in biphase-mark.

Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is logical 0. However, it is different if the bit is logical 1. These states are named “UI” (Unit Interval) in the IEC-60958 specification.

The 24 data bits are transferred LSB first.

Figure 668. Channel coding example



51.3.3 SPDIFRX decoder (SPDIFRX_DC)

Main principle

The technique used by the SPDIFRX in order to decode the S/PDIF stream is based on the measurement of the time interval between two consecutive edges. Three kinds of time intervals may be found into an S/PDIF stream:

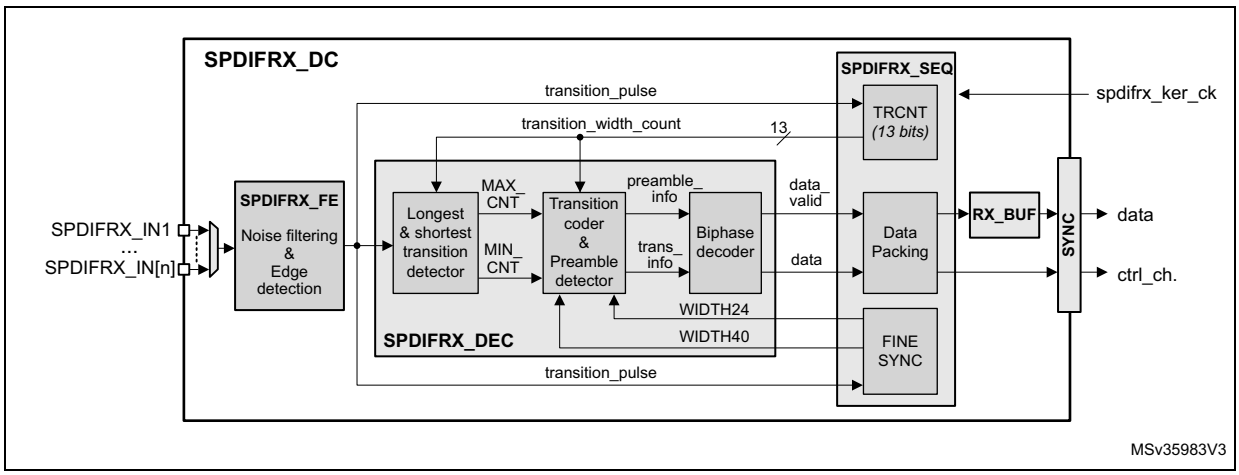
- The long time interval, having a duration of 3 x UI, noted TL. It appears only during preambles.
- The medium time interval, having a duration of 2 x UI, noted TM. It appears both in some preambles or into the information field.
- The short time interval, having a duration of 1 x UI, noted TS. It appears both in some preambles or into the information field.

The SPDIFRX_DC block is responsible of the decoding of the received S/PDIF stream. It takes care of the following functions:

- Resampling and filtering of the incoming signal
- Estimation of the time-intervals
- Estimation of the symbol rate and synchronization
- Decoding of the serial data, and check of integrity
- Detection of the block, and sub-frame preambles
- Continuous tracking of the symbol rate

Figure 669 gives a detailed view of the SPDIFRX decoder.

Figure 669. SPDIFRX decoder

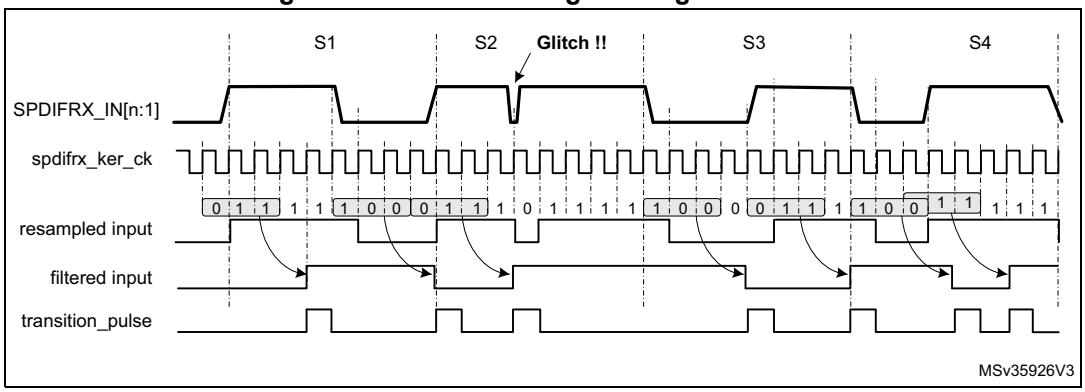


Noise filtering & rising/falling edge detection

The S/PDIF signal received on the selected SPDIFRX_IN is re-sampled using the spdifrx_ker_ck clock (acquisition clock). A simple filtering is applied in order cancel spurs. This is performed by the stage detecting the edge transitions. The edge transitions are detected as follow:

- A rising edge is detected when the sequence 0 followed by two 1 is sampled.
- A falling edge is detected when the sequence 1 followed by two 0 is sampled.
- After a rising edge, a falling edge sequence is expected.
- After a falling edge, a rising edge sequence is expected.

Figure 670. Noise filtering and edge detection



Longest and shortest transition detector

The **longest and shortest transition detector** block detects the maximum (MAX_CNT) and minimum (MIN_CNT) duration between two transitions. The TRCNT counter is used to measure the time interval duration. It is clocked by the spdifrx_ker_ck signal. On every transition pulse, the counter value is stored and the counter is reset to start counting again.

The maximum duration is normally found during the preamble period. This maximum duration is sent out as MAX_CNT. The minimum duration is sent out as MIN_CNT.

The search of the longest and shortest transition is stopped when the transition timer expires. The transition timer is like a watchdog timer that generates a trigger after 70 transitions of the incoming signal. Note that counting 70 transitions insures a delay a bit longer than a sub-frame.

Note that when the TRCNT overflows due to a too long time interval between two pulses, the SPDIFRX is stopped and the flag TERR of SPDIFRX_SR register is set to 1.

Transition coder and preamble detector

The **transition coder and preamble detector** block receives the MAX_CNT and MIN_CNT. It also receives the current transition width from the TRCNT counter (see *Figure 669*). This block encodes the current transition width by comparing the current transition width with two different thresholds, names TH_{HI} and TH_{LO}.

- If the current transition width is less than (TH_{LO} - 1), then the data received is half part of data bit '1', and is coded as TS.
- If the current transition width is greater than (TH_{LO} - 1), and less than TH_{HI}, then the data received is data bit '0', and is coded as TM.
- If the current transition width is greater than TH_{HI}, then the data received is the long pulse of preambles, and is coded as TL.
- Else an error code is generated (FERR flag is set).

The thresholds TH_{HI} and TH_{LO} are elaborated using two different methods.

If the peripheral is doing its initial synchronization ('coarse synchronization'), then the thresholds are computed as follow:

- TH_{LO} = MAX_CNT / 2.
- TH_{HI} = MIN_CNT + MAX_CNT / 2.

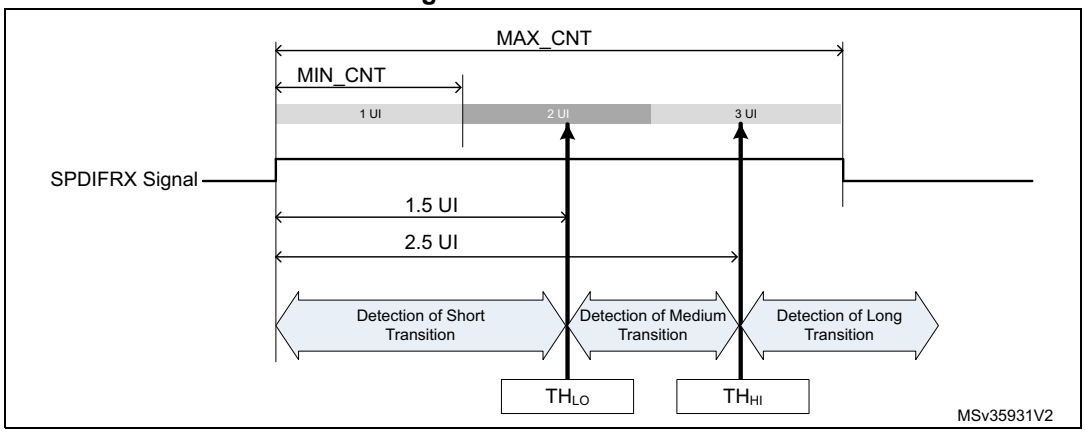
Once the 'coarse synchronization' is completed, then the SPDIFRX uses a more accurate reference in order to elaborate the thresholds. The SPDIFRX measures the length of 24 symbols (WIDTH24) for defining TH_{LO} and the length of 40 symbols (WIDTH40) for TH_{HI}. TH_{HI} and TH_{LO} are computed as follow:

- TH_{LO} = (WIDTH24) / 32
- TH_{HI} = (WIDTH40) / 32

This second synchronization phase is called the 'fine synchronization'. Refer to *Figure 673* for additional information.

As shown in the figure hereafter, TH_{LO} is ideally equal to 1.5 UI, and to TH_{HI} 2.5 UI.

Figure 671. Thresholds



The preamble detector checks four consecutive transitions of a specific sequence to determine if they form the part of preamble. Let us say TRANS0, TRANS1, TRANS2 and TRANS3 represent four consecutive transitions encoded as mentioned above. *Table 420* shows the values of these four transitions to form a preamble. Absence of this pattern indicates that these transitions form part of the data in the sub frame and bi-phase decoder will decode them.

Table 420. Transition sequence for preamble

Preamble type	Biphase data pattern	TRANS3	TRANS2	TRANS1	TRANS0
Preamble B	11101000	TL	TS	TS	TL
Preamble M	11100010	TL	TL	TS	TS
Preamble W	11100100	TL	TM	TS	TM

Bi-phase decoder

The Bi-phase decoder decodes the input bi-phase marked data stream using the transition information provided by the **transition coder and preamble detector** block. It first waits for the preamble detection information. After the preamble detection, it decodes the following transition information:

- If the incoming transition information is TM then it is decoded as a ‘0’.
- Two consecutive TS are decoded as a ‘1’.
- Any other transition sequence generates an error signal (FERR set to 1).

After decoding 28 data bits this way, this module looks for the following preamble data. If the new preamble is not what is expected, then this block generates an error signal (FERR set to 1). Refer to *Section 51.3.9: Reception errors*, for additional information on error flags.

Data packing

This block is responsible of the decoding of the IEC-60958 frames and blocks. It also handles the writing into the RX_BUF or into SPDIFRX_CSR register.

51.3.4 SPDIFRX tolerance to clock deviation

The SPDIFRX tolerance to clock deviation depends on the number of sample clock cycles in one bit slot. The fastest spdifrx_ker_ck is, the more robust the reception will be. The ratio between spdifrx_ker_ck frequency and the symbol rate must be at least 11.

Two kinds of phenomenon (at least!) can degrade the reception quality:

- The cycle-to-cycle jitter which reflects the difference of transition length between two consecutive transitions.
- The long term jitter which reflects a cumulative effect of the cycle-to-cycle jitter. It can be seen as a low-frequency symbol modulation.

51.3.5 SPDIFRX synchronization

The synchronization phase starts when setting SPDIFRXEN to 01 or 11. *Figure 672* shows the synchronization process.

If the bit WFA of SPDIFRX_CR register is set to 1, then the peripheral must first detect activity on the selected SPDIFRX_IN line before starting the synchronization process. The activity detection is performed by detecting four transitions on the selected SPDIFRX_IN. The peripheral remains in this state until transitions are not detected. This function can be particularly helpful because the IP switches in COARSE SYNC mode only if activity is present on the selected SPDIFRX_IN input, avoiding synchronization errors. See *Section 51.4: Programming procedures* for additional information.
The user can still set the SPDIFRX into STATE_IDLE by setting SPDIFRXEN to 0. If the WFA is set to 0, the peripheral starts the coarse synchronization without checking activity.

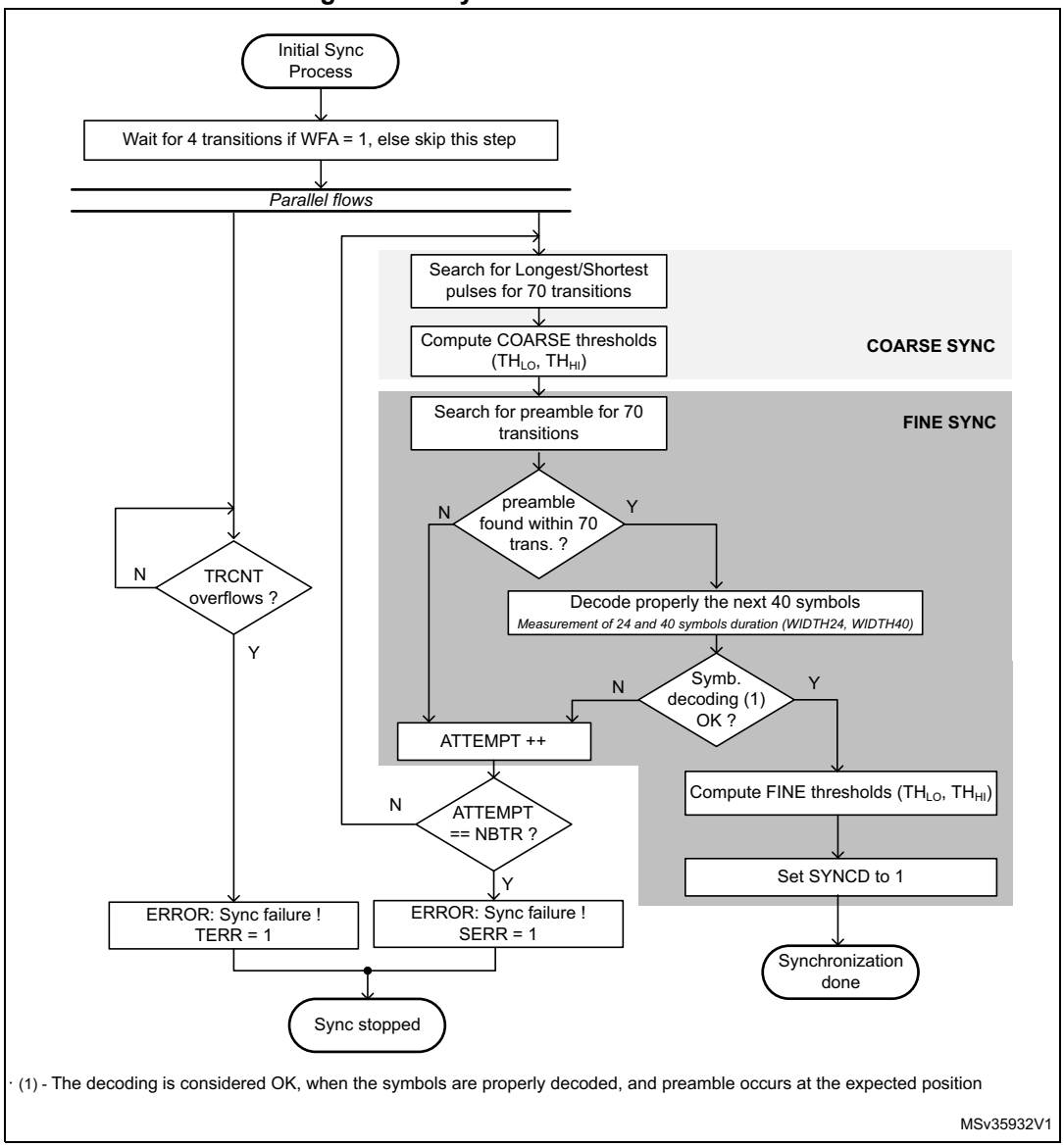
The next step consists on doing a first estimate of the thresholds (COARSE SYNC), in order to perform the fine synchronization (FINE SYNC). Due to disturbances of the SPDIFRX line, it could happen that the process is not executed first time right. For this purpose, the user can program the number of allowed re-tries (NBTR) before setting SERR error flag.
When the SPDIFRX has been able to measure properly the duration of 24 and 40 consecutive symbols then the FINE SYNC is completed, the threshold values are updated, and the flag SYNCD is set to 1. Refer to *Section : Transition coder and preamble detector* for additional information.

Two kinds of errors are detected:

- An overflow of the TRCNT, which generally means that there is no valid S/PDIF stream in the input line. This overflow is indicated by TERR flag.
- The number of retries reached the programmed value. This means that strong jitter is present on the S/PDIF signal. This error is indicated by SERR flag.

When the first FINE SYNC is completed, the reception of channel status (C) and user data (U) will start when the next “B” preamble is detected (see *Figure 676*).Then the user can read IEC-60958 C and U bits through SPDIFRX_CSR register. According to this information the user can then select the proper settings for DRFMT and RXSTEO. For example if the user detects that the current audio stream transports encoded data, then he can put RXSTEO to 0, and DRFMT to 10 prior to start data reception. Note that DRFMT and RXSTEO cannot be modified when SPDIFRXEN = 11. Writes to these fields are ignored if SPDIFRXEN is already 11, though these field can be changed with the same write instruction that causes SPDIFRXEN to become 11.
Then the SPDIFRX waits for SPDIFRXEN = 11 and the “B” preamble before starting saving audio samples.

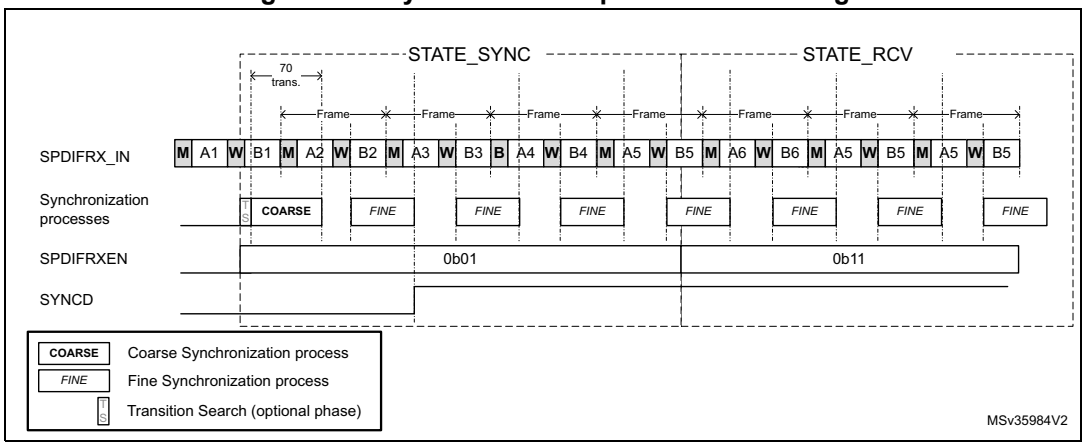
Figure 672. Synchronization flowchart



Refer to *Frame structure and synchronization error* for additional information concerning TRCNT overflow.

The FINE SYNC process is re-triggered every frame in order to update thresholds as shown in *Figure 673* in order to continuously track S/PDIF synchronization.

Figure 673. Synchronization process scheduling



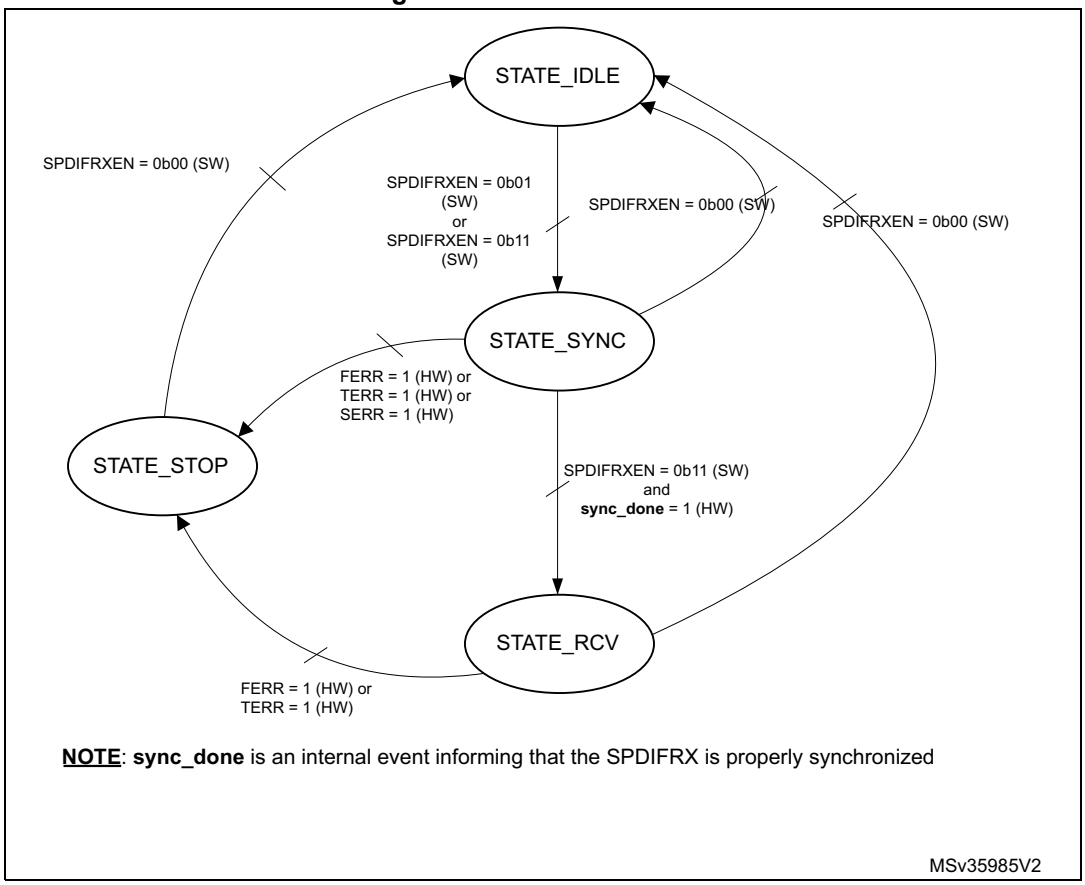
51.3.6 SPDIFRX handling

The software can control the state of the SPDIFRX through SPDIFRXEN field. The SPDIFRX can be into one of the following states:

- STATE_IDLE:
  The peripheral is disabled, the spdifrx_ker_ck domain is reset. The spdifrx_pclk domain is functional.
- STATE_SYNC:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user and channel status can be read via interrupt of DMA. The audio samples are not provided to receive buffer.
- STATE_RCV:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user, channel status and audio samples can be read via interrupt or DMA channels. When SPDIFRXEN goes to 11, the SPDIFRX waits for “B” preamble before starting saving audio samples.
- STOP_STATE:
  The peripheral is no longer synchronized, the reception of the user, channel status and audio samples are stopped. It is expected that the software re-starts the SPDIFRX.

The *Figure 674* shows the possible states of the SPDIFRX, and how to transition from one state to the other. The bits under software control are followed by the mention “(SW)”, the bits under IP control are followed by the mention “(HW)”.

Figure 674. SPDIFRX States



When SPDIFRX is in STATE_IDLE:

- The software can transition to STATE_SYNC by setting SPDIFRXEN to 01 or 11

When SPDIFRX is in STATE_SYNC:

- If the synchronization fails or if the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- When the synchronization phase is completed, if SPDIFRXEN = 01 the peripheral remains in this state.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it will be properly completed.
- The SPDIFRX goes to STATE_RCV if SPDIFRXEN = 11 and if the SYNCD = 1

When SPDIFRX is in STATE_RCV:

- If the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it will properly be completed.

When SPDIFRX is in STATE_STOP:

- The SPDIFRX stops reception and synchronization, and waits for the software to set the bit SPDIFRXEN to 0, in order to clear the error flags.

When SPDIFRXEN is set to 0, the IP is disabled, meaning that all the state machines are reset, and RX_BUF is flushed. Note as well that flags FERR, SERR and TERR are reset.

51.3.7 Data reception management

The SPDIFRX offers a double buffer for the audio sample reception. A 32-bit buffer located into the spdifrx_ker_ck clock domain (RX_BUF), and the SPDIFRX_FMTx_DR register. The valid data contained into the RX_BUF will be immediately transferred into SPDIFRX_FMTx_DR if SPDIFRX_FMTx_DR is empty.

The valid data contained into the RX_BUF will be transferred into SPDIFRX_FMTx_DR when the two following conditions are reached:

- The transition between the parity bit (P) and the next preamble is detected (this indicated that the word has been completely received).
- The SPDIFRX_FMTx_DR is empty.

Having a 2-word buffer gives more flexibility for the latency constraint.

The maximum latency allowed is T_{SAMPLE} - $2T_{PCLK}$ - $2T_{spdifrx_ker_ck}$

Where T_{SAMPLE} is the audio sampling rate of the received stereo audio samples, T_{PCLK} is the period of spdifrx_pclk clock, and $T_{spdifrx_ker_ck}$ is the period of spdifrx_ker_ck clock.

The SPDIFRX offers the possibility to use either DMA (spdifrx_dat_dma and spdifrx_ctrl_dma) or interrupts for transferring the audio samples into the memory. The recommended option is DMA, refer to *Section 51.3.11: DMA Interface* for additional information.

The SPDIFRX offers several way on handling the received data. The user can either have a separate flow for control information and audio samples, or get them all together.

For each sub-frame, the data reception register SPDIFRX_FMTx_DR contains the 24 data bits, and optionally the V, U, C, PE status bits, and the PT (see *Mixing data and control flow*).

Note that PE bit stands for Parity Error bit, and will be set to 1 when a parity error is detected in the decoded sub-frame.
The PT field carries the preamble type (B, M or W).

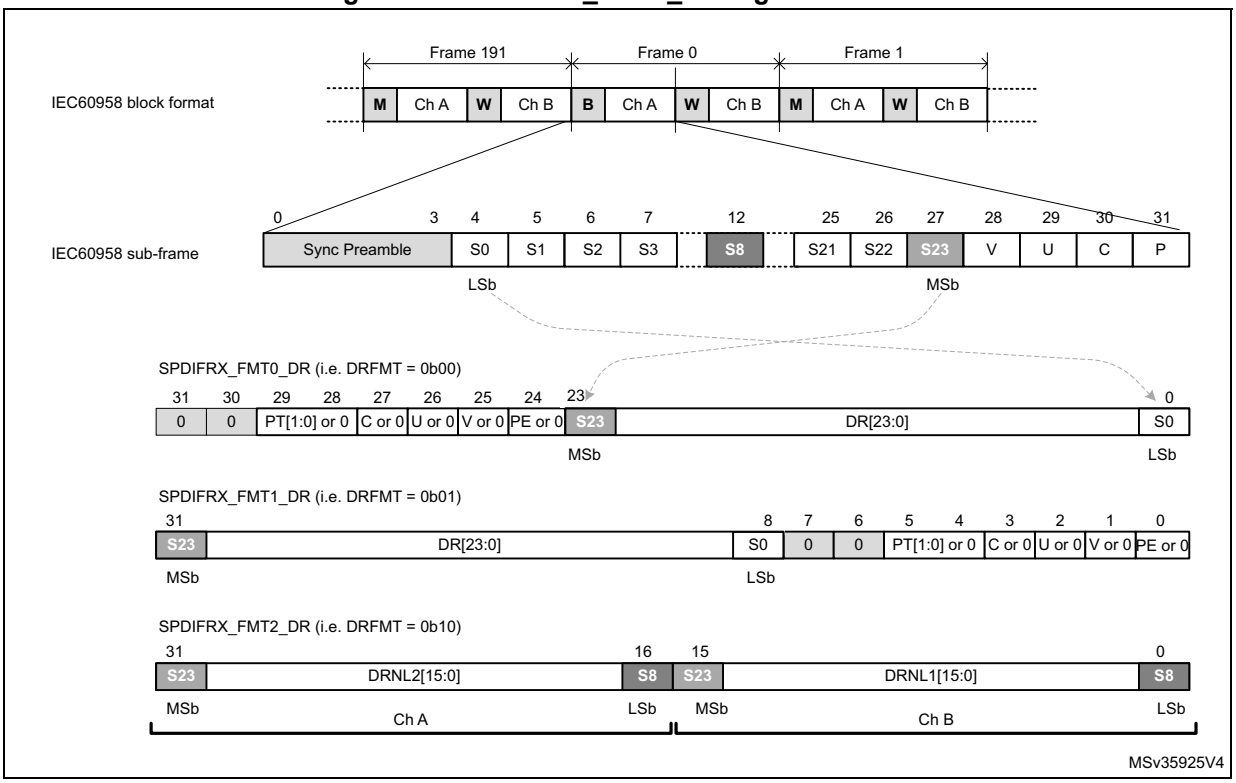
V, U and C are a direct copy of the value received from the S/PDIF interface.

The bit DRFMT allows the selection between 3 audio formats as shown in *Figure 675*.

This document describes 3 data registers: SPDIFRX_FMTx[2:0] (x = 2 to 0), but in reality there is only one physical data register, having 3 possible formats:

- When DRFMT = 0, the format of the data register is the one described by SPDIFRX_FMT0_DR
- When DRFMT = 1, the format of the data register is the one described by SPDIFRX_FMT1_DR
- When DRFMT = 2, the format of the data register is the one described by SPDIFRX_FMT2_DR"

Figure 675. SPDIFRX_FMTx_DR register format



Setting DRFMT to 00 or 01, offers the possibility to have the data either right or left aligned into the SPDIFRX_FMTx_DR register. The status information can be enabled or forced to zero according to the way the software wants to handle them.

The format given by DRFMT= 10 is interesting in non-linear mode, as only 16 bits per sub-frame are used. By using this format, the data of two consecutive sub-frames are stored into SPDIFRX_FMTx_DR, dividing by two the amount of memory footprint. Note that when RXSTEO = 1, there is no misalignment risks (i.e. data from ChA will be always stored into SPDIFRX_FMTx_DR[31:16]). If RXSTEO = 0, then there is a misalignment risk is case of overrun situation. In that case SPDIFRX_FMTx_DR[31:16] will always contain the oldest value and SPDIFRX_FMTx_DR[15:0] the more recent value (see *Figure 677*).

In this format the status information cannot be mixed with data, but the user can still get them through SPDIFRX_CSR register, and use a dedicated DMA channel or interrupt to transfer them to memory (see *Section 51.3.8: Dedicated control flow*)

Mixing data and control flow

The user can choose to use this mode in order to get the full flexibility of the handling of the control flow. The user can select which field shall be kept into the data register (SPDIFRX_FMTx_DR).

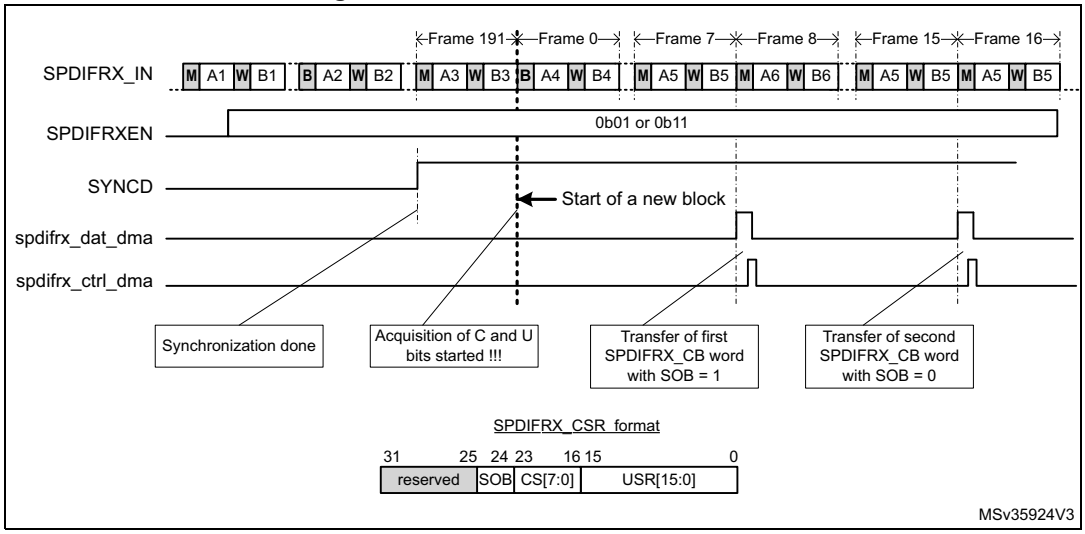
- When bit PMSK = 1, the Parity Error information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit VMSK = 1, the Validity information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit CUMSK = 1, the Channel Status, and Used data information are masked (set to 0), otherwise they are copied into SPDIFRX_FMTx_DR.
- When bit PTMSK = 1, the Preamble Type is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

51.3.8 Dedicated control flow

The SPDIFRX offers the possibility to catch both user data and channel status information via a dedicated DMA channel. This feature allows the SPDIFRX to acquire continuously the channel status and user information. The acquisition will start at the beginning of a IEC 60958 block. Two fields are available to control this path: CBDMAEN and SPDIFRXEN. When SPDIFRXEN is set to 01 or 0x11, the acquisition is started, after completion of the synchronization phase. When 8 channel status and 16 user data bits have been received, they are packed and stored into SPDIFRX_CSR register. A DMA request is triggered if the bit CBDMAEN is set to 1 (see *Figure 676*).

If CS[0] corresponds to the first bit of a new block, the bit SOB will be set to 1. Refer to *Section 51.5.8: Channel status register (SPDIFRX_CSR)*. A bit is available (CHSEL) in order to select if the user wants to select channel status information (C) from the channel A or B.

Figure 676. Channel/user data format



Note: Once the first start of block is detected (B preamble), the SPDIFRX is checking the preamble type every 8 frames.

Note: Overrun error on SPDIFRX_FMTx_DR register does not affect this path.

51.3.9 Reception errors

Frame structure and synchronization error

The SPDIFRX, detects errors, when one of the following condition occurs:

- The FERR bit is set to 1 on the following conditions:
  - For each of the 28 information bits, if one symbol transition sequence is not correct: for example if short pulses are not grouped by pairs.
  - If preambles occur to an unexpected place, or an expected preamble is not received.
- The SERR bit is set when the synchronization fails, because the number of re-tries exceeded the programmed value.
- The TERR bit is set when the counter used to estimate the width between two transitions overflows (TRCNT).
  The overflow occurs when no transition is detected during 8192 periods of spdifrx_ker_ck clock. It represents at most a time interval of 11.6 frames.

When one of those flags goes to 1, the traffic on selected SPDIFRX_IN is then ignored, an interrupt is generated if the IFEIE bit of the SPDIFRX_CR register is set.

The normal procedure when one of those errors occur is:

- Set SPDIFRXEN to 0 in order to clear the error flags
- Set SPDIFRXEN to 01 or 11 in order to restart the IP

Refer to *Figure 674* for additional information.

Parity error

For each sub-frame, an even number of zeros and ones is expected inside the 28 information bits. If not, the parity error bit PERR is set in the SPDIFRX_SR register and an interrupt is generated if the parity interrupt enable PERRIE bit is set in the SPDIFRX_CR register. The reception of the incoming data is not paused, and the SPDIFRX continue to deliver data to SPDIFRX_FMTx_DR even if the interrupt is still pending.

The interrupt is acknowledged by clearing the PERR flag through PERRCF bit.

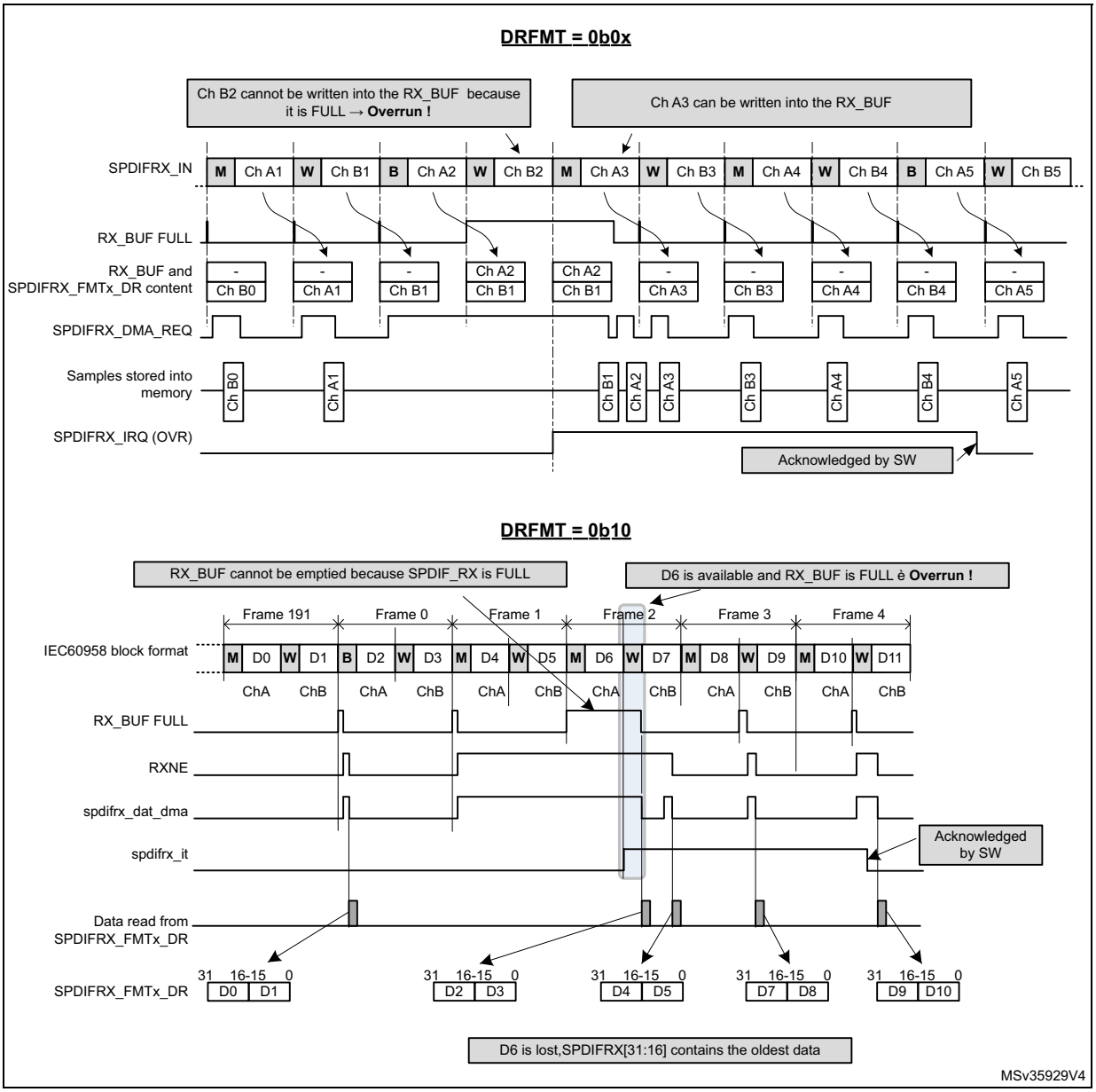
If the software wants to guarantee the coherency between the data read in the SPDIFRX_FMTx_DR register and the value of the bit PERR, the bit PMSK must be set to 0.

Overrun error

If both SPDIFRX_FMTx_DR and RX_BUF are full, while the SPDIFRX_DC needs to write a new sample in RX_BUF, this new sample is dropped, and an overrun condition is triggered. The overrun error flag OVR is set in the SPDIFRX_SR register and an interrupt is generated if the OVRIE bit of the SPDIFRX_CR register is set.

If the RXSTEO bit is set to 0, then as soon as the RX_BUF is empty, the IP will store the next incoming data, even if the OVR flag is still pending. The main purpose is to reduce as much as possible the amount of lost samples. Note that the behavior is similar independently of DRFMT value. See *Figure 677*.

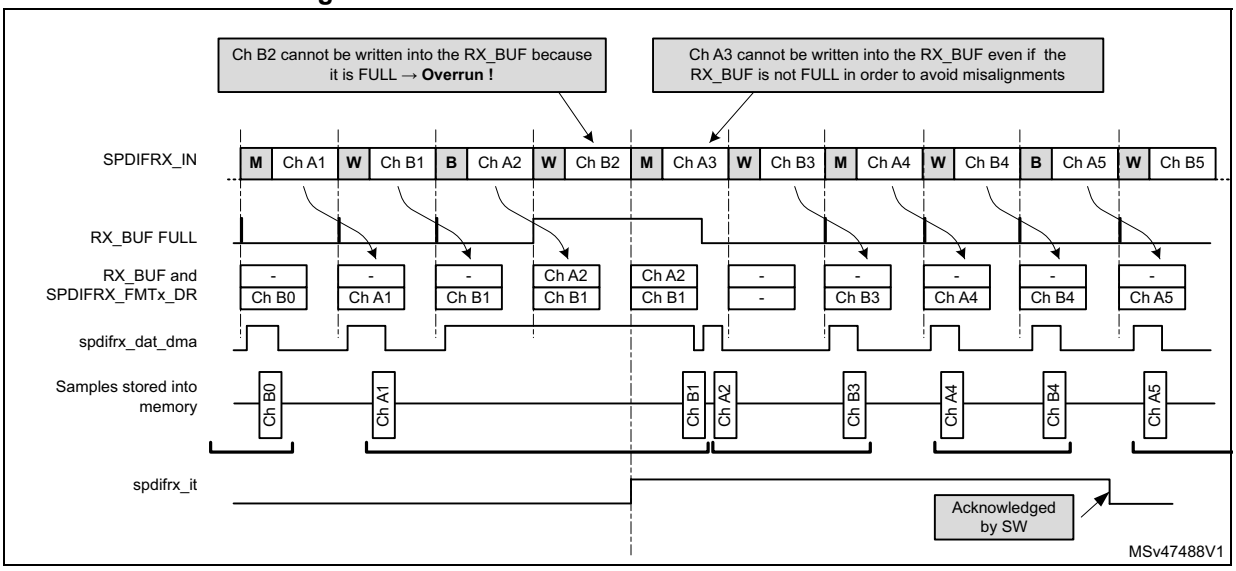
Figure 677. S/PDIF overrun error when RXSTEO = 0



If the RXSTEO bit is set to 1, it means that stereo data are transported, then the SPDIFRX has to avoid misalignment between left and right channels. So the peripheral has to drop a second sample even if there is room inside the RX_BUF in order to avoid misalignment. Then the incoming samples can be written normally into the RX_BUF even if the OVR flag is still pending. Refer to *Figure 678*.

The OVR flag is cleared by software, by setting the OVRCF bit to 1.

Figure 678. S/PDIF overrun error when RXSTEO = 1



51.3.10 Clocking strategy

The SPDIFRX block needs two different clocks:

- The APB clock (spdifrx_pclk), which is used for the register interface,
- The spdifrx_ker_ck which is mainly used by the SPDIFRX_DC part. Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized (SYNC block on *Figure 664*).

In order to decode properly the incoming S/PDIF stream the SPDIFRX_DC shall re-sample the received data with a clock at least 11 times higher than the maximum symbol rate, or 704 times higher than the audio sample rate. For example if the user expects to receive a symbol rate to up to 12.288 MHz, the sample rate shall be at least 135.2 MHz. The clock used by the SPDIFRX_DC is the spdifrx_ker_ck.

The frequency of the spdifrx_pclk must be at least equal to the symbol rate.

Table 421. Minimum spdifrx_ker_ck frequency versus audio sampling rate

Symbol Rate	Minimum spdifrx_ker_ck frequency	Comments
3.072 MHz	33.8 MHz	For 48 kHz stream
6.144 MHz	67.6 MHz	For 96 kHz stream
12.288 MHz	135.2 MHz	For 192 kHz stream

51.3.11 DMA Interface

The SPDIFRX interface is able to perform communication using the DMA.

Note: The user should refer to product specifications for availability of the DMA controller.

The SPDIFRX offers two independent DMA channels:

- A DMA channel dedicated to the data transfer
- A DMA channel dedicated to the channel status and user data transfer

The DMA mode for the data can be enabled for reception by setting the RXDMAEN bit in the SPDIFRX_CR register. In this case, as soon as the SPDIFRX_FMTx_DR is not empty, the SPDIFRX interface sends a transfer request to the DMA. The DMA reads the data received through the SPDIFRX_FMTx_DR register without CPU intervention.

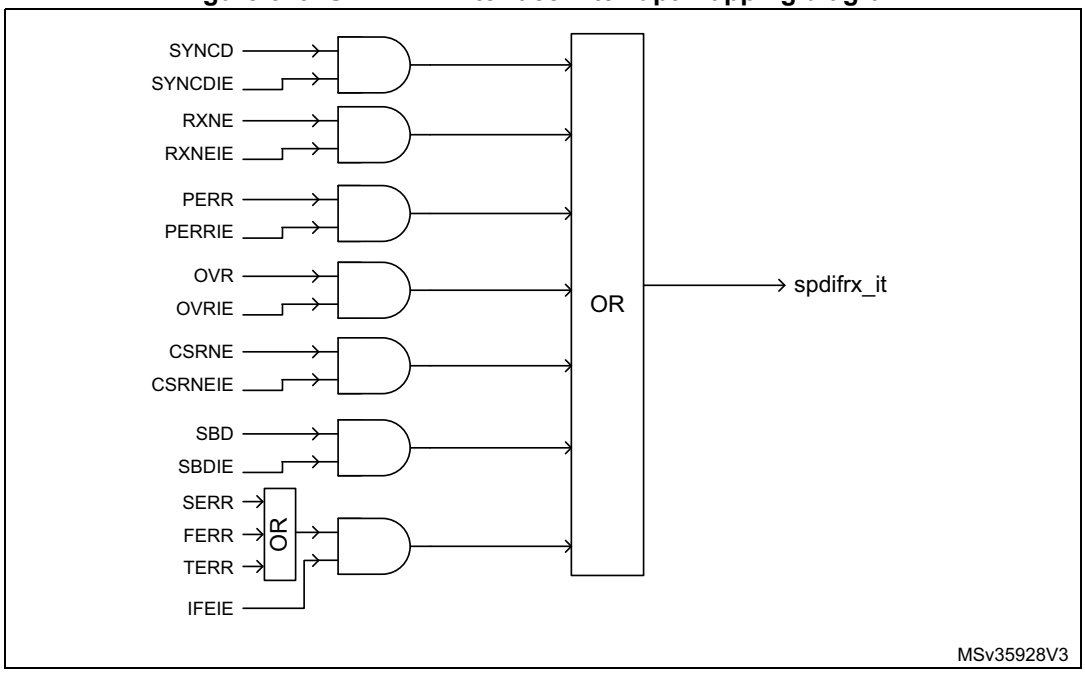
For the use of DMA for the control data refer to *Section 51.3.8: Dedicated control flow*.

51.3.12 Interrupt Generation

An interrupt line is shared between:

- Reception events for data flow (RXNE)
- Reception event for control flow (CSRNE)
- Data corruption detection (PERR)
- Transfer flow interruption (OVR)
- Frame structure and synchronization errors (SERR, TERR and FERR)
- Start of new block interrupt (SBD)
- Synchronization done (SYNCD)

Figure 679. SPDIFRX interface interrupt mapping diagram



Clearing interrupt source

- RXNE is cleared when SPDIFRX_FMTx_DR register is read
- CSRNE is cleared when SPDIFRX_CSR register is read
- FERR is cleared when SPDIFRXEN is set to 0
- SERR is cleared when SPDIFRXEN is set to 0
- TERR is cleared when SPDIFRXEN is set to 0
- Others are cleared through SPDIFRX_IFCR register

Note: The SBD event can only occur when the SPDIFRX is synchronized to the input stream (SYNCD = 1).
The SBD flag behavior is not guaranteed when the sub-frame which contains the B preamble is lost due to an overrun.

51.3.13 Register protection

The SPDIFRX block embeds some hardware protection avoid erroneous use of control registers. The table hereafter shows the bit field properties according to the SPDIFRX state.

Table 422. Bit field property versus SPDIFRX state

Registers	Field	SPDIFRXEN		
		00 (STATE_IDLE)	01 (STATE_SYNC)	11 (STATE_RCV)
SPDIFRX_CR	INSEL	rw	r	r
	WFA	rw	r	r
	NBTR	rw	r	r
	CHSEL	rw	r	r
	CBDMAEN	rw	rw	rw
	PTMSK	rw	rw	rw
	CUMSK	rw	rw	rw
	VMSK	rw	rw	rw
	PMSK	rw	rw	rw
	DRFMT	rw	rw	r
	RXSTEO	rw	rw	r
	RXDMAEN	rw	rw	rw
SPDIFRX_IMR	All fields	rw	rw	rw

The table clearly shows that fields such as INSEL must be programmed when the IP is in STATE_IDLE. In the others IP states, the hardware prevents writing to this field.

Note: Even if the hardware allows the writing of CBDMAEN and RXDMAEN “on-the-fly”, it is not recommended to enable the DMA when the IP is already receiving data.

Note: Note that each of the mask bits (PMSK, VMSK, …) can be changed “on-the-fly” at any IP state, but any change does not affect data which is already being held in SPDIFRX_FMTx_DR.

51.4 Programming procedures

The following example illustrates a complete activation sequence of the SPDIFRX block. The data path and channel status & user information will both use a dedicated DMA channel. The activation sequence is then split into the following steps:

- Wait for valid data on the selected SPDIFRX_IN input
- Synchronize to the S/PDIF stream
- Read the channel status and user information in order to setup the complete audio path
- Start data acquisition

A simple way to check if valid data are available into the SPDIFRX_IN line is to switch the SPDIFRX into the STATE_SYNC, with bit WFA set to 1. The description hereafter will focus on detection. It is also possible to implement this function as follow:

- The software has to check from time to time (i.e. every 100 ms for example) if the SPDIFRX can find synchronization. This can be done by checking if the bit TERR is set. When it is set it indicates that no activity as been found.
- Connect the SPDIFRX_IN input to an external interrupt event block in order to detect transitions of SPDIFRX_IN line. When activity is detected, then SPDIFRXEN can be set to 01 or 11.

  For those two implementations, the bit WFA is set to 0.

51.4.1 Initialization phase

- The initialization function will look like this:
- Configure the DMA transfer for both audio samples and IEC60958 channel status and user information (DMA channel selection and activation, priority, number of data to transfer, circular/no circular mode, DMA interrupts)
- Configure the destination address:
  - Configure the address of the SPDIFRX_CSR register as source address for IEC60958 channel status and user information
  - Configure the address of the SPDIFRX_FMTx_DR register as source address for audio samples
  - Enable the generation of the spdifrx_ker_ck. Refer to *Table 421* in order to define the minimum clock frequency versus supported audio sampling rate.
    Note that the audio sampling rate of the received stream is not known in advance. This means that the user has to select a spdifrx_ker_ck frequency at least 704 times higher than the maximum audio sampling rate the application is supposed to

handle: for example if the application is able to handle streams to up to 96 kHz, then $F_{spdifrx_ker_ck}$ shall be at least 704 x 96 kHz = 67.6 MHz

- Enable interrupt for errors and event signaling (IFEIE = SYNCDIE = OVRIE, PERRIE = 1, others set to 0). Note that SYNCDIE can be set to 0.
- Configure the SPDIFRX_CR register:
  - INSEL shall select the wanted input
  - NBTR = 2, WFA = 1 (16 re-tries allowed, wait for activity before going to synchronization phase),
  - PTMSK = CUMSK = 1 (Preamble, C and U bits are not mixed with data)
  - VMSK = PMSK = 0 (Parity error and validity bit mixed with data)
  - CHSEL = 0 (channels status will be read from sub-frame A)
  - DRFMT = 01 (data aligned to the left)
  - RXSTEO = 1 (expected stereo mode linear)
  - CBDMAEN = RXDMAEN = 1 (enable DMA channels)
  - SPDIFRXEN = 01 (switch SPDIFRX to STATE_SYNC)
- The CPU can enter in WFI mode

Then the CPU will receive interrupts coming either from DMA or SPDIFRX.

51.4.2 Handling of interrupts coming from SPDIFRX

When an interrupt from the SPDIFRX is received, then the software has to check what is the source of the interrupt by reading the SPDIFRX_SR register.

- If SYNCD is set to 1, then it means that the synchronization has been properly completed. No action has to be performed in our case as the DMA is already programmed. The software just needs to wait for DMA interrupt in order to read channel status information.
  The SYNCD flag must be cleared by setting SYNCDCF bit of SPDIFRX_IFCR register to 1.
- If TERR or SERR or FERR are set to 1, the software has to set SPDIFRXEN to 0, and re-start from the initialization phase.
  - TERR indicates that a time-out occurs either during synchronization phase or after.
  - SERR indicates that the synchronization fails because the maximum allowed re-tries have been reached.
  - FERR indicates that the reading of information after synchronization fails (unexpected preamble, bad data decoding...).
- If PERR is set to 1, it means that a parity error has been detected, so one of the received audio sample or the channel status or user data bits are corrupted. The action taken here depends on the application: one action could be to drop the current channel status block as it is not reliable. There is no need to re-start from the initialization phase, as the synchronization is not lost.
  The PERR flag must be cleared by setting PERRCF bit of SPDIFRX_IFCR register to 1.

51.4.3 Handling of interrupts coming from DMA

If an interrupt is coming from the DMA channel used of the channel status (SPDIFRX_CSR):

If no error occurred (i.e. PERR), the CPU can start the decoding of channel information. For example bit 1 of the channel status informs the user if the current stream is linear or not. This information is very important in order to set-up the proper processing chain. In the same way, bits 24 to 27 of the channel status give the sampling frequency of the stream incoming stream.

Thanks to that information, the user can then configure the RXSTEO bit and DRFMT field prior to start the data reception. For example if the current stream is non linear PCM then RXSTEO is set to 0, and DRFMT is set to 10. Then the user can enable the data reception by setting SPDIFRXEN to 11.

The bit SOB, when set to 1 indicates the start of a new block. This information will help the software to identify the bit 0 of the channel status. Note that if the DMA generates an interrupt every time 24 values are transferred into the memory, then the first word will always correspond to the start of a new block.

If an interrupt is coming from the DMA channel used of the audio samples (SPDIFRX_FMTx_DR):

The process performed here depends of the data type (linear or non-linear), and on the data format selected.

For example in linear mode, if PE or V bit is set a special processing can be performed locally in order to avoid spurs on output. In non-linear mode those bits are not important as data frame have their own checksum.

51.5 SPDIFRX interface registers

51.5.1 Control register (SPDIFRX_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSEL[2:0]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **INSEL[2:0]**: SPDIFRX input selection(1)
000: SPDIFRX_IN1 selected
001: SPDIFRX_IN2 selected
010: SPDIFRX_IN3 selected
011: SPDIFRX_IN4 selected
others reserved

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WFA**: Wait for activity(1)
This bit is set/reset by software
1: The SPDIFRX waits for activity on SPDIFRX_IN line (4 transitions) before performing the synchronization
0: The SPDIFRX does not wait for activity on SPDIFRX_IN line before performing the synchronization

Bits 13:12 **NBTR[1:0]**: Maximum allowed re-tries during synchronization phase(1)
00: No re-try is allowed (only one attempt)
01: 3 re-tries allowed
10: 15 re-tries allowed
11: 63 re-tries allowed

Bit 11 **CHSEL**: Channel selection(1)
This bit is set/reset by software
1: The control flow will take the channel status from channel B
0: The control flow will take the channel status from channel A

Bit 10 **CBDMAEN**: Control buffer DMA enable for control flow(1)
This bit is set/reset by software
1: DMA mode is enabled for reception of channel status and used data information.
0: DMA mode is disabled for reception of channel status and used data information.
When this bit is set, the DMA request is made whenever the CSRNE flag is set.

Bit 9 **PTMSK**: Mask of preamble type bits(1)
This bit is set/reset by software
1: The preamble type bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead
0: The preamble type bits are copied into the SPDIFRX_FMTx_DR

Bit 8 **CUMSK**: Mask of channel status and user bits(1)
This bit is set/reset by software
1: The channel status and user bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead
0: The channel status and user bits are copied into the SPDIFRX_FMTx_DR

Bit 7 **VMSK**: Mask of validity bit(1)
This bit is set/reset by software
1: The validity bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead
0: The validity bit is copied into the SPDIFRX_FMTx_DR

Bit 6 **PMSK**: Mask parity error bit(1)
This bit is set/reset by software
1: The parity error bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead
0: The parity error bit is copied into the SPDIFRX_FMTx_DR

Bits 5:4 **DRFMT[1:0]**: RX data format(1)
This bit is set/reset by software
11: reserved
10: Data sample are packed by setting two 16-bit sample into a 32-bit word
01: Data samples are aligned in the left (MSB)
00: Data samples are aligned in the right (LSB)

Bit 3 **RXSTEO**: Stereo mode(1)
This bit is set/reset by software
1: The peripheral is in STEREO mode
0: The peripheral is in MONO mode
This bit is used in case of overrun situation in order to handle misalignment

Bit 2 **RXDMAEN**: Receiver DMA enable for data flow(1)
This bit is set/reset by software
1: DMA mode is enabled for reception.
0: DMA mode is disabled for reception.
When this bit is set, the DMA request is made whenever the RXNE flag is set.

Bits 1:0 **SPDIFRXEN[1:0]**: Peripheral block enable(1)
This field is modified by software.
It shall be used to change the peripheral phase among the three possible states: STATE_IDLE, STATE_SYNC and STATE_RCV.

00: Disable SPDIFRX (STATE_IDLE).
01: Enable SPDIFRX Synchronization only
10: Reserved
11: Enable SPDIF Receiver

Note: it is not possible to transition from STATE_RCV to STATE_SYNC, the user shall first go the STATE_IDLE.
it is possible to transition from STATE_IDLE to STATE_RCV: in that case the peripheral transitions from STATE_IDLE to STATE_SYNC and as soon as the synchronization is performed goes to STATE_RCV.

1. Refer to *Section 51.3.13: Register protection* for additional information on fields properties.

51.5.2 Interrupt mask register (SPDIFRX_IMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFE IE	SYNCD IE	SBLK IE	OVR IE	PERR IE	CSRNE IE	RXNE IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **IFEIE**: Serial Interface Error Interrupt Enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A SPDIFRX interface interrupt is generated whenever SERR=1, TERR=1 or FERR=1 in the SPDIFRX_SR register.

Bit 5 **SYNCDIE**: Synchronization Done
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A SPDIFRX interface interrupt is generated whenever SYNCD = 1 in the SPDIFRX_SR register.

Bit 4 **SBLKIE**: Synchronization Block Detected Interrupt Enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A SPDIFRX interface interrupt is generated whenever SBD = 1 in the SPDIFRX_SR register.

Bit 3 **OVRIE**: Overrun error Interrupt Enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A SPDIFRX interface interrupt is generated whenever OVR=1 in the SPDIFRX_SR register

Bit 2 **PERRIE**: Parity error interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A SPDIFRX interface interrupt is generated whenever PERR=1 in the SPDIFRX_SR register

Bit 1 **CSRNEIE: Control Buffer Ready Interrupt Enable**
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A SPDIFRX interface interrupt is generated whenever CSRNE = 1 in the SPDIFRX_SR register.

Bit 0 **RXNEIE**: RXNE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A SPDIFRX interface interrupt is generated whenever RXNE=1 in the SPDIFRX_SR register

51.5.3 Status register (SPDIFRX_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WIDTH5[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
							r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **WIDTH5[14:0]**: Duration of 5 symbols counted with spdifrx_ker_ck

This value represents the amount of spdifrx_ker_ck clock periods contained on a length of 5 consecutive symbols. This value can be used to estimate the S/PDIF symbol rate. Its accuracy is limited by the frequency of spdifrx_ker_ck.

For example if the spdifrx_ker_ck is fixed to 84 MHz, and WIDTH5 = 147d. The estimated sampling rate of the S/PDIF stream is:

Fs = 5 x $F_{spdifrx_ker_ck}$ / (WIDTH5 x 64) ~ 44.6 kHz, so the closest standard sampling rate is 44.1 kHz.

Note that WIDTH5 is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TERR**: Time-out error

This bit is set by hardware when the counter TRCNT reaches its max value. It indicates that the time interval between two transitions is too long. It generally indicates that there is no valid signal on SPDIFRX_IN input.

This flag is cleared by writing SPDIFRXEN to 0

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register

0: No sequence error is detected

1: Sequence error is detected

Bit 7 **SERR**: Synchronization error

This bit is set by hardware when the synchronization fails due to amount of re-tries for NBTR.

This flag is cleared by writing SPDIFRXEN to 0

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: No synchronization error is detected

1: Synchronization error is detected

Bit 6 **FERR**: Framing error

This bit is set by hardware when an error occurs during data reception: preamble not at the expected place, short transition not grouped by pairs...

This is set by the hardware only if the synchronization has been completed (SYNCD = 1).

This flag is cleared by writing SPDIFRXEN to 0

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: no Manchester Violation detected

1: Manchester Violation detected

Bit 5 **SYNCD**: Synchronization Done
This bit is set by hardware when the initial synchronization phase is properly completed.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_CLR_SR register.
An interrupt is generated if SYNCDIE = 1 in the SPDIFRX_IMR register
0: Synchronization is pending
1: Synchronization is completed

Bit 4 **SBD**: Synchronization Block Detected
This bit is set by hardware when a “B” preamble is detected
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_CLR_SR register.
An interrupt is generated if SBLKIE = 1 in the SPDIFRX_IMR register
0: No “B” preamble detected
1: “B” preamble has been detected

Bit 3 **OVR**: Overrun error
This bit is set by hardware when a received data is ready to be transferred in the SPDIFRX_FMTx_DR register while RXNE = 1 and both SPDIFRX_FMTx_DR and RX_BUF are full.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_CLR_SR register.
An interrupt is generated if OVRIE=1 in the SPDIFRX_IMR register.
0: No Overrun error
1: Overrun error is detected
Note: When this bit is set, the SPDIFRX_FMTx_DR register content will not be lost but the last data received will.

Bit 2 **PERR**: Parity error
This bit is set by hardware when the data and status bits of the sub-frame received contain an odd number of 0 and 1.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_CLR_SR register.
An interrupt is generated if PIE = 1 in the SPDIFRX_IMR register.
0: No parity error
1: Parity error

Bit 1 **CSRNE**: The Control Buffer register is not empty
This bit is set by hardware when a valid control information is ready.
This flag is cleared when reading SPDIFRX_CSR register.
An interrupt is generated if CBRDYIE = 1 in the SPDIFRX_IMR register
0: No control word available on SPDIFRX_CSR register
1: A control word is available on SPDIFRX_CSR register

Bit 0 **RXNE**: Read data register not empty
This bit is set by hardware when a valid data is available into SPDIFRX_FMTx_DR register.
This flag is cleared by reading the SPDIFRX_FMTx_DR register.
An interrupt is generated if RXNEIE=1 in the SPDIFRX_IMR register.
0: Data is not received
1: Received data is ready to be read.

51.5.4 Interrupt flag clear register (SPDIFRX_IFCR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCD CF	SBD CF	OVR CF	PERR CF	Res.	Res.
										w	w	w	w		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **SYNCDCF**: Clears the Synchronization Done flag
Writing 1 in this bit clears the flag SYNCD in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 4 **SBDCF**: Clears the Synchronization Block Detected flag
Writing 1 in this bit clears the flag SBD in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 3 **OVRCF**: Clears the Overrun error flag
Writing 1 in this bit clears the flag OVR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 2 **PERRCF**: Clears the Parity error flag
Writing 1 in this bit clears the flag PERR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bits 1:0 Reserved, must be kept at reset value.

51.5.5 Data input register (SPDIFRX_FMT0_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT. Here is the format when DRFMT = 00:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PT[1:0]		C	U	V	PE	DR[23:16]							
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **PT[1:0]:** Preamble Type

These bits indicate the preamble received.

00: not used

01: Preamble B received

10: Preamble M received

11: Preamble W received

Note that if PTMSK = 1, this field is forced to zero

Bit 27 **C**: Channel Status bit

Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 26 **U**: User bit

Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 25 **V**: Validity bit

Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 24 **PE**: Parity Error bit

Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

Bits 23:0 **DR[23:0]**: Data value

Contains the 24 received data bits, aligned on D[23]

51.5.6 Data input register (SPDIFRX_FMT1_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT. Here is the format when DRFMT = 01:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[7:0]								Res.	Res.	PT[1:0]		C	U	V	PE
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:8 **DR[23:0]**: Data value
Contains the 24 received data bits, aligned on D[23]

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PT[1:0]:** Preamble Type
These bits indicate the preamble received.
00: not used
01: Preamble B received
10: Preamble M received
11: Preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 3 **C**: Channel Status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 2 **U**: User bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 1 **V**: Validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 0 **PE**: Parity Error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

51.5.7 Data input register (SPDIFRX_FMT2_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT.

The data format proposed when DRFMT = 10, is dedicated to non-linear mode, as only 16 bits are used (bits 23 to 8 from S/PDIF sub-frame).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DRNL2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRNL1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **DRNL2[15:0]**: Data value
This field contains the Channel A

Bits 15:0 **DRNL1[15:0]**: Data value
This field contains the Channel B

51.5.8 Channel status register (SPDIFRX_CSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]							
							r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SOB**: Start Of Block
This bit indicates if the bit CS[0] corresponds to the first bit of a new block
0: CS[0] is not the first bit of a new block
1: CS[0] is the first bit of a new block

Bits 23:16 **CS[7:0]**: Channel A status information
Bit CS[0] is the oldest value

Bits 15:0 **USR[15:0]**: User data information
Bit USR[0] is the oldest value, and comes from channel A, USR[1] comes channel B.
So USR[n] bits come from channel A is n is even, otherwise they come from channel B.

51.5.9 Debug information register (SPDIFRX_DIR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TLO[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	THI[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **TLO[12:0]**: Threshold LOW (TLO = 1.5 x UI / $T_{spdifrx_ker_ck}$)
This field contains the current threshold LOW estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of TLO is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:
Sampling Rate = [2 x TLO x $T_{spdifrx_ker_ck}$ +/- $T_{spdifrx_ker_ck}$] x 2/3
Note that TLO is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **THI[12:0]**: Threshold HIGH (THI = 2.5 x UI / $T_{spdifrx_ker_ck}$)
This field contains the current threshold HIGH estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THI is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:
Sampling Rate = [2 x THI x $T_{spdifrx_ker_ck}$ +/- $T_{spdifrx_ker_ck}$] x 2/5
Note that THI is updated by the hardware when SYNCD goes high, and then every frame.

51.5.10 SPDIFRX interface register map

Table 423 gives the SPDIFRX interface register map and reset values.

Table 423. SPDIFRX interface register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPDIFRX_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSEL[2:0]			Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPDIFRX_IMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEIE	SYNCDIE	SBLKIE	OVRIE	PERRIE	CSRNEIE	RXNEIE
	Reset value																										0	0	0	0	0	0	0
0x08	**SPDIFRX_SR**	Res.	WIDTH5[14:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x0C	**SPDIFRX_IFCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCDCF	SBDCF	OVRCF	PERRCF	Res.	Res.
	Reset value																											0	0	0	0		
0x10	**SPDIFRX_FMT0_DR**	Res.	Res.	PT[1:0]		C	U	V	PE	DR[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPDIFRX_FMT1_DR**	DR[23:0]																								Res.	PT[1:0]		C	U	V	PE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPDIFRX_FMT2_DR**	DRNL2[15:0]																DRNL1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**SPDIFRX_CSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]								USR[15:0]															
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPDIFRX_DIR**	Res.	Res.	Res.	TLO[12:0]													Res.	Res.	Res.	THI[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

52 Single Wire Protocol Master Interface (SWPMI)

52.1 Introduction

The Single Wire Protocol Master Interface (SWPMI) is the master interface corresponding to the Contactless front-end (CLF) defined in the ETSI TS 102 613 technical specification.

The principle of the Single wire protocol (SWP) is based on the transmission of digital information in full duplex mode:

- S1 signal (from Master to Slave) is transmitted by a digital modulation (L or H) in the voltage domain (refer to *Figure 680: S1 signal coding*),
- S2 signal (from Slave to Master) is transmitted by a digital modulation (L or H) in the current domain (refer to *Figure 681: S2 signal coding*).

Figure 680. S1 signal coding

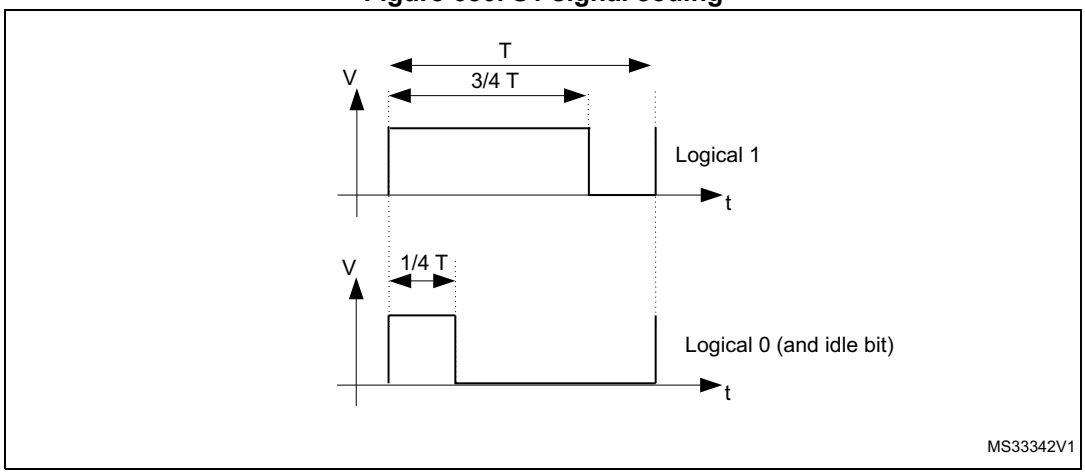
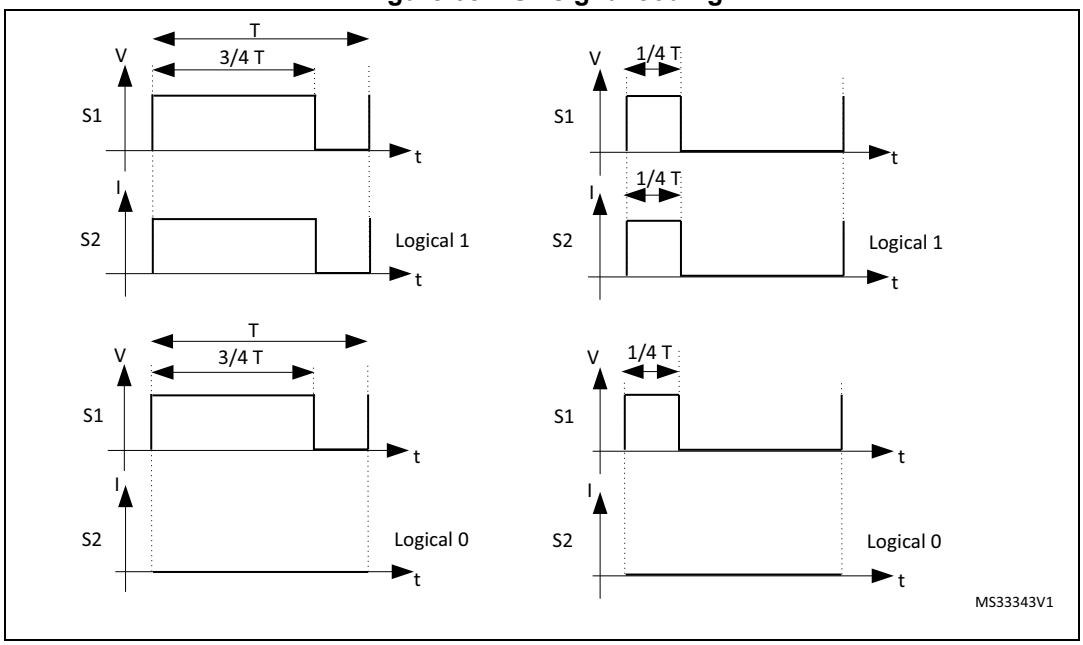


Figure 681. S2 signal coding



52.2 SWPMI main features

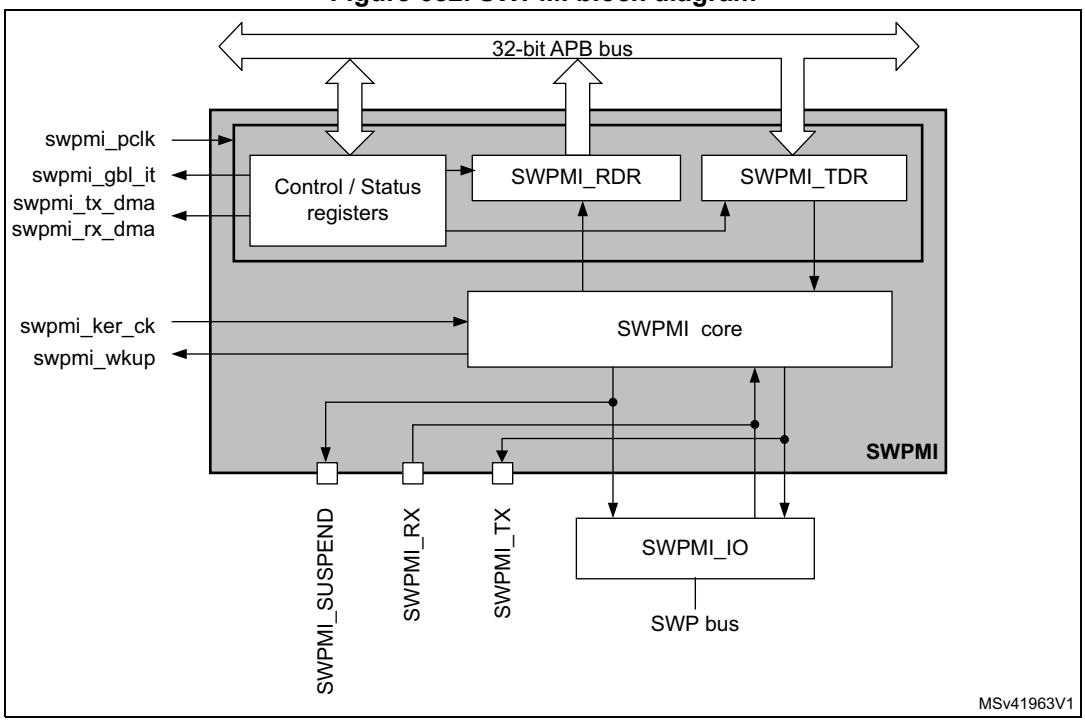
The SWPMI module main features are the following (see *Figure 52.3.4: SWP bus states*):

- Full-duplex communication mode
- Automatic SWP bus state management
- Automatic handling of Start of frame (SOF)
- Automatic handling of End of frame (EOF)
- Automatic handling of stuffing bits
- Automatic CRC-16 calculation and generation in transmission
- Automatic CRC-16 calculation and checking in reception
- 32-bit Transmit data register
- 32-bit Receive data register
- Multi software buffer mode for efficient DMA implementation and multi frame buffering
- Configurable bit-rate up to 2 Mbit/s
- Configurable interrupts
- CRC error, underrun, overrun flags
- Frame reception and transmission complete flags
- Slave resume detection flag
- Loopback mode for test purpose
- Embedded SWPMI_IO transceiver compliant with ETSI TS 102 613 technical specification
- Dedicated mode to output SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals on GPIOs, in case of external transceiver connection

52.3 SWPMI functional description

52.3.1 SWPMI block diagram

Figure 682. SWPMI block diagram



Refer to the bit SWPSRC in *Section 7.7.18: RCC Domain 2 Kernel Clock Configuration Register (RCC_D2CCIP1R)* to select the swpmi_ker_ck (SWPMI core clock source).

Note: In order to support the exit from Stop mode by a RESUME by slave, it is mandatory to select HSI for swpmi_ker_ck. If this feature is not required, swpmi_pclk can be selected, and SWPMI must be disabled before entering the Stop mode.

52.3.2 SWPMI pins and internal signals

Table 424 lists the SWPMI slave inputs and output signals connected to package pins or balls, while *Table 425* shows the internal SWPMI signals.

Table 424. SWPMI input/output signals connected to package pins or balls

Signal name	Signal type	Description
SWPMI_SUSPEND	Digital output	SWPMI suspend signal
SWPMI_TX	Digital output	SWPMI transmit signal
SWPMI_RX	Digital input	SWPMI receive signal

Table 425. SWPMI internal input/output signals

Signal name	Signal type	Description
swpmi_pclk	Digital input	APB clock
swpmi_ker_ck	Digital input	SWPMI kernel clock
swpmi_wkup	Digital output	SWPMI wakeup signal
swpmi_gbl_it	Digital output	SWPMI interrupt signal
swpmi_tx_dma	Digital output	SWPMI DMA transmit request
swpmi_rx_dma	Digital output	SWPMI DMA receive request

52.3.3 SWP initialization and activation

The initialization and activation will set the SWPMI_IO state from low to high.

When using the internal transceiver, the procedure is the following:

1. Configure the SWP_CLASS bit in SWPMI_OR register according to the VDD voltage (3 V or 1.8 V),
2. Set SWPTEN in SWPMI_CR register to enable the SWPMI_IO transceiver and set the SWPMI_IO to low level (SWP bus DEACTIVATED)
3. Wait for the RDYF flag in SWPMI_SR register to be set (polling the flag or enabling the interrupt with RDYIE bit in SWPMI_IER register),
4. Set SWPACT bit in SWPMI_CR register to ACTIVATE the SWP i.e. to move from DEACTIVATED to SUSPENDED.

52.3.4 SWP bus states

The SWP bus can have the following states: DEACTIVATED, SUSPENDED, ACTIVATED.

Several transitions are possible:

- ACTIVATE: transition from DEACTIVATED to SUSPENDED state,
- SUSPEND: transition from ACTIVATED to SUSPENDED state,
- RESUME by master: transition from SUSPENDED to ACTIVATED state initiated by the master,
- RESUME by slave: transition from SUSPENDED to ACTIVATED state initiated by the slave,
- DEACTIVATE: transition from SUSPENDED to DEACTIVATED state.

ACTIVATE

During and just after reset, the SWPMI_IO is configured in analog mode. Refer to *Section 52.3.3: SWP initialization and activation* to activate the SWP bus.

SUSPEND

The SWP bus stays in the ACTIVATED state as long as there is a communication with the slave, either in transmission or in reception. The SWP bus switches back to the SUSPENDED state as soon as there is no more transmission or reception activity, after 7 idle bits.

RESUME by master

Once the SWPMI is enabled, the user can request a SWPMI frame transmission. The SWPMI first sends a transition sequence and 8 idle bits (RESUME by master) before starting the frame transmission. The SWP moves from the SUSPENDED to ACTIVATED state after the RESUME by master (refer to *Figure 683: SWP bus states*).

RESUME by slave

Once the SWPMI is enabled, the SWP can also move from the SUSPENDED to ACTIVATED state if the SWPMI receives a RESUME from the slave. The RESUME by slave sets the SRF flag in the SWPMI_ISR register.

DEACTIVATE

Deactivate request

If no more communication is required, and if SWP is in the SUSPENDED mode, the user can request to switch the SWP to the DEACTIVATED mode by disabling the SWPMI peripheral. The software must set DEACT bit in the SWPMI_CR register in order to request the DEACTIVATED mode. If no RESUME by slave is detected by SWPMI, the DEACTF flag is set in the SWPMI_ISR register and the SWPACT bit is cleared in the SWPMI_ICR register. In case a RESUME by slave is detected by the SWPMI while the software is setting DEACT bit, the SRF flag is set in the SWPMI_ISR register, DEACTF is kept cleared, SWPACT is kept set and DEACT bit is cleared.

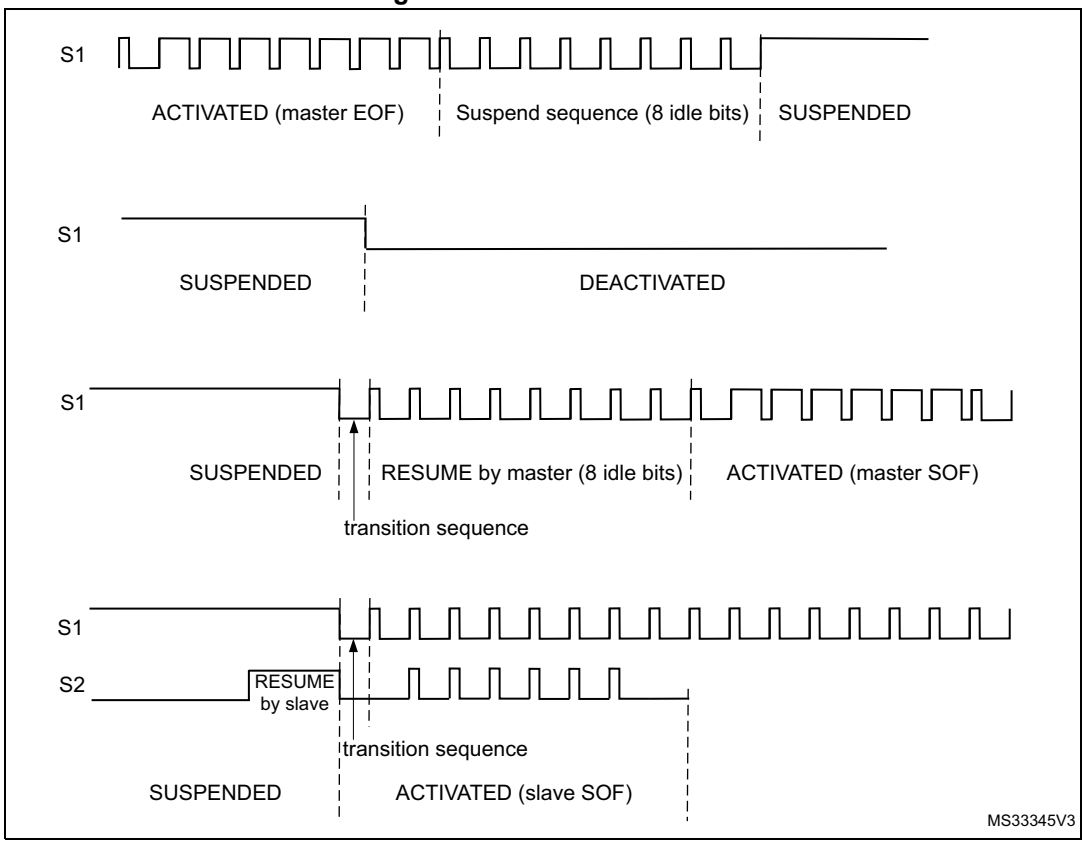
In order to activate SWP again, the software must clear DEACT bit in the SWPMI_CR register before setting SWPACT bit.

Deactivate mode

In order to switch the SWP to the DEACTIVATED mode immediately, ignoring any possible incoming RESUME by slave, the user must clear SWPACT bit in the SWPMI_CR register.

Note: In order to further reduce current consumption, configure the SWPMI_IO port as output push pull low in GPIO controller and then clear the SWPTEN bit in SWPMI_CR register (refer to Section 10: General-purpose I/Os (GPIO)).

Figure 683. SWP bus states



52.3.5 SWPMI_IO (internal transceiver) bypass

A SWPMI_IO (transceiver), compliant with ETSI TS 102 613 technical specification, is embedded in the microcontroller. Nevertheless, this is possible to bypass it by setting SWP_TBYP bit in SWPMI_OR register. In this case, the SWPMI_IO is disabled and the SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals are available as alternate functions on three GPIOs (refer to “Pinouts and pin description” in product datasheet). This configuration is selected to connect an external transceiver.

Note: In SWPMI_IO bypass mode, SWPTEN bit in SWPMI_CR register must be kept cleared.

52.3.6 SWPMI Bit rate

The bit rate must be set in the SWPMI_BRR register, according to the following formula:

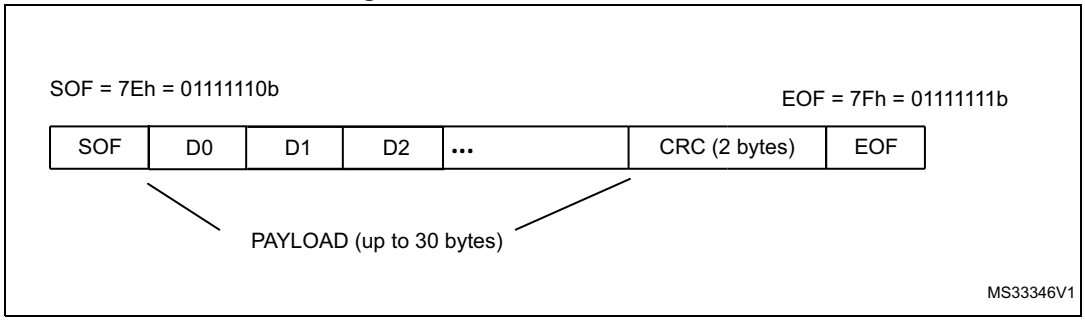
$F_{SWP} = F_{swpmi_ker_ck} / ((BR[7:0]+1)x4)$

Note: The maximum bitrate is 2 Mbit/s.

52.3.7 SWPMI frame handling

The SWP frame is composed of a Start of frame (SOF), a Payload from 1 to 30 bytes, a 16-bit CRC and an End of frame (EOF) (Refer to *Figure 684: SWP frame structure*).

Figure 684. SWP frame structure



The SWPMI embeds one 32-bit data register for transmission (SWPMI_TDR), and one 32-bit data register for reception (SWPMI_RDR).

In transmission, the SOF insertion, the CRC calculation and insertion, and the EOF insertion are managed automatically by the SWPMI. The user only has to provide the Payload content and size. A frame transmission starts as soon as data is written into the SWPMI_TDR register. Dedicated flags indicate an empty transmit data register and a complete frame transmission event.

In reception, the SOF deletion, the CRC calculation and checking, and the EOF deletion are managed automatically by the SWPMI. The user only has to read the Payload content and size. Dedicated flags indicate a full receive data register, a complete frame reception and possibly CRC error events.

The stuffing bits insertion (in transmission) and stuffing bits deletion (in reception) are managed automatically by the SWPMI core. These operations are transparent for the user.

52.3.8 Transmission procedure

Before starting any frame transmission, the user must activate the SWP. Refer to *Section 52.3.3: SWP initialization and activation*.

There are several possible software implementations for a frame transmission: No software buffer mode, Single software buffer mode, and Multi software buffer mode.

The software buffer usage requires the use of a DMA channel to transfer data from the software buffer in the RAM memory to the transmit data register in the SWPMI peripheral.

No software buffer mode

This mode does not require the use of DMA. The SWP frame transmission handling is done by polling status flags in the main loop or inside the SWPMI interrupt routine. There is a 32-bit transmit data register (SWPMI_TDR) in the SWPMI, thus writing to this register will trigger the transmission of up to 4 bytes.

The No software buffer mode is selected by clearing TXDMA bit in the SWPMI_CR register.

The frame transmission is started by the first write to the SWPMI_TDR register. The low significant byte of the first 32-bit word (bits [7:0]) written into the SWPMI_TDR register) indicates the number of data bytes in the payload, and the 3 other bytes of this word must

contain the first 3 bytes of the payload (bits [15:8] contain the first byte of the payload, bits [23:16] the second byte and bits [31:24] the third byte). Then, the following writes to the SWPMI_TDR register will only contain the following payload data bytes, up to 4 for each write.

Note: The low significant byte of the first 32-bit word written into the SWPMI_TDR register is coding the number of data bytes in the payload. This number could be from 1 to 30. Any other value in the low significant byte will be ignored and the transmission will not start.

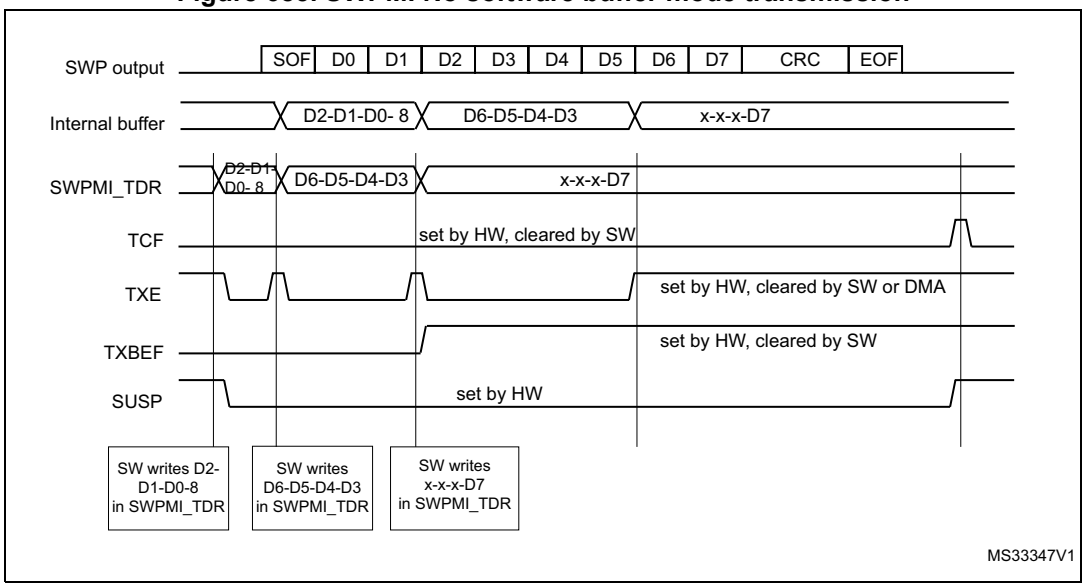
Writing to the SWPMI_TDR register will induce the following actions:

- Send the transition sequence and 8 idle bits (RESUME by master) if the SWP bus state is SUPENDED (this will not happen if the SWP bus state is already ACTIVATED),
- Send a Start of frame (SOF),
- Send the payload according to the SWPMI_TRD register content. If the number of bytes in the payload is greater than 3, the SWPMI_TDR needs to be refilled by software, each time the TXE flag in the SWPMI_ISR register is set, and as long as the TXBEF flag is not set in the SWPMI_ISR register,
- Send the 16-bit CRC, automatically calculated by the SWPMI core,
- Send an End of frame (EOF).

The TXE flag is cleared automatically when the software is writing to the SWPMI_TDR register.

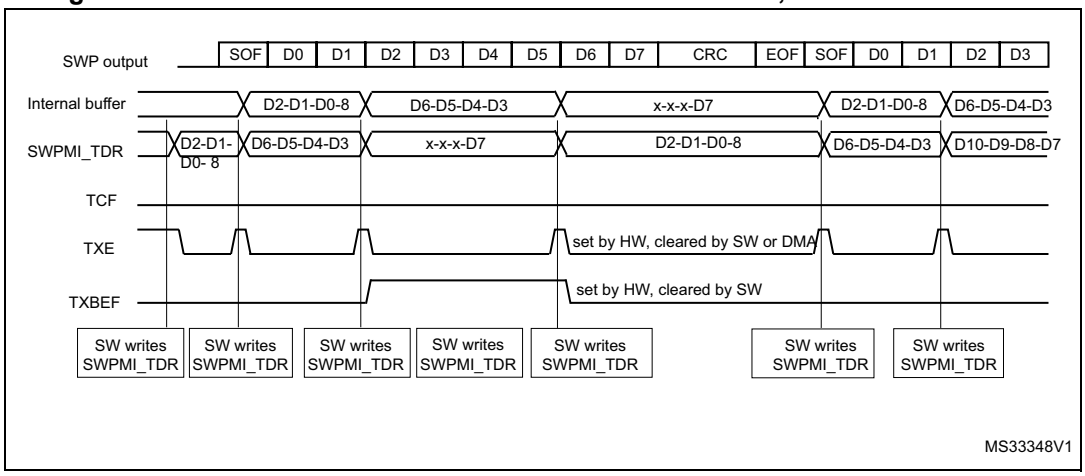
Once the complete frame is sent, provided that no other frame transmission has been requested (i.e. SWPMI_TDR has not been written again after the TXBEF flag setting), TCF and SUSP flags are set in the SWPMI_ISR register 7 idle bits after the EOF transmission, and an interrupt is generated if TCIE bit is set in the SWPMI_IER register (refer to *Figure 685: SWPMI No software buffer mode transmission*).

Figure 685. SWPMI No software buffer mode transmission



If another frame transmission is requested before the end of the EOF transmission, the TCF flag is not set and the frame will be consecutive to the previous one, with only one idle bit in between (refer to *Figure 686: SWPMI No software buffer mode transmission, consecutive frames*).

Figure 686. SWPMI No software buffer mode transmission, consecutive frames



Single software buffer mode

This mode allows to transmit a complete SWP frame without a CPU intervention, using the DMA. The DMA will refill the 32-bit SWPMI_TDR register, and the software can poll the end of the frame transmission using the SWPMI_TXBEF flag.

The Single software buffer mode is selected by setting TXDMA bit and clearing TXMODE bit in the SWPMI_CR register.

The DMA channel or stream must be configured in following mode (refer to DMA section):

- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode disabled,
- data transfer direction set to read from memory.
- the number of words to be transfered must be set according to the SWP frame length,
- the source address is the SWP frame buffer in RAM,
- the destination address is the SWPMI_TDR register.

Then the user must:

1. Set TXDMA bit in the SWPMI_CR register,
2. Set TXBEIE bit in the SWPMI_IER register,
3. Fill the buffer in the RAM memory (with the number of data bytes in the payload on the least significant byte of the first word),
4. Enable stream or channel in DMA module to start DMA transfer and frame transmission.

A DMA request is issued by SWPMI when TXE flag in SWPMI_ISR is set. The TXE flag is cleared automatically when the DMA is writing to the SWPMI_TDR register.

In the SWPMI interrupt routine, the user must check TXBEF bit in the SWPMI_ISR register. If it is set, and if another frame needs to be transmitted, the user must:

1. Disable stream or channel in DMA module
2. Update the buffer in the RAM memory with the content of the next frame to be sent
3. Configure the total number of words to be transferred in DMA module
4. Enable stream or channel in DMA module to start next frame transmission
5. Set CTXBEF bit in the SWPMI_ICR register to clear the TXBEF flag

Multi software buffer mode

This mode allows to work with several frame buffers in the RAM memory, in order to ensure a continuous transmission, keeping a very low CPU load, and allowing more latency for buffer update by software thanks to the DMA. The software can check the DMA counters at any time and update SWP frames accordingly in the RAM memory.

The Multi software buffer mode must be used in combination with DMA in circular mode.

Each transmission buffer in the RAM memory must have a fixed length of eight 32-bit words, whatever the number of bytes in the SWP frame payload. The transmission buffers in the RAM memory must be filled by the software, keeping an offset of 8 between two consecutive ones. The first data byte of the buffer is the number of bytes of the frame payload. See the buffer example in *Figure 687: SWPMI Multi software buffer mode transmission*

The Multi software buffer mode is selected by setting both TXDMA and TXMODE bits in SWPMI_CR register.

For example, in order to work with 4 transmission buffers, the user must configure the DMA as follows:

The DMA channel or stream must be configured in following mode (refer to DMA section):

- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode enabled,
- data transfer direction set to read from memory,
- the number of words to be transfered must be set to 32 (8 words per buffer),
- the source address is the buffer1 in RAM,
- the destination address is the SWPMI_TDR register.

Then, the user must:

1. Set TXDMA in the SWPMI_CR register
2. Set TXBEIE in the SWPMI_IER register
3. Fill buffer1, buffer2, buffer3 and buffer4 in the RAM memory (with the number of data bytes in the payload on the least significant byte of the first word)
4. Enable stream or channel in DMA module to start DMA.

In the SWPMI interrupt routine, the user must check TXBEF bit in the SWPMI_ISR register. If it is set, the user must set CTXBEF bit in SWPMI_ICR register to clear TXBEF flag and the user can update buffer1 in the RAM memory.

In the next SWPMI interrupt routine occurrence, the user will update buffer2, and so on.

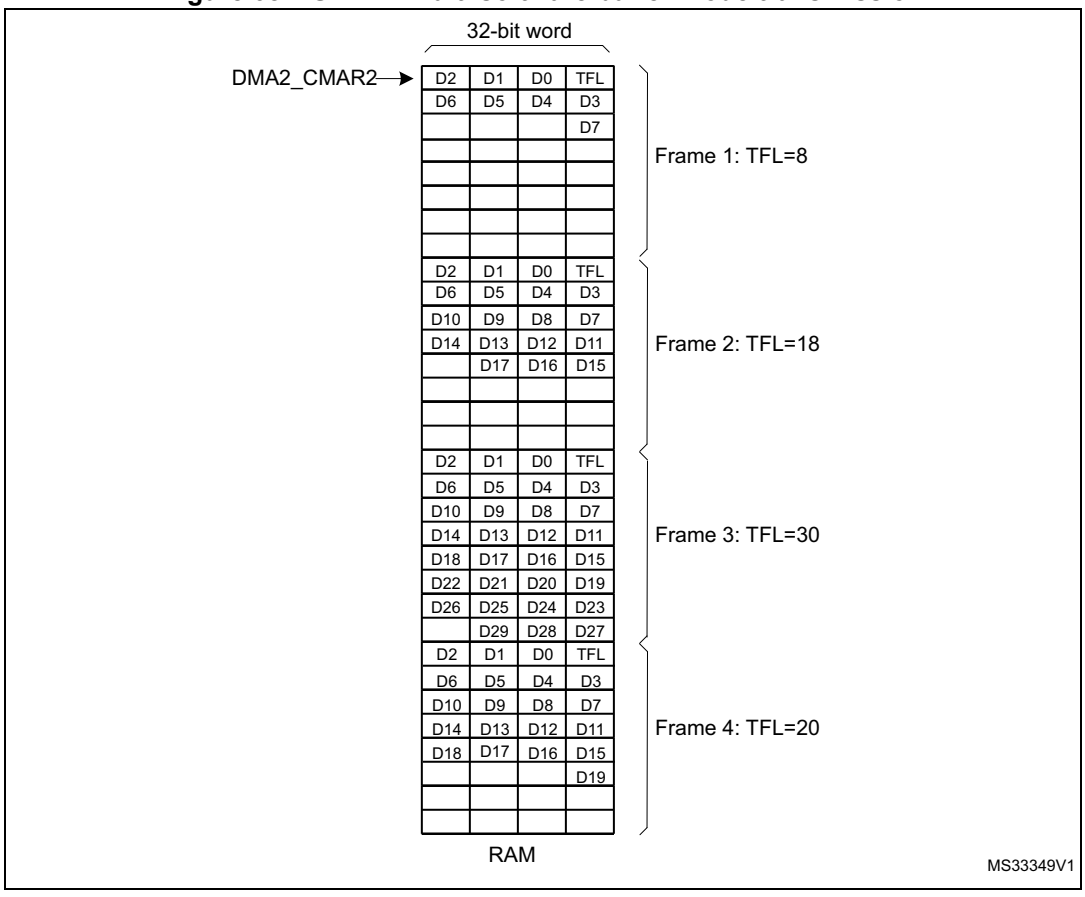
The Software can also read the DMA counter (number of data to transfer) in the DMA registers in order to retrieve the frame which has already been transferred from the RAM memory and transmitted. For example, if the software works with 4 transmission buffers, and if the DMA counter equals 17, it means that two buffers are ready for updating in the RAM area. This is useful in case several frames are sent before the software can handle the SWPMI interrupt. If this happens, the software will have to update several buffers.

When there are no more frames to transmit, the user must disable the circular mode in the DMA module. The transmission will stop at the end of the buffer4 transmission.

If the transmission needs to stop before (for example at the end of buffer2), the user must set the low significant byte of the first word to 0 in buffer3 and buffer4.

TXDMA bit in the SWPMI_CR register will be cleared by hardware as soon as the number of data bytes in the payload is read as 0 in the least significant byte of the first word.

Figure 687. SWPMI Multi software buffer mode transmission



52.3.9 Reception procedure

Before starting any frame reception, the user must activate the SWP (refer to *Section 52.3.3: SWP initialization and activation*.

Once SWPACT bit is set in the SWPMI_CR register, a RESUME from slave state sets the SRF flag in the SWPMI_ISR register and automatically enables the SWPMI for the frame reception.

If the SWP bus is already in the ACTIVATED state (for example because a frame transmission is ongoing), the SWPMI core does not need any RESUME by slave state, and the reception can take place immediately.

There are several possible software implementations for a frame reception:

- No software buffer mode,
- Single software buffer mode,
- Multi software buffer mode.

The software buffer usage requires the use of a DMA channel to transfer data from the receive data register in the SWPMI peripheral to the software buffer in the RAM memory.

No software buffer mode

This mode does not require the use of DMA. The SWP frame reception handling is done by polling status flags in the main loop or inside the SWPMI interrupt routine. There is a 32-bit receive data register (SWPMI_RDR) in the SWPMI, allowing to receive up to 4 bytes before reading this register.

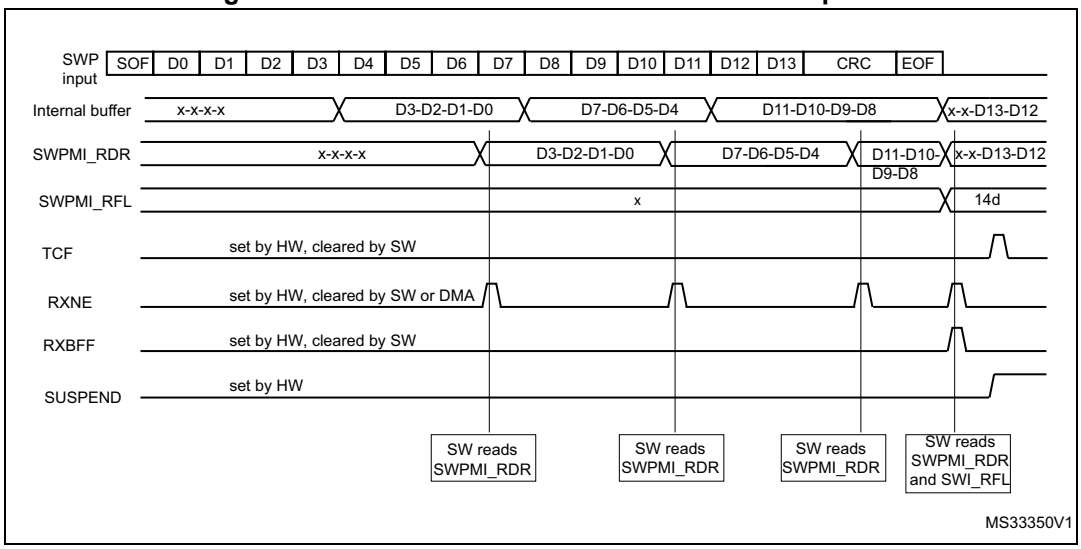
The No software buffer mode is selected by resetting RXDMA bit in the SWPMI_CR register.

Once a Start of frame (SOF) is received, the following bytes (payload) are stored in the SWPMI_RDR register. Once the SWPMI_RDR is full, the RXNE flag is set in SWPMI_ISR and an interrupt is generated if RIE bit is set in SWPMI_IER register. The user can read the SWPMI_RDR register and the RXNE flag is cleared automatically when the software is reading the SWPMI_RDR register.

Once the complete frame has been received, including the CRC and the End of frame (EOF), both RXNE and RXBFF flags are set in the SWPMI_ISR register. The user must read the last byte(s) of the payload in the SWPMI_RDR register and set CRXBFF flag in SWPMI_ICR in order to clear the RXBFF flag. The number of data bytes in the payload is available in the SWPMI_RFL register. Again, the RXNE flag is reset automatically when the software is reading the SWPMI_RDR register (refer to *Figure 688: SWPMI No software buffer mode reception*).

Reading the SWPMI_RDR register while RXNE is cleared will return 0.

Figure 688. SWPMI No software buffer mode reception



Single software buffer mode

This mode allows to receive a complete SWP frame without any CPU intervention using the DMA. The DMA transfers received data from the 32-bit SWPMI_RDR register to the RAM memory, and the software can poll the end of the frame reception using the SWPMI_RBFF flag.

The Single software buffer mode is selected by setting RXDMA bit and clearing RXMODE bit in the SWPMI_CR register.

The DMA must be configured as follows:

The DMA channel or stream must be configured in following mode (refer to DMA section):

- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode disabled,
- data transfer direction set to read from peripheral,
- the number of words to be transfered must be set to 8,
- the source address is the SWPMI_RDR register,
- the destination address is the SWP frame buffer in RAM.

Then the user must:

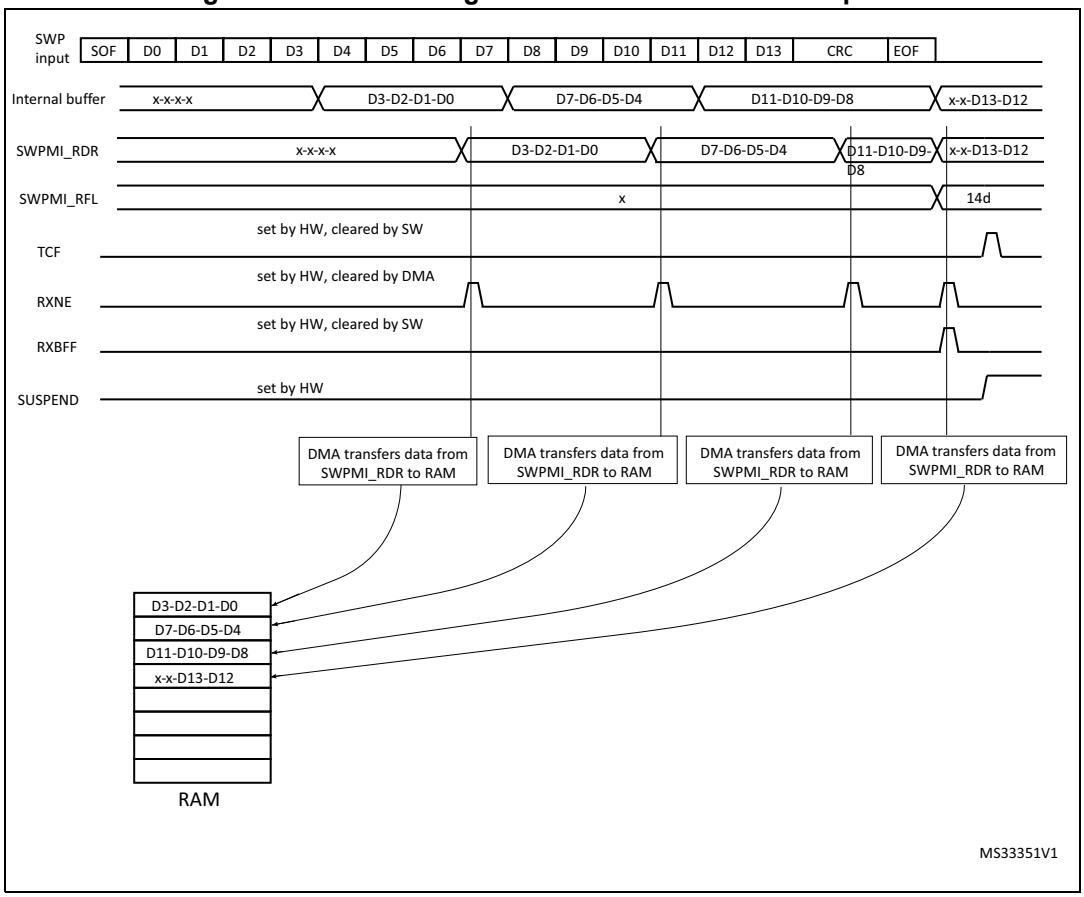
1. Set RXDMA bit in the SWPMI_CR register
2. Set RXBFIE bit in the SWPMI_IER register
3. Enable stream or channel in DMA module.

A DMA request is issued by SWPMI when RXNE flag is set in SWPMI_ISR. The RXNE flag is cleared automatically when the DMA is reading the SWPMI_RDR register.

In the SWPMI interrupt routine, the user must check RXBFF bit in the SWPMI_ISR register. If it is set, the user must:

1. Disable stream or channel in DMA module
2. Read the number of bytes in the received frame payload in the SWPMI_RFL register
3. Read the frame payload in the RAM buffer
4. Enable stream or channel in DMA module
5. Set CRXBFF bit in the SWPMI_ICR register to clear RXBFF flag (refer to *Figure 689: SWPMI single software buffer mode reception*).

Figure 689. SWPMI single software buffer mode reception



Multi software buffer mode

This mode allows to work with several frame buffers in the RAM memory, in order to ensure a continuous reception, keeping a very low CPU load, using the DMA. The frame payloads are stored in the RAM memory, together with the frame status flags. The software can check the DMA counters and status flags at any time to handle the received SWP frames in the RAM memory.

The Multi software buffer mode must be used in combination with the DMA in circular mode.

The Multi software buffer mode is selected by setting both RXDMA and RXMODE bits in SWPMI_CR register.

In order to work with n reception buffers in RAM, the DMA channel or stream must be configured in following mode (refer to DMA section):

- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode enabled,
- data transfer direction set to read from peripheral,
- the number of words to be transfered must be set to 8 x n (8 words per buffer),
- the source address is the SWPMI_TDR register,
- the destination address is the buffer1 address in RAM

Then the user must:

1. Set RXDMA in the SWPMI_CR register
2. Set RXBFIE in the SWPMI_IER register
3. Enable stream or channel in the DMA module.

In the SWPMI interrupt routine, the user must check RXBFF in the SWPMI_ISR register. If it is set, the user must set CRXBFF bit in the SWPMI_ICR register to clear RXBFF flag and the user can read the first frame payload received in the first buffer (at the RAM address set in DMA2_CMAR1).

The number of data bytes in the payload is available in bits [23:16] of the last 8th word.

In the next SWPMI interrupt routine occurrence, the user will read the second frame received in the second buffer (address set in DMA2_CMAR1 + 8), and so on (refer to *Figure 690: SWPMI Multi software buffer mode reception*).

In case the application software cannot ensure to handle the SMPMI interrupt before the next frame reception, each buffer status is available in the most significant byte of the 8th buffer word:

- The CRC error flag (equivalent to RXBERF flag in the SWPMI_ISR register) is available in bit 24 of the 8th word. Refer to *Section 52.3.10: Error management* for an CRC error description.
- The receive overrun flag (equivalent to RXOVRF flag in the SWPMI_ISR register) is available in bit 25 of the 8th word. Refer to *Section 52.3.10: Error management* for an overrun error description.
- The receive buffer full flag (equivalent to RXBFF flag in the SWPMI_ISR register) is available in bit 26 of the 8th word.

In case of a CRC error, both RXBFF and RXBERF flags are set, thus bit 24 and bit 26 are set.

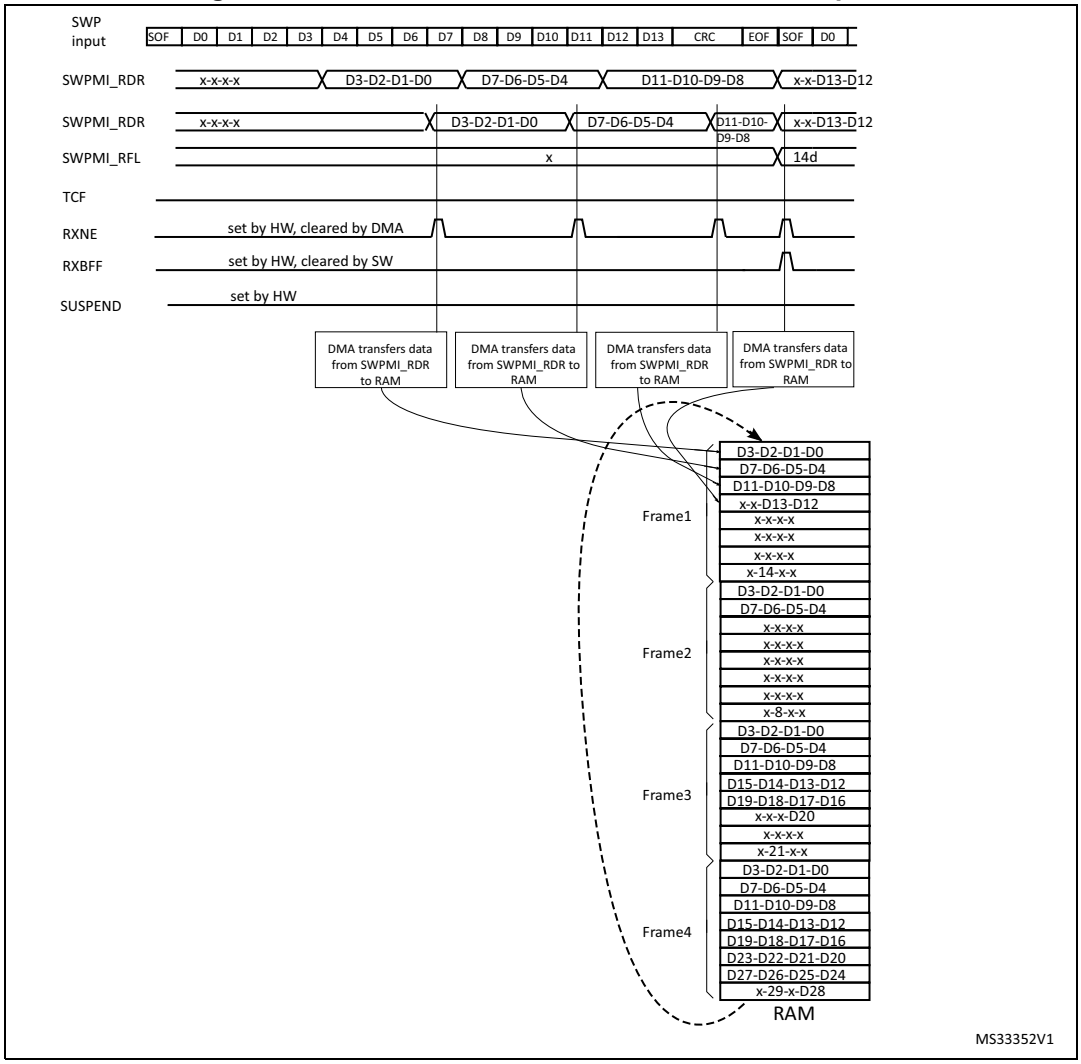
In case of an overrun, an overrun flag is set, thus bit 25 is set. The receive buffer full flag is set only in case of an overrun during the last word reception; then, both bit 25 and bit 26 are set for the current and the next frame reception.

The software can also read the DMA counter (number of data to transfer) in the DMA registers in order to retrieve the frame which has already been received and transferred into the RAM memory through DMA. For example, if the software works with 4 reception buffers,

and if the DMA counter equals 17, it means that two buffers are ready for reading in the RAM area.

In Multi software buffer reception mode, if the software is reading bits 24, 25 and 26 of the 8th word, it does not need to clear RXBERF, RXOVRF and RXBFF flags after each frame reception.

Figure 690. SWPMI Multi software buffer mode reception



52.3.10 Error management

Underrun during payload transmission

During the transmission of the frame payload, a transmit underrun is indicated by the TXUNRF flag in the SWPMI_ISR register. An interrupt is generated if TXBUNREIE bit is set in the SWPMI_IER register.

If a transmit underrun occurs, the SWPMI stops the payload transmission and sends a corrupted CRC (the first bit of the first CRC byte sent is inverted), followed by an EOF. If DMA is used, TXDMA bit in the SWPMI_CR register is automatically cleared.

Any further write to the SWPMI_TDR register while TXUNRF is set will be ignored. The user must set CTXUNRF bit in the SWPMI_ICR register to clear TXUNRF flag.

Overrun during payload reception

During the reception of the frame payload, a receive overrun is indicated by RXOVRF flag in the SWPMI_ISR register. If a receive overrun occurs, the SWPMI does not update SWPMI_RDR with the incoming data. The incoming data will be lost.

The reception carries on up to the EOF and, if the overrun condition disappears, the RXBFF flag is set. When RXBFF flag is set, the user can check the RXOVRF flag. The user must set CRXOVRF bit in the SWPMI_ICR register to clear RXBOVRF flag.

If the user wants to detect the overrun immediately, RXBOVREIE bit in the SWPMI_IER register can be set in order to generate an interrupt as soon as the overrun occurs.

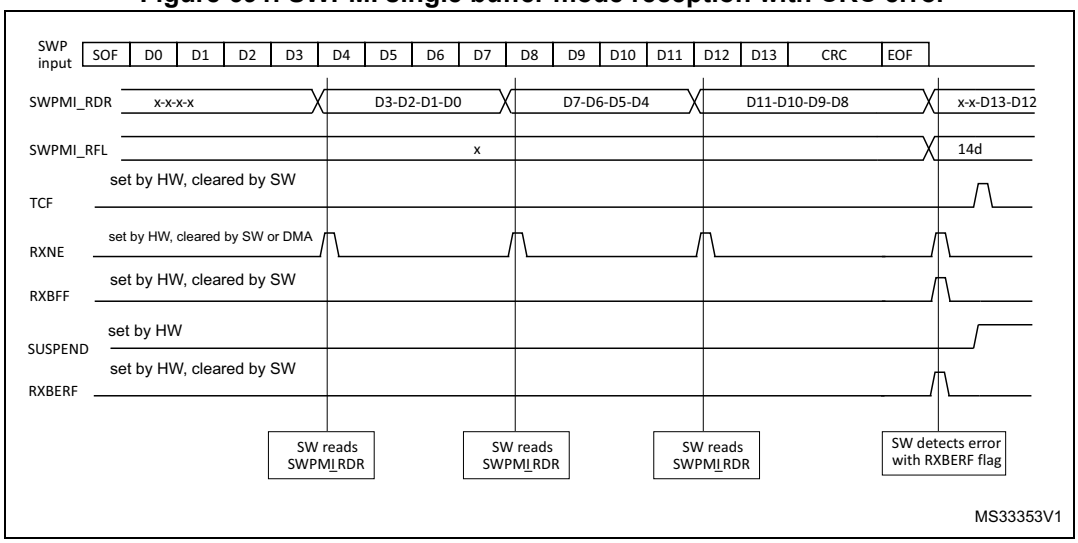
The RXOVRF flag is set at the same time as the RXNE flag, two SWPMI_RDR reads after the overrun event occurred. It indicates that at least one received byte was lost, and the loaded word in SWPMI_RDR contains the bytes received just before the overrun.

In Multi software buffer mode, if RXOVRF flag is set for the last word of the received frame, then the overrun bit (bit 25 of the 8th word) is set for both the current and the next frame.

CRC error during payload reception

Once the two CRC bytes have been received, if the CRC is wrong, the RXBERF flag in the SWPMI_ISR register is set after the EOF reception. An interrupt is generated if RXBEIE bit in the SWPMI_IER register is set (refer to*Figure 691: SWPMI single buffer mode reception with CRC error*).The user must set CRXBERF bit in SWPMI_ICR to clear RXBERF flag.

Figure 691. SWPMI single buffer mode reception with CRC error



Missing or corrupted stuffing bit during payload reception

When a stuffing bit is missing or is corrupted in the payload, RXBERF and RXBFF flags are set in SWPMI_ISR after the EOF reception.

Corrupted EOF reception

Once an SOF has been received, the SWPMI accumulates the received bytes until the reception of an EOF (ignoring any possible SOF). Once an EOF has been received, the SWPMI is ready to start a new frame reception and waits for an SOF.

In case of a corrupted EOF, RXBERF and RXBFF flags will bet set in the SWPMI_ISR register after the next EOF reception.

Note: In case of a corrupted EOF reception, the payload reception carries on, thus the number of bytes in the payload might get the value 31 if the number of received bytes is greater than 30. The number of bytes in the payload is read in the SWPMI_RFL register or in bits [23:16] of the 8th word of the buffer in the RAM memory, depending on the operating mode.

52.3.11 Loopback mode

The loopback mode can be used for test purposes. The user must set LPBK bit in the SWPMI_CR register in order to enable the loopback mode.

When the loopback mode is enabled, SWPMI_TX and SWPMI_RX signals are connected together. As a consequence, all frames sent by the SWPMI will be received back.

52.4 SWPMI low-power modes

Table 426. Effect of low-power modes on SWPMI

Mode	Description
Sleep	No effect. SWPMI interrupts cause the device to exit the Sleep mode.
Stop	A RESUME from SUSPENDED mode issued by the slave can wake up the device from Stop mode if the swpmi_ker_ck is HSI (refer to *Section 52.3.1: SWPMI block diagram*).
Standby	The SWPMI is stopped.

52.5 SWPMI interrupts

All SWPMI interrupts are connected to the NVIC.

To enable the SWPMI interrupt, the following sequence is required:

1. Configure and enable the SWPMI interrupt channel in the NVIC
2. Configure the SWPMI to generate SWPMI interrupts (refer to the SWPMI_IER register).

Table 427. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
Receive buffer full	RXBFF	RXBFIE	yes	no	no
Transmit buffer empty	TXBEF	TXBEIE	yes	no	no
Receive buffer error (CRC error)	RXBERF	RXBEIE	yes	no	no
Receive buffer overrun	RXOVRF	RXBOVEREIE	yes	no	no
Transmit buffer underrun	TXUNRF	TXBUNREIE	yes	no	no
Receive data register not empty	RXNE	RIE	yes	no	no
Transmit data register full	TXE	TIE	yes	no	no
Transfer complete flag	TCF	TCIE	yes	no	no
Slave resume flag	SRF	SRIE	yes	yes (1)	no
Transceiver ready flag	RDYF	RDYIE	yes	no	no

1. If HSI is selected for swpmi_ker_ck.

52.6 SWPMI registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

52.6.1 SWPMI Configuration/Control register (SWPMI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWP EN	DEACT	Res.	Res.	Res.	Res.	SWP ACT	LPBK	TXMOD E	RXMO DE	TXDMA	RXDMA
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWPTEN**: Single wire protocol master transceiver enable

This bit is used to enable the transceiver and control the SWPMI_IO with SWPMI (refer to *Section 52.3.3: SWP initialization and activation*).

0: SPWMI_IO pin is controlled by GPIO controller

1: SWPMI_IO transceiver is controlled by SWPMI

Bit 10 **DEACT**: Single wire protocol master interface deactivate

This bit is used to request the SWP DEACTIVATED state. Setting this bit has the same effect as clearing the SWPACT, except that a possible incoming RESUME by slave will keep the SWP in the ACTIVATED state.

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **SWPACT**: Single wire protocol master interface activate

This bit is used to activate the SWP bus (refer to *Section 52.3.3: SWP initialization and activation*).

0: SWPMI_IO is pulled down to ground, SWP bus is switched to DEACTIVATED state

1: SWPMI_IO is released, SWP bus is switched to SUSPENDED state

To be able to set SWPACT bit, DEACT bit must be have been cleared previously.

Bit 4 **LPBK**: Loopback mode enable

This bit is used to enable the loopback mode

0: Loopback mode is disabled

1: Loopback mode is enabled

Note: This bit cannot be written while SWPACT bit is set.

Bit 3 **TXMODE**: Transmission buffering mode

This bit is used to choose the transmission buffering mode. This bit is relevant only when TXDMA bit is set (refer to *Table 428: Buffer modes selection for transmission/reception*).

0: SWPMI is configured in Single software buffer mode for transmission

1: SWPMI is configured in Multi software buffer mode for transmission.

Note: This bit cannot be written while SWPACT bit is set.

Bit 2 **RXMODE**: Reception buffering mode

This bit is used to choose the reception buffering mode. This bit is relevant only when TXDMA bit is set (refer to *Table 428: Buffer modes selection for transmission/reception*).

0: SWPMI is configured in Single software buffer mode for reception

1: SWPMI is configured in Multi software buffer mode for reception.

Note: This bit cannot be written while SWPACT bit is set.

Bit 1 **TXDMA**: Transmission DMA enable

This bit is used to enable the DMA mode in transmission

0: DMA is disabled for transmission

1: DMA is enabled for transmission

Note: TXDMA is automatically cleared if the payload size of the transmitted frame is given as 0x00 (in the least significant byte of TDR for the first word of a frame). TXDMA is also automatically cleared on underrun events (when TXUNRF flag is set in the SWP_ISR register)

Bit 0 **RXDMA**: Reception DMA enable

This bit is used to enable the DMA mode in reception

0: DMA is disabled for reception

1: DMA is enabled for reception

Table 428. Buffer modes selection for transmission/reception

Buffer mode	No software buffer	Single software buffer	Multi software buffer
RXMODE/TXMODE	x	0	1
RXDMA/TXDMA	0	1	1

52.6.2 SWPMI Bitrate register (SWPMI_BRR)

Address offset: 0x04

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **BR[7:0]**: Bitrate prescaler

This field must be programmed to set SWP bus bitrate, taking into account the $F_{swpmi_ker_ck}$ programmed in the RCC (Reset and Clock Control), according to the following formula:

$F_{SWP} = F_{swpmi_ker_ck} / ((BR[7:0]+1)x4)$

Note: The programmed bitrate must stay within the following range: from 100 kbit/s up to 2 Mbit/s.

BR[7:0] cannot be written while SWPACT bit is set in the SWPMI_CR register.

52.6.3 SWPMI Interrupt and Status register (SWPMI_ISR)

Address offset: 0x0C

Reset value: 0x0000 02C2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RDYF	DEACTF	SUSP	SRF	TCF	TXE	RXNE	TXUNRF	RXOVRF	RXBERF	TXBEF	RXBFF
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RDYF**: transceiver ready flag

This bit is set by hardware as soon as transceiver is ready. After setting the SWPTEN bit in SWPMI_CR register to enable the SWPMI_IO transceiver, software must wait for this flag to be set before setting the SWPACT bit to activate the SWP bus.

0: transceiver not ready

1: transceiver ready

Bit 10 **DEACTF**: DEACTIVATED flag

This bit is a status flag, acknowledging the request to enter the DEACTIVATED mode.

0: SWP bus is in ACTIVATED or SUSPENDED state

1: SWP bus is in DEACTIVATED state

If a RESUME by slave state is detected by the SWPMI while DEACT bit is set by software, the SRF flag will be set, DEACTF will not be set and SWP will move in ACTIVATED state.

Bit 9 **SUSP**: SUSPEND flag

This bit is a status flag, reporting the SWP bus state

0: SWP bus is in ACTIVATED state

1: SWP bus is in SUSPENDED or DEACTIVATED state

Bit 8 **SRF**: Slave resume flag

This bit is set by hardware to indicate a RESUME by slave detection. It is cleared by software, writing 1 to CSRF bit in the SWPMI_ICR register.

0: No Resume by slave state detected

1: A Resume by slave state has been detected during the SWP bus SUSPENDED state

Bit 7 **TCF**: Transfer complete flag

This flag is set by hardware as soon as both transmission and reception are completed and SWP is switched to the SUSPENDED state. It is cleared by software, writing 1 to CTCF bit in the SWPMI_ICR register.

0: Transmission or reception is not completed

1: Both transmission and reception are completed and SWP is switched to the SUSPENDED state

Bit 6 **TXE**: Transmit data register empty

This flag indicates the transmit data register status

0: Data written in transmit data register SWPMI_TDR is not transmitted yet

1: Data written in transmit data register SWPMI_TDR has been transmitted and SWPMI_TDR can be written to again

Bit 5 **RXNE**: Receive data register not empty

This flag indicates the receive data register status

0: Data is not received in the SWPMI_RDR register

1: Received data is ready to be read in the SWPMI_RDR register

Bit 4 **TXUNRF**: Transmit underrun error flag

This flag is set by hardware to indicate an underrun during the payload transmission i.e. SWPMI_TDR has not been written in time by the software or the DMA. It is cleared by software, writing 1 to the CTXUNRF bit in the SWPMI_ICR register.

0: No underrun error in transmission

1: Underrun error in transmission detected

Bit 3 **RXOVRF**: Receive overrun error flag

This flag is set by hardware to indicate an overrun during the payload reception, i.e. SWPMI_RDR has not be read in time by the software or the DMA. It is cleared by software, writing 1 to CRXOVRF bit in the SWPMI_ICR register.

0: No overrun in reception

1: Overrun in reception detected

Bit 2 **RXBERF**: Receive CRC error flag

This flag is set by hardware to indicate a CRC error in the received frame. It is set synchronously with RXBFF flag. It is cleared by software, writing 1 to CRXBERF bit in the SWPMI_ICR register.

0: No CRC error in reception

1: CRC error in reception detected

Bit 1 **TXBEF**: Transmit buffer empty flag

This flag is set by hardware to indicate that no more SWPMI_TDR update is required to complete the current frame transmission. It is cleared by software, writing 1 to CTXBEF bit in the SWPMI_ICR register.

0: Frame transmission buffer no yet emptied

1: Frame transmission buffer has been emptied

Bit 0 **RXBFF**: Receive buffer full flag

This flag is set by hardware when the final word for the frame under reception is available in SWPMI_RDR. It is cleared by software, writing 1 to CRXBFF bit in the SWPMI_ICR register.

0: The last word of the frame under reception has not yet arrived in SWPMI_RDR

1: The last word of the frame under reception has arrived in SWPMI_RDR

52.6.4 SWPMI Interrupt Flag Clear register (SWPMI_ICR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CRDYF	Res.	Res.	CSRF	CTCF	Res.	Res.	CTXUNRF	CRXOVRF	CRXBERF	CTXBEF	CRXBFF
				rc_w1			rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **CRDYF** Clear transceiver ready flag
Writing 1 to this bit clears the RDYF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **CSRF**: Clear slave resume flag
Writing 1 to this bit clears the SRF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

Bit 7 **CTCF**: Clear transfer complete flag
Writing 1 to this bit clears the TCF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CTXUNRF**: Clear transmit underrun error flag
Writing 1 to this bit clears the TXUNRF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

Bit 3 **CRXOVRF**: Clear receive overrun error flag
Writing 1 to this bit clears the RXBOCREF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

Bit 2 **CRXBERF**: Clear receive CRC error flag
Writing 1 to this bit clears the RXBERF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

Bit 1 **CTXBEF**: Clear transmit buffer empty flag
Writing 1 to this bit clears the TXBEF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

Bit 0 **CRXBFF**: Clear receive buffer full flag
Writing 1 to this bit clears the RXBFF flag in the SWPMI_ISR register
Writing 0 to this bit does not have any effect

52.6.5 SWPMI Interrupt Enable register (SMPMI_IER)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RDYIE	Res.	Res.	SRIE	TCIE	TIE	RIE	TXUNREIE	RXOVREIE	RXBEIE	TXBERIE	RXBFIE
				rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RDYIE**: Transceiver ready interrupt enable
0: Interrupt is inhibited
1: A SWPMI interrupt is generated whenever RDYF flag is set in the SWPMI_ISR register

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SRIE**: Slave resume interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever SRF flag is set in the SWPMI_ISR register

Bit 7 **TCIE**: Transmit complete interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever TCF flag is set in the SWPMI_ISR register

Bit 6 **TIE**: Transmit interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever TXE flag is set in the SWPMI_ISR register

Bit 5 **RIE**: Receive interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever RXNE flag is set in the SWPMI_ISR register

Bit 4 **TXUNRIE**: Transmit underrun error interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever TXBUNRF flag is set in the SWPMI_ISR register

Bit 3 **RXOVRIE**: Receive overrun error interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever RXBOVRF flag is set in the SWPMI_ISR register

Bit 2 **RXBERIE**: Receive CRC error interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever RXBERF flag is set in the SWPMI_ISR register

Bit 1 **TXBEIE**: Transmit buffer empty interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever TXBEF flag is set in the SWPMI_ISR register

Bit 0 **RXBFIE**: Receive buffer full interrupt enable
0: Interrupt is inhibited
1: An SWPMI interrupt is generated whenever RXBFF flag is set in the SWPMI_ISR register

52.6.6 SWPMI Receive Frame Length register (SWPMI_RFL)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFL[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **RFL[4:0]**: Receive frame length

RFL[4:0] is the number of data bytes in the payload of the received frame. The two least significant bits RFL[1:0] give the number of relevant bytes for the last SWPMI_RDR register read.

52.6.7 SWPMI Transmit data register (SWPMI_TDR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TD[31:0]**: Transmit data

Contains the data to be transmitted.

Writing to this register triggers the SOF transmission or the next payload data transmission, and clears the TXE flag.

52.6.8 SWPMI Receive data register (SWPMI_RDR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RD[31:0]**: received data
Contains the received data
Reading this register is clearing the RXNE flag.

52.6.9 SWPMI Option register (SWPMI_OR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														SWP_ CLASS	SWP_ TBYP
														rw	rw

Bits 31:2 Reserved, must be kept at reset value

Bit 1 **SWP_CLASS**: SWP class selection
This bit is used to select the SWP class (refer to *Section 52.3.3: SWP initialization and activation*).
0: Class C: SWPMI_IO uses directly VDD voltage to operate in class C.
This configuration must be selected when VDD is in the range [1.62 V to 1.98 V]
1: Class B: SWPMI_IO uses an internal voltage regulator to operate in class B.
This configuration must be selected when VDD is in the range [2.70 V to 3.30 V]

Bit 0 **SWP_TBYP**: SWP transceiver bypass
This bit is used to bypass the internal transceiver (SWPMI_IO), and connect an external transceiver.
0: Internal transceiver is enabled. The external interface for SWPMI is SWPMI_IO (SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals are not available on GPIOs)
1: Internal transceiver is disabled. SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals are available as alternate function on GPIOs. This configuration is selected to connect an external transceiver

52.6.10 SWPMI register map and reset value table

Table 429. SWPMI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SWPMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWPTEN	DEACT	Res.	Res.	Res.	Res.	SWPACT	LPBK	TXMODE	RXMODE	TXDMA	RXDMA
	Reset value																					0	0					0	0	0	0	0	0
0x04	SWPMI_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0x08	RESERVED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0C	SWPMI_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDYF	DEACTF	SUSP	SRF	TCF	TXE	RXNE	TXUNRF	RXOVRF	RXBERF	TXBEF	RXBFF
	Reset value																					0	0	1	0	1	1	0	0	0	0	1	0
0x10	SWPMI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRDYF	Res.	Res.	CSRF	CTCF	Res.	Res.	CTXUNRF	CRXOVRF	CRXBERF	CTXBEF	CRXBFF
	Reset value																					0			0	0			0	0	0	0	0
0x14	SWPMI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDYIE	Res.	Res.	SRIE	TCIE	TIE	RIE	TXUNRIE	RXOVRIE	RXBERIE	TXBEIE	RXBFIE
	Reset value																					0			0	0	0	0	0	0	0	0	0
0x18	SWPMI_RFL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFL[4:0]				
	Reset value																												0	0	0	0	0
0x1C	SWPMI_TDR	TD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SWPMI_RDR	RD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	SWPMI_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWP_CLASS	SWP_TBYP
	Reset value																															0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

53 Management data input/output (MDIOS)

53.1 MDIOS introduction

An MDIO bus can be useful in systems where a master chip needs to manage (configure and get status data from) one or multiple slave chips. The bus protocol uses only two signals:

- MDC: the Management Data Clock
- MDIO: the data line carrying the opcode (write or read), the slave (port) address, the MDIOS register address, and the data

In each transaction, the master either reads the contents of an MDIOS register in one of its slaves, or it writes data to an MDIOS register in one of its slaves.

The MDIOS peripheral serves as a slave interface to an MDIO bus. An MDIO master can use the MDC/MDIO lines to write and read 32 16-bit MDIOS registers which are held in the MDIOS. These MDIOS registers are managed by the firmware, thus allowing the MDIO master to configure the application running on the STM32 and get status information from it.

The MDIOS can operate in Stop mode, optionally waking up the STM32 if the MDIO master performs a read or a write to one of its MDIOS registers.

53.2 MDIOS main features

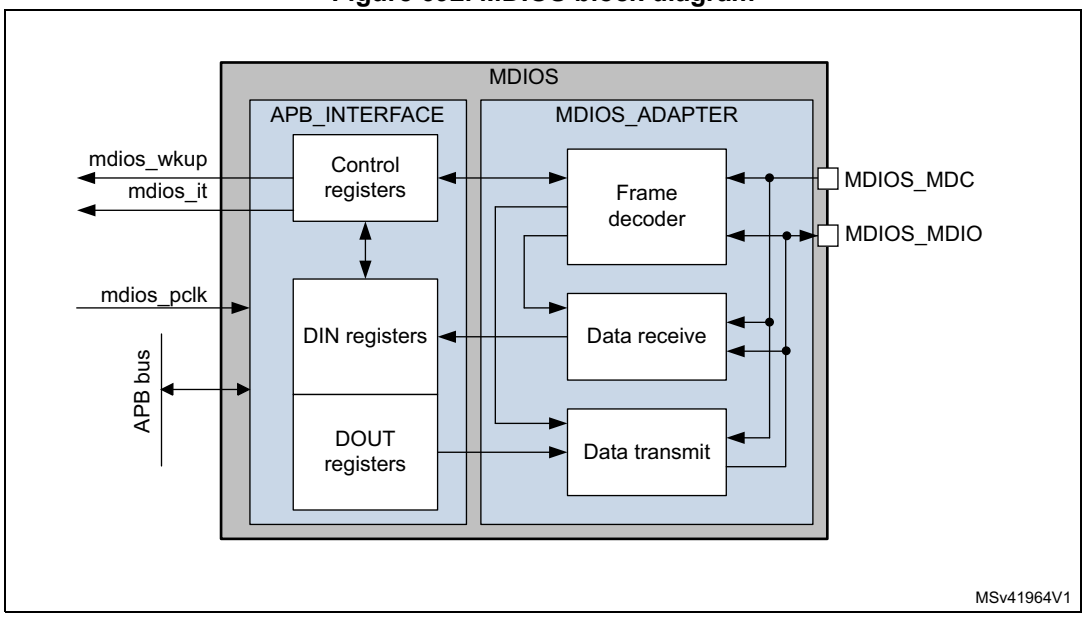
The MDIOS includes the following features:

- 32 MDIOS registers addresses, each of which is managed using separate input and output data registers:
  - 32 x 16-bit firmware read/write, MDIOS read-only output data registers
  - 32 x 16-bit firmware read-only, MDIOS write-only input data registers
- Configurable slave (port) address
- Independently maskable interrupts/events:
  - MDIOS register write
  - MDIOS register read
  - MDIOS protocol error
- Able to operate in and wake up from Stop mode

53.3 MDIOS functional description

53.3.1 MDIOS block diagram

Figure 692. MDIOS block diagram



53.3.2 MDIOS pins and internal signals

Table 430 lists the MDIOS inputs and output signals connected to package pins or balls, while *Table 431* shows the internal PWR signals.

Table 430. MDIOS input/output signals connected to package pins or balls

Signal name	Signal type	Description
MDIOS_MDC	Digital input	MDIO master clock
MDIOS_MDIO	Digital input/ output	MDIO signal (opcode, address, input/output data)

Table 431. MDIOS internal input/output signals

Signal name	Signal type	Description
mdios_wkup	Digital output	MDIOS wakeup signal
mdios_it	Digital output	MDIOS interrupt signal
mdios_pclk	Digital input	APB clock

53.3.3 MDIOS protocol

The MDIOS protocol uses two signals:

1. MDIOS_MDC: the clock, always driven by the master
2. MDIOS_MDIO: signal carrying the opcode, address, and bidirectional data

Each transaction is performed using a “frame”. Each frame contains 32 bits: 14 control bits, 2 turn-around bits, and then 16 data bits, each passed serially.

- 14 control bits, driven by the master
  - 2 start bits: always “01”
  - 2 opcode bits: read=”10”, write=”01”
  - 5 port address bits, indicating which slave device is being addressed
  - 5 MDIOS register address bits, up to 32 MDIOS registers can be addressed in each slave
- 2 turn-around state bits
  - On write operations, the master drives “10”
  - On read operations, the first bit is high-impedance, and the second bit is driven by the slave to ‘0’
- 16 data bits
  - On write operations, data written to slave’s MDIOS register is driven by the master
  - On read operations, data read from slave’s MDIOS register is driven by the slave

Each frame is usually preceded by a preamble, where the MDIOS stays at ‘1’ for 32 MDC clocks. The master can continue to keep MDIO at ‘1’, indicating the “idle” condition, when it has no frame to send.

When MDIO signal is driven by the master, MDIOS samples it using the rising edge of MDC. When MDIOS drives MDIO, the output changes on the rising edge of MDC.

Figure 693. MDIO protocol write frame waveform

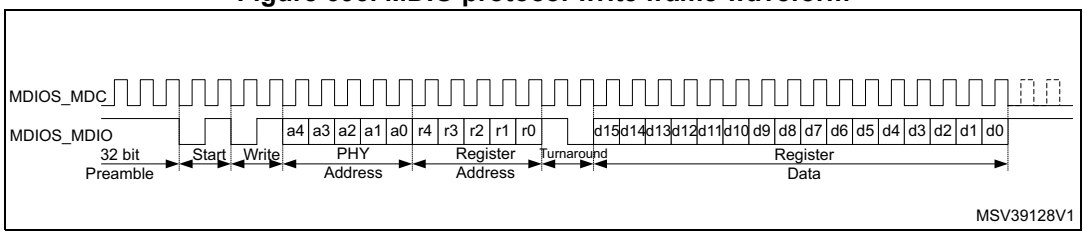
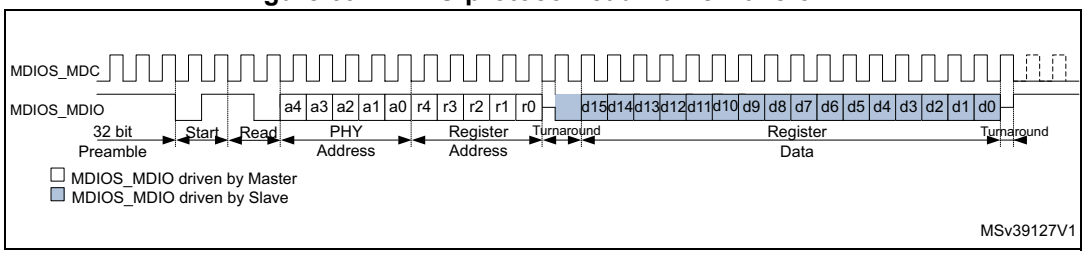


Figure 694. MDIO protocol read frame waveform



53.3.4 MDIOS enabling and disabling

The MDIOS is enabled by setting the EN bit in the MDIOS_CR register. When EN=1, the MDIOS monitors the MDIO bus and service frames addressed to one of its MDIOS registers.

When the MDIOS is enabled (setting EN to ‘1’), the same write operation to the MDIOS_CR register must properly set the PORT_ADDRESS[4:0] field to indicate the slave port address. A frame is ignored by the MDIOS if its port address is not the same as PORT_ADDRESS[4:0] (presumably intended for another slave).

When EN=0, the MDIOS ignores the frames being transmitted on the MDC/MDIO lines, and the IP is in a reduced consumption mode. Clearing EN also clears all of the DIN registers. If EN is cleared while the MDIOS is driving read data, it immediately releases the bus and does not drive the rest of the data. If EN is cleared while the MDIOS is receiving a frame, the frame is aborted and the data is lost.

When the MDIOS is enabled, then disabled and subsequently re-enabled, the status flags are not cleared. For a correct operation the firmware shall clear the status flag before re-enabling the MDIOS.

53.3.5 MDIOS data

From the point of view of the MDIO master, there are 32 16-bit MDIOS registers in the MDIOS which can be written and read. In reality, for each MDIOS register ‘n’ there are two sets of registers: DINn[15:0] and DOUTn[15:0].

Input data

When the MDIO master transmits a frame which writes to MDIOS register ‘n’ in the MDIOS, it is the DINn[15:0] register which is updated with the incoming data. The DIN registers (DIN0 - DIN31) can be read by the firmware, but they can be written only by the MDIO master via the MDIO bus.

The contents of DINn change immediately after the MDC rising edge when the last data bit is sampled.

If the firmware happens to read the contents of DINn at the moment that it is being updated, there is a possibility that the value read is corrupted (a bit-by-bit cross between the old value and the new value). For this reason, **the frmware should assure that two subsequent reads from the same DINn register give the same value and assure that the data was stable when it was read**. In the very worst case, the firmware would need to read DINn four times: first to get the old value, second to get an incoherent value (when reading at the moment the register changes), third to get the new value, and forth to confirm the new value.

If the firmware uses the WRF interrupt and can guarantee that it reads the DINn register before any new MDIOS write frame completes, the firmware can perform a single read.

If the MDIO master performs a write operation with a register address that is greater than 31, the MDIOS ignores the frame (the data is not saved and no flag is set).

Output data

When the MDIOS receives a frame which requests to read register ‘n’, it returns the value found in the DOUTn[15:0] register. Thus, if the MDIO master expects to read the same value which it previously wrote to MDIOS register ‘n’, the firmware must copy the data from

DINn to DOUTn each time new data is written to DINn. For correct operation, the firmware must copy the data to the DOUTn register within a preamble (if the master sends preambles before each frame) plus 15 cycles time.

When an MDIOS register is read via the MDIO bus, the MDIOS passes the 16-bit value (from the corresponding DOUTn register) to the MDIOS clock domain during the 15th cycle of the read frame. If the firmware attempts to write the DOUTn register while the MDIO Master is currently reading MDIOS register ‘n’, then the firmware write operation will be ignored if it occurs during the 15th cycle of the frame (during a one-MDC-cycle window). Therefore, **after writing a DOUTn register, the firmware should read back the same DOUTn register and confirm that the value was actually written**. If the DOUTn register does not contain the value which was written, then the firmware can simply try writing and re-reading again.

If the MDIOS frequency is very slow compared to the mdios_pclk frequency, then it might be best not to tie up the CPU by continuously writing and re-reading a DOUT register. Please note that the read flag (RDFn) is set as soon as the DOUTn value is passed to the MDIOS clock domain. Thus, when a write to DOUTn is ignored (when the value read back is not the value which was just written), then the firmware can use a read interrupt to know when it is able to write DOUTn.

Here is a procedure which can be used if the MDC clock is very slow:

1. Write DOUTn.
2. Assure that all of the read flags are zero (MDIOS_RDFR = 0x0000). Clear the flags if necessary using MDIOS_CRDFR.
3. Read back the same DOUTn register and compare the value with the value which was written in step 1.
4. If the values are the same, then the procedure is done. Otherwise, continue to step 5.
5. Enable read interrupts by setting the RDIE bit in MDIOS_CR1.
6. In the interrupt routine, assure that RDFn is set. (no other read flags will be set before bit n).
7. There is a 31 cycle + preamble time window (if the master sends a preamble before each frame) to write DOUTn safely without needing to do a read-back and compare. If this maximum delay cannot be guaranteed, go back to step #1.

If the MDIO master performs a read operation with a register address which is greater than 31, the MDIOS returns a data value of all zeros.

53.3.6 MDIOS APB frequency

Whenever the firmware reads from an MDIOS_DINRn register or writes to an MDIOS_DOUTRn register, the frequency of the APB bus must be at least 1.5 times the MDC frequency. For example, if MDC is at 20MHz, the APB must be at 30MHz or faster.

53.3.7 Write/read flags and interrupts

When MDIOS register ‘n’ is written via the MDIO bus, the WRFn bit in the MDIOS_WRFR register is set. WRFn becomes ‘1’ a the moment that DINn is updated, which is when the last data bit is sampled on a write frame. An interrupt is generated if WRIEN=1 (in the MDIOS_CR register). WRFn is cleared by software by writing ‘1’ to CWRFn (in the MDIOS_CWRFR register).

When MDIOS register ‘n’ is read via the MDIO bus, the RDFn bit in the MDIOS_RDFR register is set. RDFn becomes ‘1’ at the moment that DOUTn is copied to the MDC clock domain, which is on the 15th cycle of a read frame. An interrupt is generated if RDIEN=1 (in the MDIOS_CR register). RDFn is cleared by software by writing ‘1’ to CRDFn (in the MDIOS_CRDFR register).

53.3.8 MDIOS error management

There are three types of errors with their corresponding error flags:

- Preamble error: PERF (bit 0 of MDIOS_SR register)
- Start error: SERF (bit 1 of MDIOS_SR register)
- Turnaround error: TERF (bit 2 of MDIOS_SR register)

Each error flag is set by hardware when the corresponding error condition occurs. Each flag can be cleared by writing ‘1’ to the corresponding bit in the clear flag register (MDIOS_CLRFR).

An interrupt occurs if any of the three error flags is set while EIE=1 (MDIOS_CR).

Besides setting an error flag, the MDIOS performs no action for a frame in which an error is detected: the DINn registers are not updated and the MDIO line is not forced during the data phase.

For a given frame, errors do not accumulate. For example, if a preamble error is detected, no check is done for a start error or a turnaround error for the rest of the current frame.

When DPC=0, following an detected error, all new frames and errors will be ignored until a complete full preamble has been detected.

When DPC=1 (Disable Preamble Check, MDIOS_CR[7]), all frames and new errors are ignored as long as one of the error flags is set. As soon as the error bit is cleared, the MDIOS starts looking for a start sequence. Thus, the application must clear the error flag only when it is sure that no frame is currently in progress. Otherwise, the MDIOS will likely misinterpret the bits being sent and become desynchronized with the master.

Preamble errors

A preamble error occurs when a start sequence begins (with MDIO sampled at ‘0’) without being immediately preceded by a preamble (MDIO sampled at ‘1’ for at least 32 consecutive clocks).

Preamble errors are not reported after the MDIOS is first enabled (EN=1 in MDIOS_CR) until after a full preamble is received. This is to avoid an error condition when the peripheral frame detection is enabled while a preamble or frame is already in progress. In this case, the MDIOS ignores the first frame (since it did not first detect a full preamble), but does not set PERF.

If the DPC bit (Disable Preamble Check, MDIOS_CR[7]) is set, then the MDIO Master can send frames without preceding preambles and no preamble error will be signaled. When DPC=1, the application must assure that the master is not in the process of sending a frame at the moment that the MDIOS is enabled (EN is set). Otherwise, the slave might become desynchronized with the master.

Start errors

A start error occurs when an illegal start sequence occurs or if an illegal command is given. The start sequence must always be “01”, and the command must be either “01” (write) or “10” (read).

As with preamble errors, start errors are not reported until after a full preamble is received.

Turnaround errors

A turnaround error occurs when an error is detected in the turnaround bits of write frames. The 15th bit of the write frame must be ‘1’ and the 16th bit must be ‘0’.

Turnaround errors are only reported after a full preamble is received, there is no start error, the port address in the current frame matches and the register address is in the supported range 0 to 31.

53.3.9 MDIOS in Stop mode

The MDIOS can operate in Stop mode, responding to all reads, performing all writes, and causing the STM32 to wakeup from Stop mode on MDIOS interrupts.

53.3.10 MDIOS interrupts

There is a single interrupt vector for the three types of interrupts (write, read, and error). Any of these interrupt sources can wake the STM32 up from Stop mode. All interrupt flags need to be cleared in order to clear the interrupt line.

Table 432. Interrupt control bits

Interrupt event	Event flag	Enable control bit
Write interrupt	WRF[31:0]	WRIE
Read interrupt	RDF[31:0]	RDIE
Error interrupt	PERF (preamble), SERF (start), TERF (turnaround)	EIE

53.4 MDIOS registers

53.4.1 MDIOS configuration register (MDIOS_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PORT_ADDRESS[4:0]					DPC	Res.	Res.	Res.	EIE	RDIE	WRIE	EN
			rw	rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **PORT_ADDRESS[4:0]**: Slave's address.
Can be written only when the peripheral is disabled (EN=0).
If the address given by the MDIO master matches PORT_ADRESS[4:0], then the MDIOS services the frame. Otherwise the frame is ignored.

Bit 7 **DPC**: Disable Preamble Check.
0: MDIO Master must give preamble before each frame.
1: MDIO Master can send each frame without a preceding preamble, and the MDIOS will not signal a preamble error.
When this bit is set, the application must be sure that no frame is currently in progress when the MDIOS is enabled. Otherwise, the MDIOS can become desynchronized with the master.
This bit cannot be changed unless EN=0 (though it can be changed at the same time that EN is being set).

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **EIE**: Error interrupt enable.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if any of the error flags (PERF, SERF, or TERF in the MDIOS_SR register) is set.

Bit 2 **RDIE**: Register Read Interrupt Enable.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if any of the read flags (RDF[31:0] in the MDIOS_RDFR register) is set.

Bit 1 **WRIE**: Register write interrupt enable.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if any of the read flags (WRF[31:0] in the MDIOS_WRFR register) is set.

Bit 0 **EN**: Peripheral enable.
0: MDIOS is disabled
1: MDIOS is enabled and monitoring the MDIO bus (MDC/MDIO)

53.4.2 MDIOS write flag register (MDIOS_WRFR)

Address offset: 0x04

Power-on reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRF[31:0]**: Write flags for MDIOS registers 0 to 31.

Each bit is set by hardware when the MDIO master performs a write to the corresponding MDIOS register. An interrupt is generates if WRIE (in MDIOS_CR) is set.

Each bit is cleared by software by writing ‘1’ to the corresponding CWRF bit in the MDIOS_CWRFR register.

For WRFn:

0: MDIOS register ‘n’ has not been written by the MDIO master

1: MDIOS register ‘n’ has been written by the MDIO master and the data is available in DINn[15:0] in the MDIOS_DINRn register

53.4.3 MDIOS clear write flag register (MDIOS_CWRFR)

Address offset: 0x08

Power-on reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CWRF[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CWRF[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **CWRF[31:0]**: Clear the write flag

Writing ‘1’ to CWRFn clears the WRFn bit in the MDIOS_WRF register.

53.4.4 MDIOS read flag register (MDIOS_RDFR)

Address offset: 0x0C

Power-on reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDF[**31**:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDF[31:0]**: Read flags for MDIOS registers 0 to 31.

Each bit is set by hardware when the MDIO master performs a read from the corresponding MDIOS register. An interrupt is generates if RDIE (in MDIOS_CR) is set.

Each bit is cleared by software by writing ‘1’ to the corresponding CRDF bit in the MDIOS_CRDFR register.

For RDFn:

0: MDIOS register ‘n’ has not been read by the MDIO master

1: MDIOS register ‘n’ has been read by the MDIO master

53.4.5 MDIOS clear read flag register (MDIOS_CRDFR)

Address offset: 0x10

Power-on reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRDF[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRDF[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **CRDF[31:0]**: Clear the read flag

Writing ‘1’ to CRDFn clears the RDFn bit in the MDIOS_RDF register.

53.4.6 MDIOS status register (MDIOS_SR)

Address offset: 0x14

Power-on reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERF	SERF	PERF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 TERF: Turnaround error flag
0: No turnaround error has occurred
1: A turnaround error has occurred
Writing ‘1’ to CTERF (MDIOS_CLRFR) clears this bits.

Bit 1 SERF: Start error flag
0: No start error has occurred
1: A start error has occurred
Writing ‘1’ to CSERF (MDIOS_CLRFR) clears this bits.

Bit 0 PERF: Preamble error flag
0: No preamble error has occurred
1: A preamble error has occurred
Writing ‘1’ to CPERF (MDIOS_CLRFR) clears this bits.
This bit will not get set if DPC (Disable Preamble Check, MDIOS_CR[7]) is set.

Note: Writes to MDIOS_SR have no effect.

53.4.7 MDIOS clear flag register (MDIOS_CLRFR)

Address offset: 0x18

Power-on reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTERF	CSERF	CPERF
													rc_w1	rc_w1	rc_w1

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CTERF**: Clear the turnaround error flag
Writing ‘1’ to this bit clears the TERF flag (MDIOS_SR).
When DPC=’1’ (MDIOS_CR[7]), the TERF flag must be cleared only when there is not a frame already in progress.

Bit 1 **CSERF**: Clear the start error flag
Writing ‘1’ to this bit clears the SERF flag (MDIOS_SR).
When DPC=’1’ (MDIOS_CR[7]), the SERF flag must be cleared only when there is not a frame already in progress.

Bit 0 **CPERF**: Clear the preamble error flag
Writing ‘1’ to this bit clears the PERF flag (MDIOS_SR).

Note: Reading MDIOS_CLRFR returns all zeros.

53.4.8 MDIOS input data register x (MDIOS_DINRx)

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINn[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DINn[15:0]**: Input data received from MDIO Master during write frames
This field written by hardware with the 16-bit data received in a write frame which is addressed to MDIOS register ‘n’.

53.4.9 MDIOS output data register x (MDIOS_DOUTRx)

Address offset: 0x180 + 0x04 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUTn[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DOUTn[15:0]**: Output data sent to MDIO Master during read frames
This field is written by SW. These 16 bits are serially output on the MDIO bus during read frames which address the MDIOS register ‘n’.

53.4.10 MDIOS register map

Table 433. MDIOS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**MDIOS_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_ADDRESS[4:0]					Res.	Res.	Res.	Res.	EEI	RDIE	WRIE	EN
	Reset value																				0	0	0	0	0					0	0	0	0
0x04	**MDIOS_WRFR**	WRF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**MDIOS_CWRFR**	CWRF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**MDIOS_RDFR**	RDF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**MDIOS_CRDFR**	CRDF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**MDIOS_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERF	SERF	PERF
	Reset value																														0	0	0
0x18	**MDIOS_CLRFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTERF	CSERF	CPERF
	Reset value																														0	0	0
0x1C - 0xFC	**Reserved**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	**MDIOS_DINR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIN0[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**MDIOS_DINR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIN1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x17C	**MDIOS_DINR31**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIN31[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x180	**MDIOS_DOUTR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOUT0[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	**MDIOS_DOUTR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOUT1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	

Table 433. MDIOS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1FC	**MDIOS_DOUTR31**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOUT31[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

54 Secure digital input/output MultiMediaCard interface (SDMMC)

54.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.mmca.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 4.51*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit.(HS200 SDMMC_CK speed limited to maximum allowed I/O speed)
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 4.1.* (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.* Card support for two different databus modes: 1-bit (default) and 4-bit. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

54.2 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14.
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 695. SDMMC “no response” and “no data” operations

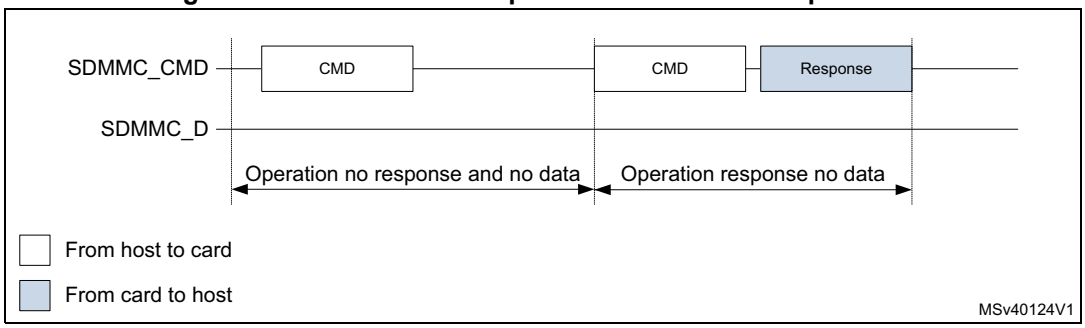
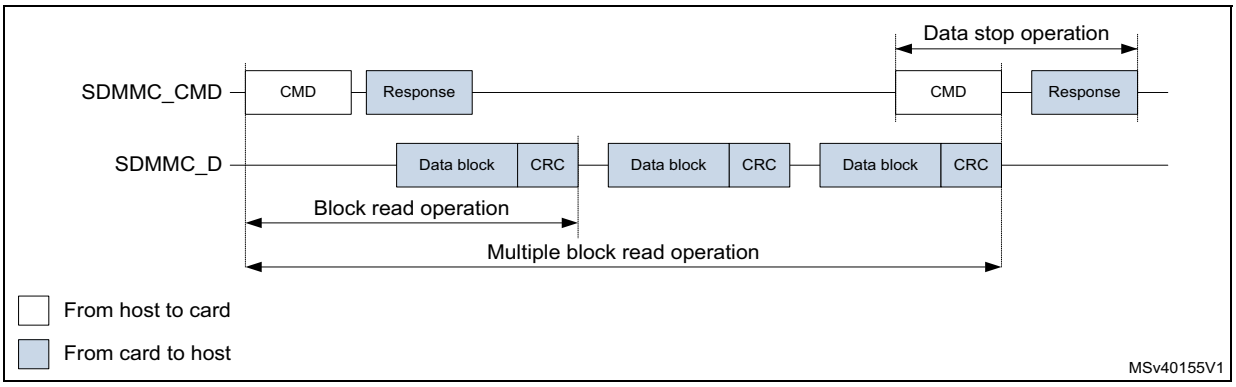
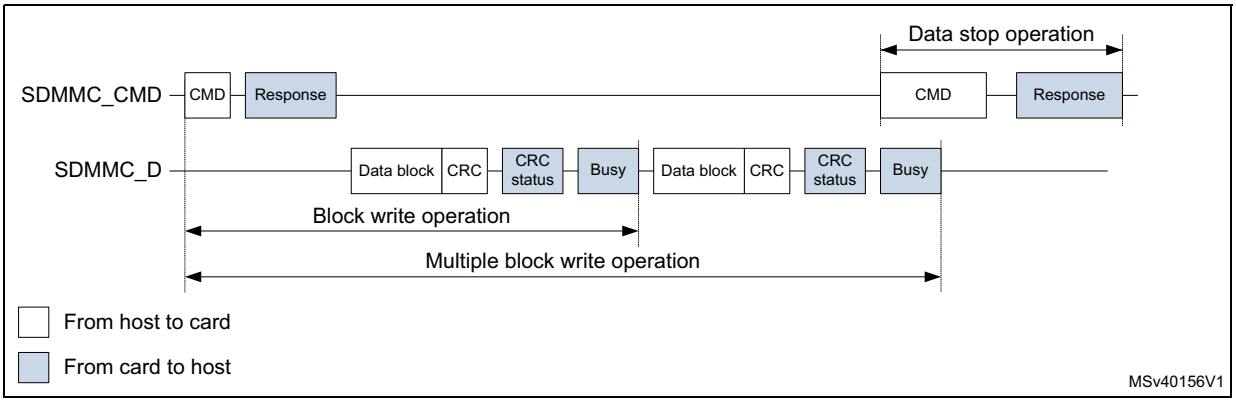


Figure 696. SDMMC (multiple) block read operation



*Note: The Stop Transmission command is not required at the end of a e•*MMC *multiple block read with predefined block count.*

Figure 697. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

Note: The SDMMC will not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 698. SDMMC (sequential) stream read operation

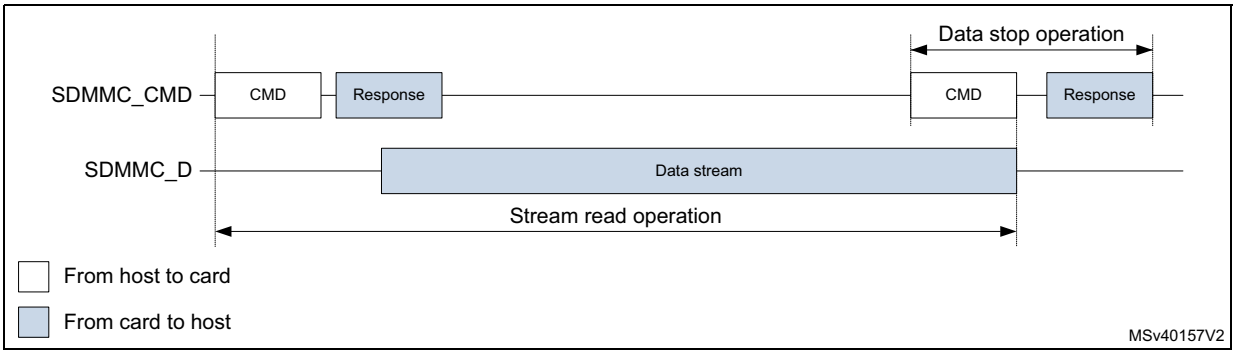
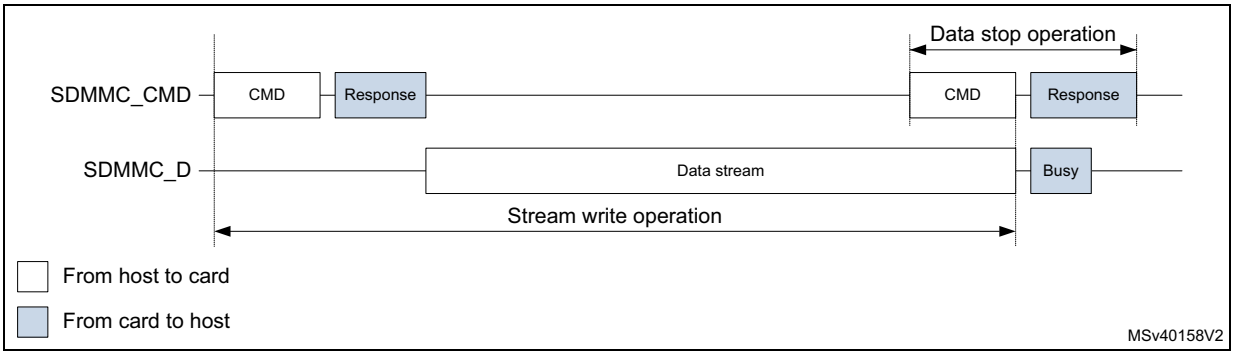


Figure 699. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

54.3 SDMMC operation modes

Table 434. SDMMC operation modes SD & SDIO

SDIO Bus Speed modes(1)(2)	Max Bus Speed (3) [MByte/s]	Max Clock frequency [MHz](4)	Signal Voltage [V]
DS (Default Speed)	12.5	25	3.3
HS (High Speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8
SDR104	104	208	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 435. SDMMC operation modes *e*•MMC

***e*•MMC bus speed modes** (1)(2)	**Max bus speed** (3) **[MByte/s]**	**Max clock frequency [MHz]**(4)	**Signal voltage [V]**
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V
High speed HS200	200	200	1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8bit bus width.
4. Maximum frequency depending on maximum allowed IO speed.

HS200 mode requires variable delay support using sampling point tuning.

54.4 SDMMC functional description

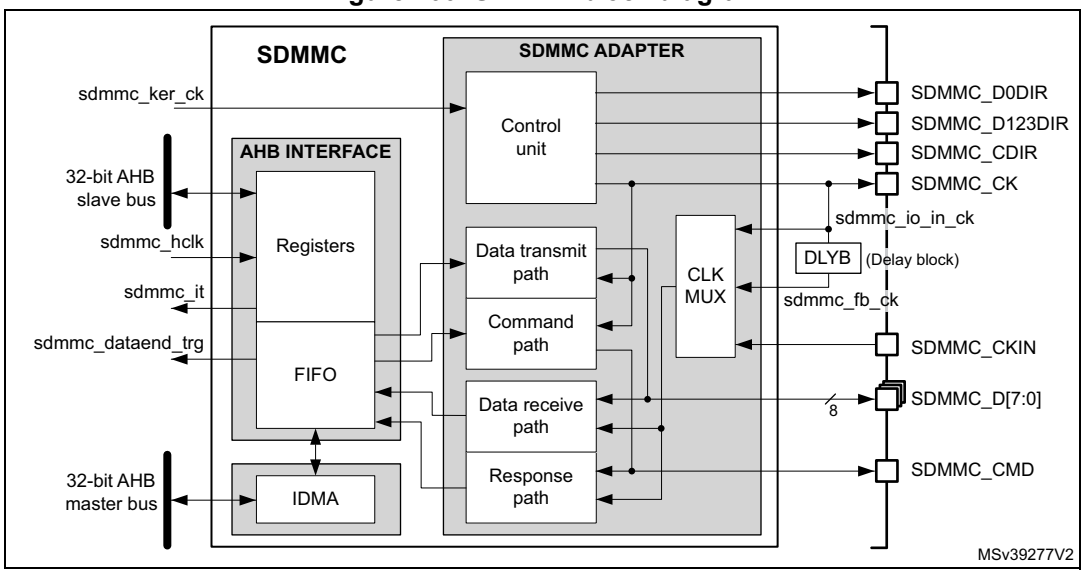
The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the *e*•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

54.4.1 SDMMC diagram

Figure 700 shows the SDMMC block diagram.

Figure 700. SDMMC block diagram



54.4.2 SDMMC pins and internal signals

Table 436 lists the SDMMC internal input/output signals, *Table 437* the SDMMC pins (alternate functions).

Table 436. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt
sdmmc_dataend_trg	Digital output	SDMMC data end trigger for MDMA
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 437. SDMMC pins

Signal name	Signal type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)

Table 437. SDMMC pins

Signal name	Signal type	Description
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

54.4.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 701*.

Figure 701. SDMMC Command and data phase relation

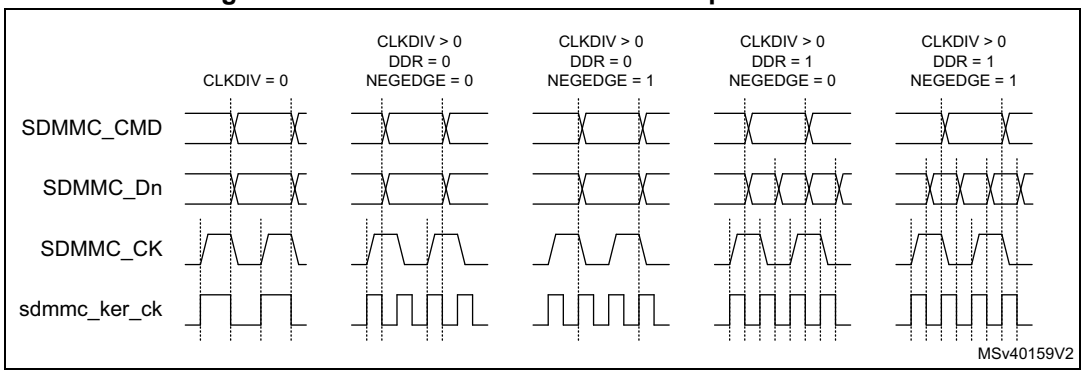


Table 438. SDMMC Command and data phase selection

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	generated on sdmmc_ker_ck falling edge	
>0	0	0	generated on sdmmc_ker_ck rising edge	generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then allows using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

54.4.4 SDMMC adapter

The SDMMC adapter (see *Figure 700: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

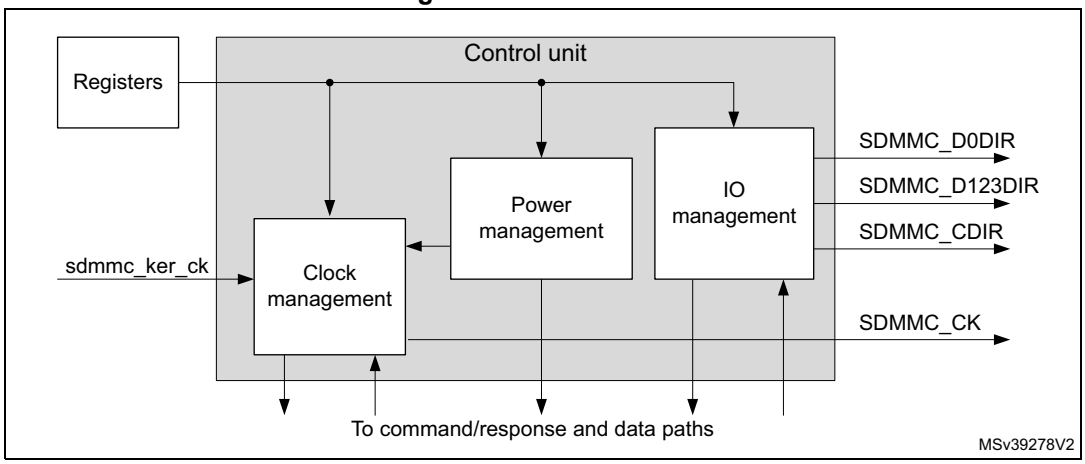
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 702*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 702. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for i.e. flow control.

The clock outputs are inactive:

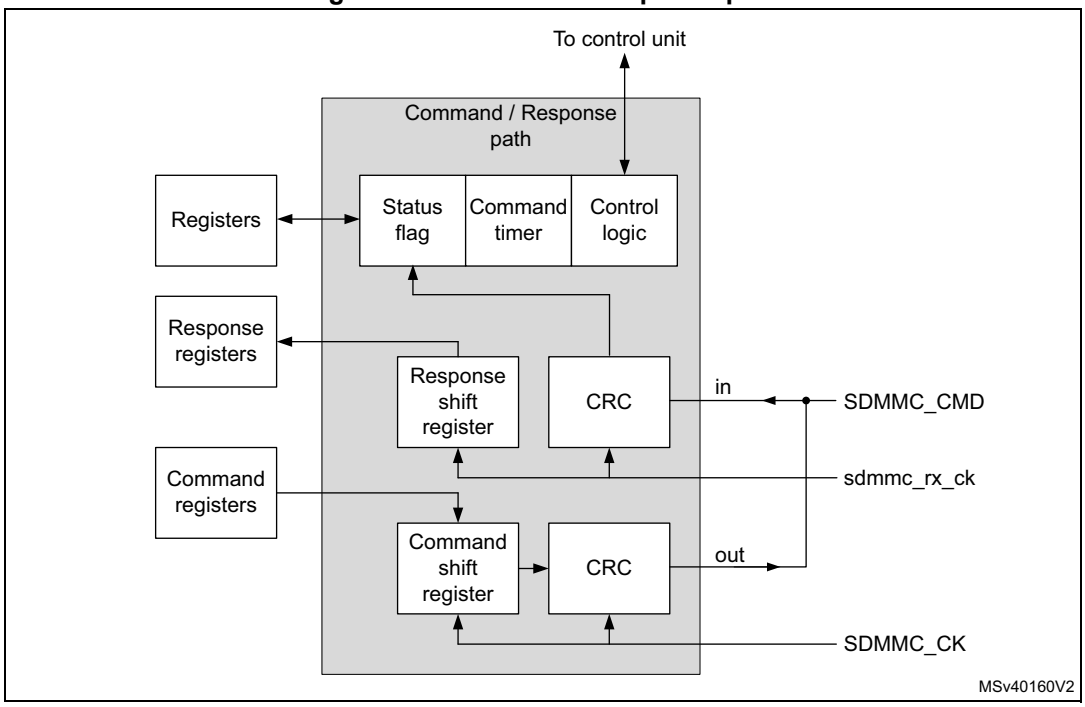
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock will be stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock will be restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/Response path

The Command/Response path subunit transfers commands and responses on the SDMMC_CMD line. The Command path is clocked on the SDMMC_CK and sends commands to the card,.The Response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 703. Command/Response path

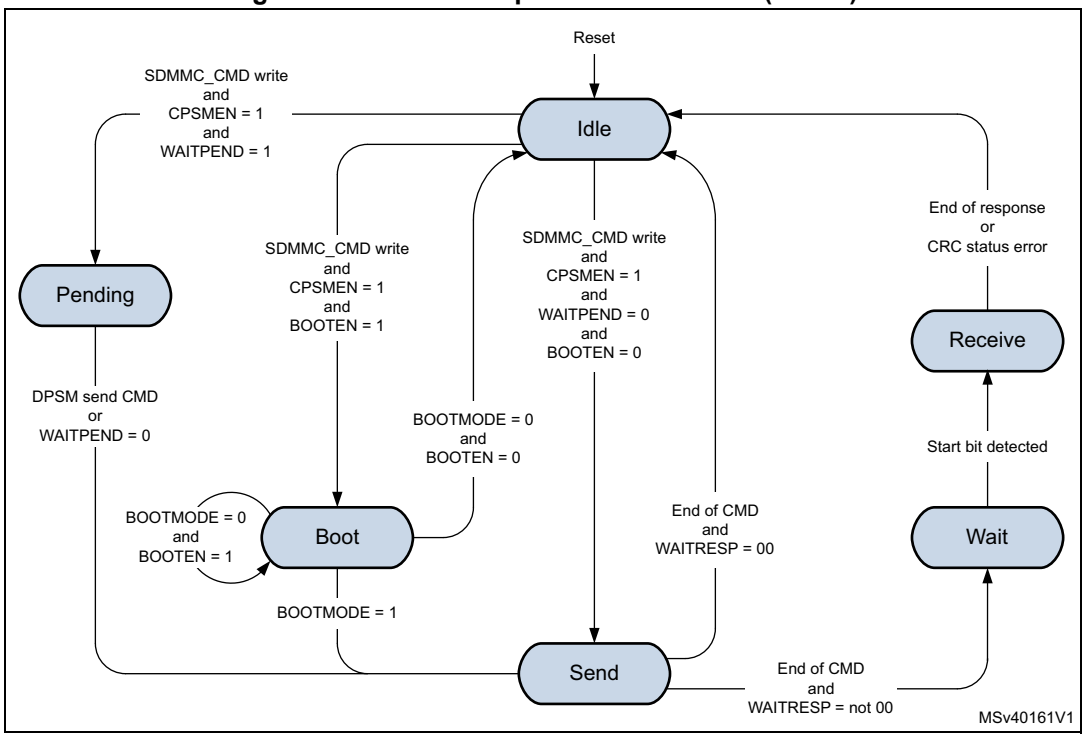


Command/Response path state machine (CPSM)

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active, i.e. not in the Idle state, the CPSMACT bit is set.

Figure 704. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM will activate the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 & BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal will be issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period will be terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal will be issue to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM will move to the Idle state and generate the CMDSENT flag. When BOOTMODE = 1 & BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command End bit, otherwise the

CMDSENT flag is generated immediately after the command End bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM will move to the Wait state and start the response timeout.

- **Wait**: The Command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a Start bit.
  a) If a Start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (Response Start bit) from one of the cards.
  a) When a Start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host will send a response by its self and on detecting the Start bit the CPSM move to the Receive state.
- **Receive**: The command response will be received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present will be verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period will be started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there will be no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period will be terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 & BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response End bit, otherwise the CMDREND flag is generated immediately after the response End bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal will be issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH =< 5 bytes the CPSM moves to the Sent state en generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal will be issued to the DPSM to start the data transfer. The CPSM waits for a sendCMD signal from the

DPSM before moving to the Sent state. This enables i.e. the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM will move to the Sent state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal will be issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal will be issued to the DPSM, before moving to the Idle state.The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

Note: The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A Command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 439.*.

The Command token data is taken from 2 registers, one containing a 32-bits argument and the other containing the 6-bits command index (six bits sent to a card).

Table 439. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the Command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A Response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received Command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 440*, *Table 441* and *Table 442*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the End bit (always 1).

The Response token data is stored in 5 registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 440. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 441. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 442. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The Command/Response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 701*.

The Command and Short Response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status).

For the Long Response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

CRC[6:0] = Remainder [(M(x) * x^7) / G(x)]

G(x) = $x^7 + x^3 + 1$

M(x) = (first bit) * x^n + (second bit) * x^{n-1}+... + (last bit before CRC) * x^0

Where n = 39 or 119.

The CPSM allows to send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 443*.

Table 443. Specific Commands overview

VSWITCH	BOOTEN	BOOTMODE	CMDTRANS	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start Voltage Switch Sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The Command/Response path implements the status flags and associated clear bits shown in *Table 444*:

Table 444. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response. (CPSM moves from SEND to IDLE)
CMDREND	Set at the end of the command response when the CRC is OK. (CPSM moves from RECEIVE to IDLE)
CCRCFAIL	Set at the end of the command response when the CRC is FAIL. (CPSM moves from RECEIVE to IDLE)
CTIMEOUT	Set after the command when no response start bit received before the timeout. (CPSM moves from WAIT to IDLE)
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped. (no impact on CPSM)
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5ms + 1ms. (no impact on CPSM)
CPSMACT	Command transfer in progress. (CPSM not in Idle state)

The Command path error handling is shown in *Table 448*:

Table 445. Command path error handling

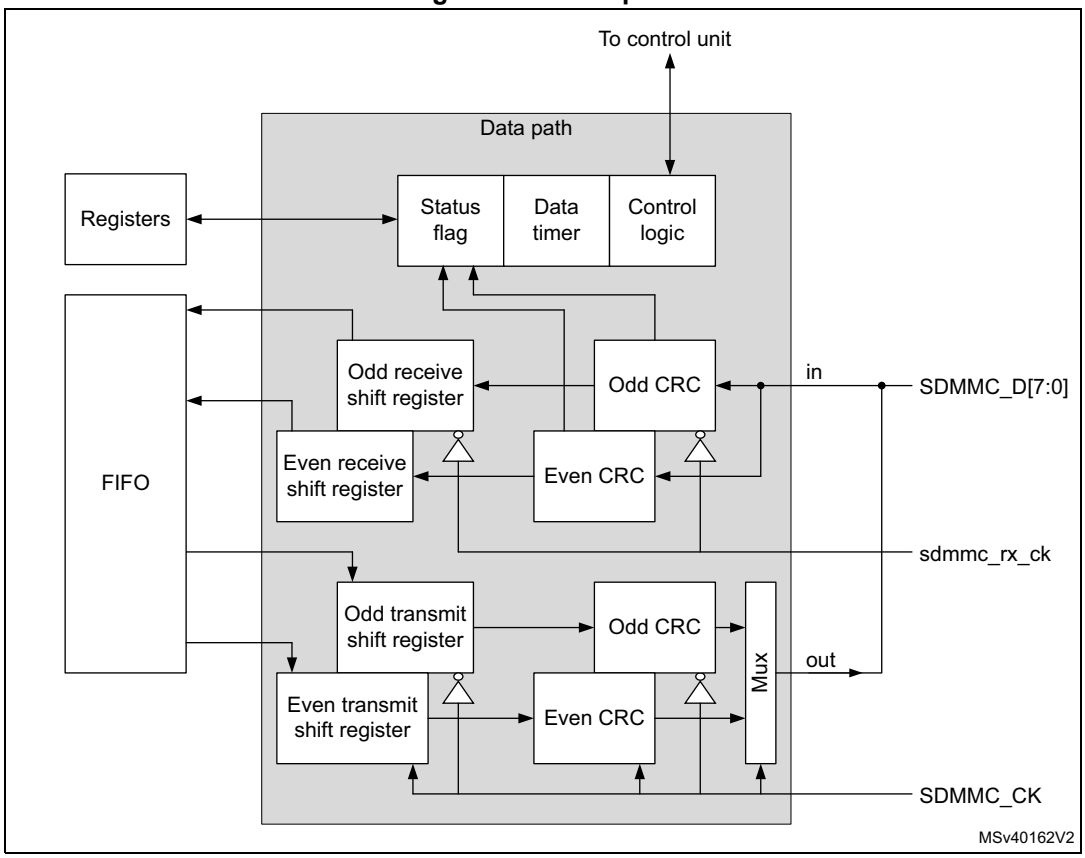
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command shall be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card.The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 705* shows the data path block diagram.

Figure 705. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 706* and *Figure 707*:

- On the rising edge of the clock Odd bytes are sampled.
- On the falling edge of the clock Even bytes are sampled.
- Data payload size is always a multiple of 2 Bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, End bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and Busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode the SDMMC_CK clock division shall be >= 2.

Figure 706. DDR mode data packet clocking

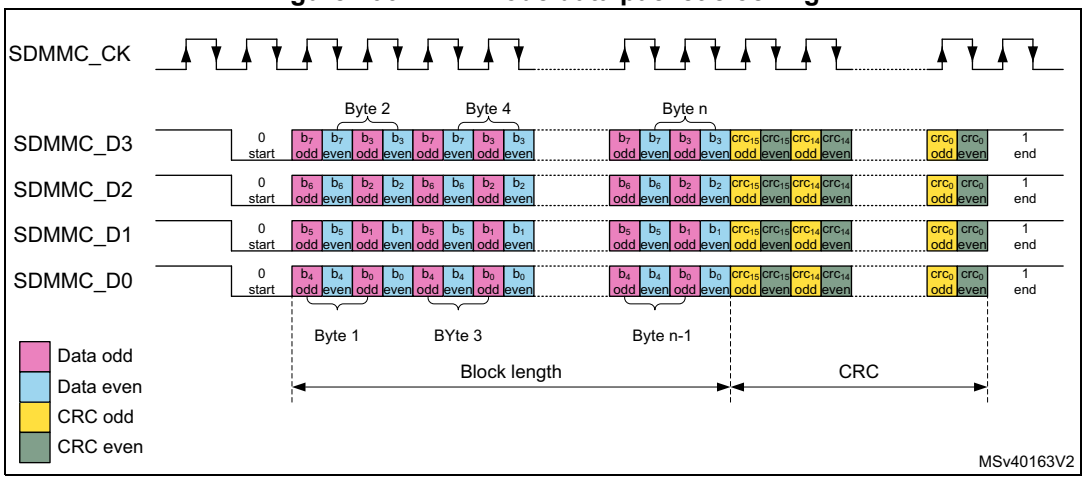
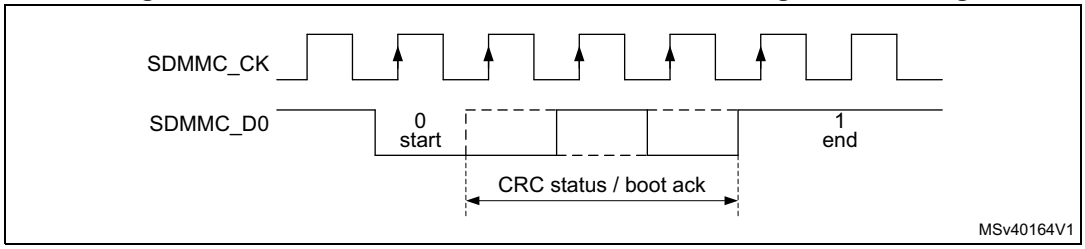


Figure 707. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

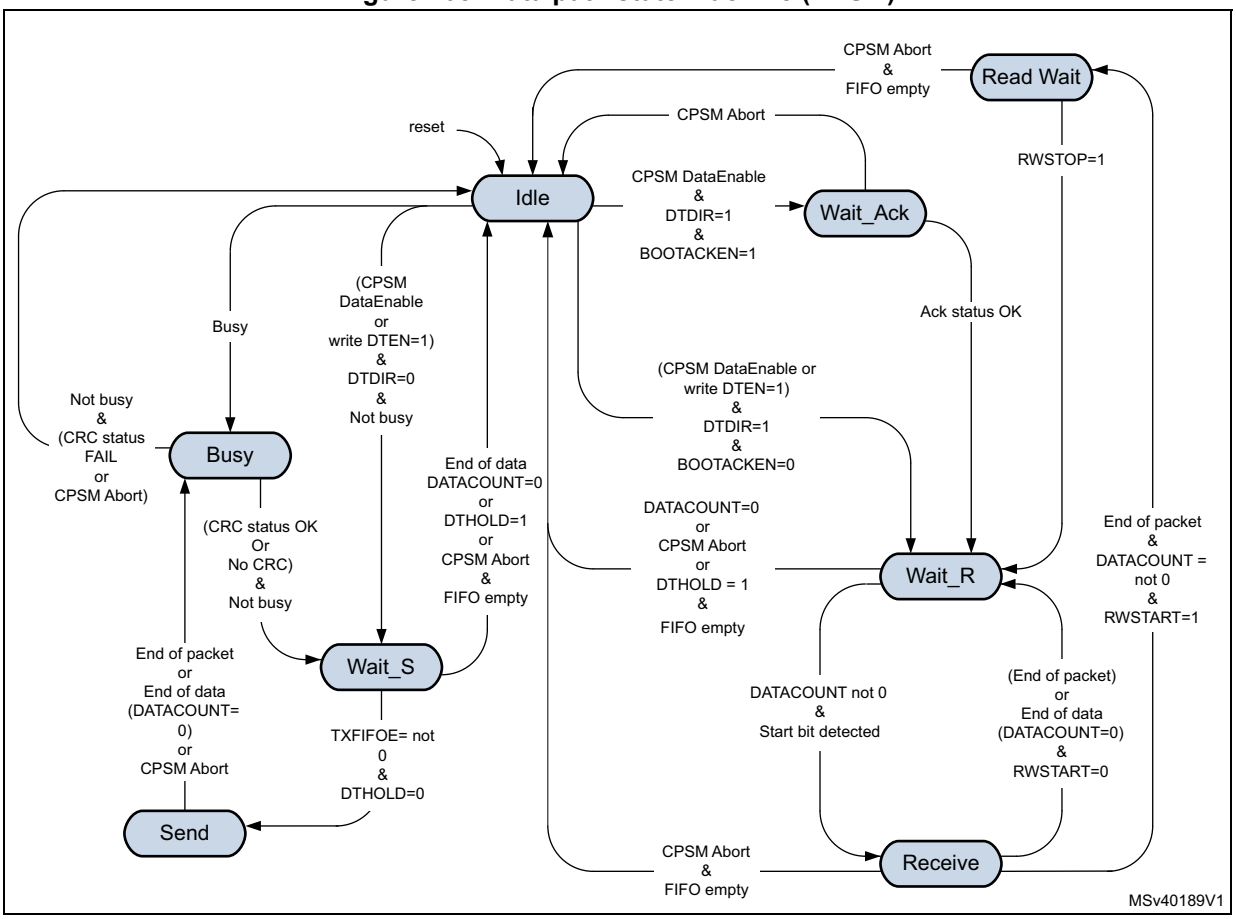
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 708*. When ever the DPSM is active, i.e. not in the Idle state, the DPSMACT bit is set.

Figure 708. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM will be activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM will activate the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

  – When the data direction bit (DTDIR) indicates send, moves to the Wait_S.

  – When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM will keep the SDMMC_CK clock active and move to the Busy state.

Note: DTEN shall not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the Ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the WAIT_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there will be no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When Holding the timeout is disabled. When an CPSM Abort signal is received during Holding, the transfer is Aborted.

- When DATACOUNT = 0, the transfer is completed normally and there will be no DHOLD flag.

– When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **ReadWait** state: the data path ReadWait the bus.
  – The DPSM moves to the Wait_R state when the ReadWait stop bit (RWSTOP) is set, and start the receive timeout.
  – If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  – In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  – In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.

a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
- Wait_R state when RWSTART= 0 and start the receive timeout.
- ReadWait state when RWSTART = 1 and DATACOUNT > zero, and generate the DBCKEND flag.

b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
- When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
- When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.

  – In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  – When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  – When an CPSM_Abort signal is received:.
  - If the CPSM_Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM_Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  – When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  - if the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state and generate the DATAEND flag.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the FIFO will be flushed, then the DPSM will move to the Idle state and issues the DHOLD flag. (The DBCKEND flag will also be set, and can be cleared at the same time as the DHOLD flag.)
    - When IDMA is disabled generate the DBCKEND flag. wait for the FIFO to be reset with FIFORST, then DPSM will move to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM_Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM_Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, normal operation is continued, there will be no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and End bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and End bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream Byte according to DATACOUNT, the DPSM issues a trigger on the sendCMD signal. This signal is used by the CPSM to sent any pending command. (i.e. CMD12 Stop Transmission command)
  - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  - When receiving the CPSM Abort signal
    - If the CPSM_Abort signal is received before the 2 last bits of the transfer with DATACOUNT=0, the transfer is aborted. The DPSM will sent a last data bit followed by an End bit. The FIFO will be disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
    - If the CPSM_Abort signal is received during or after the 2 last bits of the transfer

with DATACOUNT=0, the transfer is completed normally, there will be no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it will remain in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO will be disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM_Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM_Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT=0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM_Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an Abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0,

- In receive the start bit shall be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 446. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	yes	1	00
SDIO multibyte	0	(DATALENGTH)	yes	1	01
e•MMC stream	0	(DATALENGTH)	no	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 447*:

Table 447. Data path status flags and clear bits

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished. (DATACOUNT = 0). (DPSM moves from WAIT_S to IDLE)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from WAIT_R to IDLE)
	BOOT	

Table 447. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in BUSY when there is still data to send and wait for Abort) (DPSM moves from BUSY to IDLE when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in RECEIVE when there is still data to be received and wait for Abort) (DPSM moves from RECEIVE to IDLE when all data has been received or DPSM has been started with DTEN)
	BOOT	
ACKFAIL	BOOT	Set at the end of the BOOT ACK when FAIL. (DPSM stays in Wait_Ack and wait for Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in BUSY and wait for Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in BUSY and wait for Abort) (When DPSM has been started with DTEN move to IDLE) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in WAIT_R and wait for Abort) (When DPSM has been started with DTEN move to IDLE)
	BOOT	
ACKTIMEOUT	BOOT	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for Abort)
DBCKEND	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from WAIT_S to IDLE)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from RECEIVE to READWAIT)
	BOOT	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from WAIT_S to IDLE)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from WAIT_R to IDLE)
DABORT	CMD R1b	When Abort event has been sent by the CPSM and busy has finished. (DPSM moves from BUSY to IDLE)
	TX	When Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from Any state to IDLE)
	RX	
	BOOT	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from BUSY to IDLE)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 448*:

Table 448. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No Start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM_Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM_Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM_Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM_Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM_Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM_Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 449*.

Table 449. Data FIFO access

Data FIFO access	IDMAEN
From FW via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by FW via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH shall be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   – Configures the FIFO in transmit mode.
3. Enabled the data transfer
   – either by sending a Command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   – The DPSM will stay in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.
   – The SDMMC start sending data as long as FIFO is not empty.
5. When the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
6. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, FW shall stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 450*.

Table 450. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by FW via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses. The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH shall be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM will write the received data to the FIFO.
5. When the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
6. When last data has been received end of data (DATAEND flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   – SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, FW shall read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This will cause the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, FW shall stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This will cause the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 451*.

Table 451. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

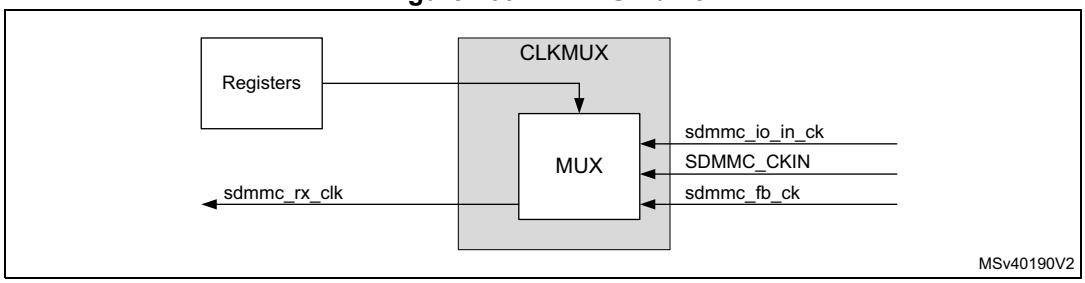
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 709. CLKMUX unit



The sdmmc_rx_ck source shall be changed when the CPSM and DPSM are in the Idle state.

54.4.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer

multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.

- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that will generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

54.4.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- double buffered channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE0. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Double buffered channel

In double buffer configuration the data at the memory side is subsequently accessed from 2 buffers, one located from base address IDMABASE0 and a second located from base address IDMABASE1. This allows firmware to process one memory buffer while the IDMA is accessing the other memory buffer. The size of the memory buffers is defined by IDMABSIZE. The buffer size shall be an integer multiple of the burst size. It is possible to update the base address of the buffers on-the-fly when the channel is enabled, the following rule apply:

- When IDMABACT bit is ‘0’ the IDMA hardware uses the IDMABASE0 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE0 data will not be changed. Firmware is allowed to write IDMABASE1.
- When IDMABACT bit is ‘1’ the IDMA hardware uses the IDMABASE1 to access memory. When attempting to write to this register by Firmware the write is discarded, IDMABASE1 data will not be changed. Firmware is allowed to write IDMABASE0.

When the IDMA has finished transferring the data of one buffer the buffer transfer complete flag (IDMABTC) is set and the IDMABACT bit toggles where after the IDMA continues transferring data from the other buffer. When the IDMA has finished transferring all data and the DPSM has completed the transfer the DATAEND flag is set.

The IDMABASEn address shall be word aligned.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space.

On a IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set. Depending when the IDMA transfer error occurs, it will normally cause the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (HOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a HOLD or DATAEND flag is set.

The IDMATE will be generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, HOLD, or DATAEND).

54.4.7 MDMA request generation

The internal trigger line from the SDMMC allows passing direct request to MDMA controller to enable successive transfers from/to different internal RAM addresses without CPU use.

When a data transfer from/to the card completes successfully, the DATAEND flag of the status register is set. The event is signaled to an MDMA request input through the sdmmc_dataend_trg output. It can trigger the clearance of the DATAEND and CMDREND flags and, eventually, a new transfer start, through MDMA direct access to the SDMMC control and configuration registers, thus without CPU intervention.

The action to program in the MDMA according to the SDMMC requests is provided in the following table:

Table 452. SDMMC connections to MDMA

Trigger signal	Event signaled	Event occurrence condition	MDMA transfer configuration	MDMA action
sdmmc_ dataend_trg	End of successful data transfer	DATAEND = 1	single	Set DATAENDC

54.4.8 AHB and SDMMC_CK clock relation

The AHB shall at least have 3x more bandwidth than the SDMMC bus bandwidth i.e. for SDR50 4-bit mode (50Mbyte/s) the minimum sdmmc_hclk frequency is 37.5MHz (150Mbyte/s).

Table 453. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	19.5
e•MMC HS	8	52	39
e•MMC DDR52	8	52	78
e•MMC HS200	8	200	150
SD DS / SDR12	4	25	9.4
SD HS / SDR25	4	50	18.8
SD DDR50	4	50	37.5
SD SDR50	4	100	37.5
SD SDR104	4	208	78

54.5 Card functional description

54.5.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO read wait operation by stopping the clock
- SDIO read wait operation by SDMMC_D2 signaling

Table 454. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
ReadWait SDMMC_CK clock stop (START)	X	1	0	1	1
ReadWait SDMMC_CK clock stop (STOP)	X	1	1	1	1
ReadWait SDMMC_D2 signaling (START)	X	0	0	1	1
ReadWait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

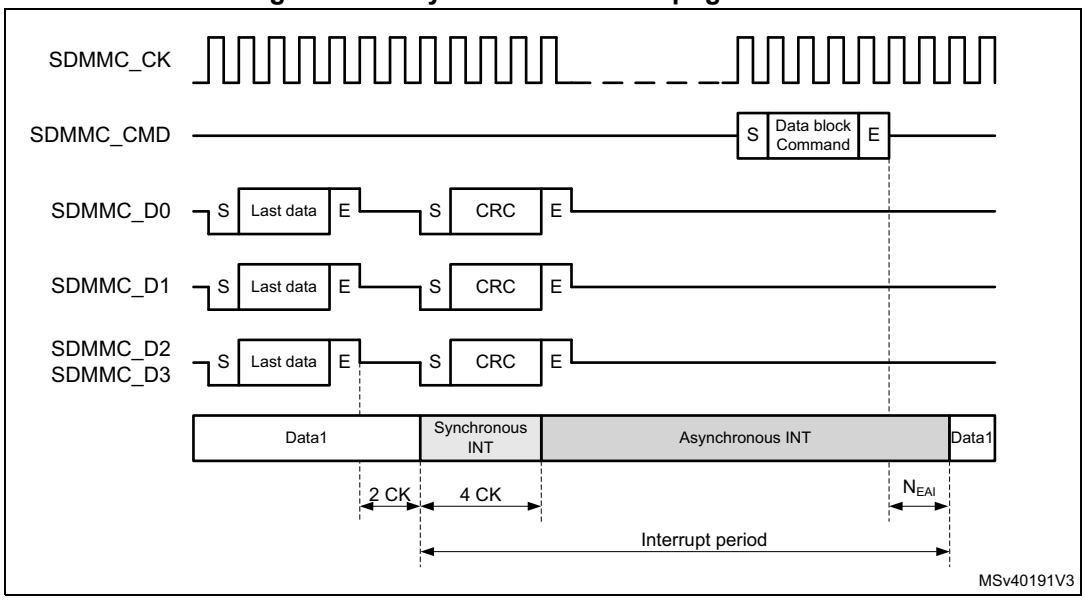
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 454*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 710. Asynchronous interrupt generation



The timing of the interrupt period is depended on the bus speed mode:

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the End bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN shall not be used to start data transfer with SD and e•MMC cards.

Figure 711. Synchronous interrupt period data read

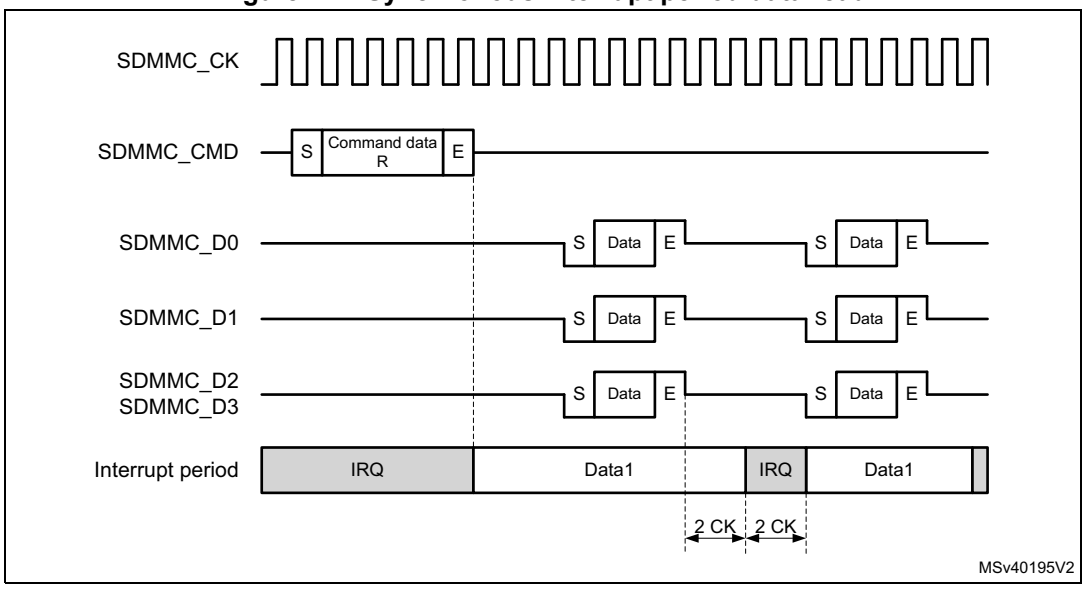
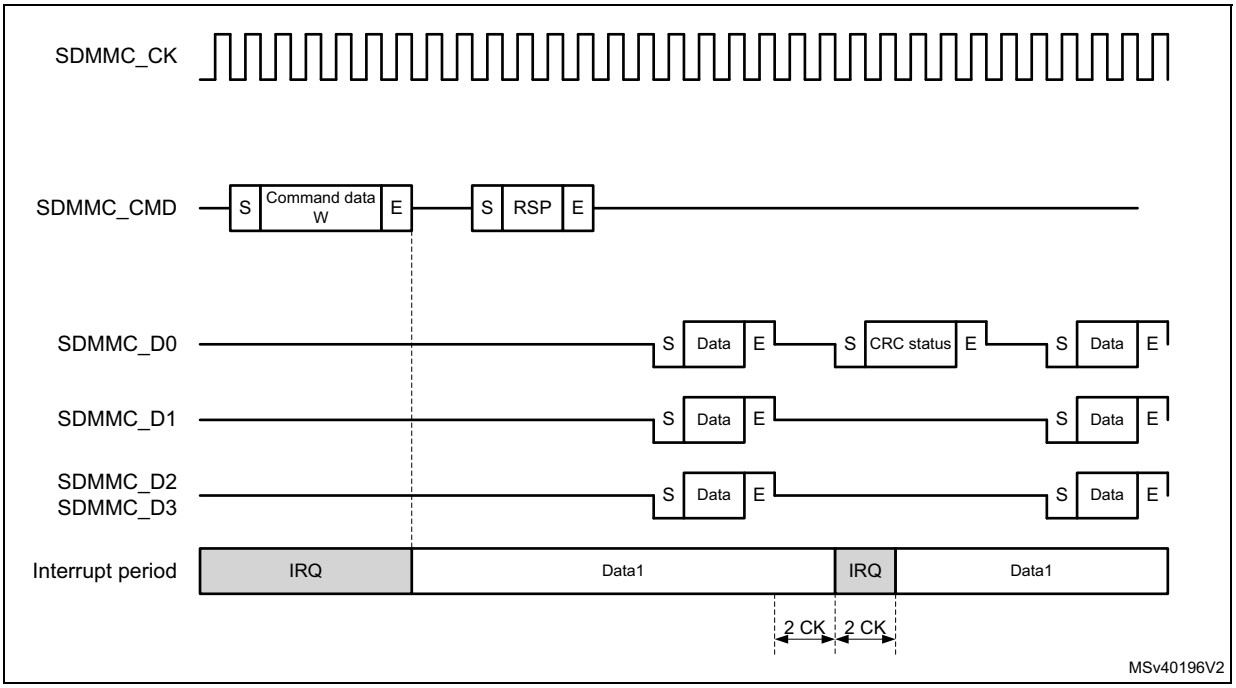


Figure 712. Synchronous interrupt period data write



In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the End bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the End bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command End bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host will resume the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN shall not be used to start data transfer with SD and e•MMC cards.

Figure 713. Asynchronous interrupt period data read

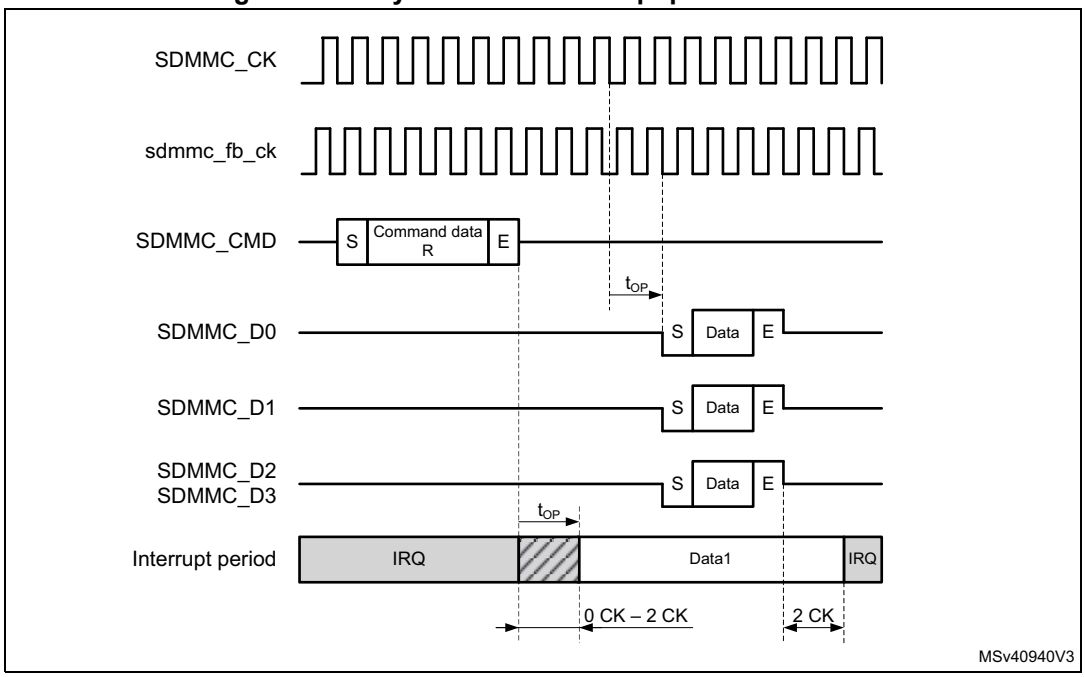
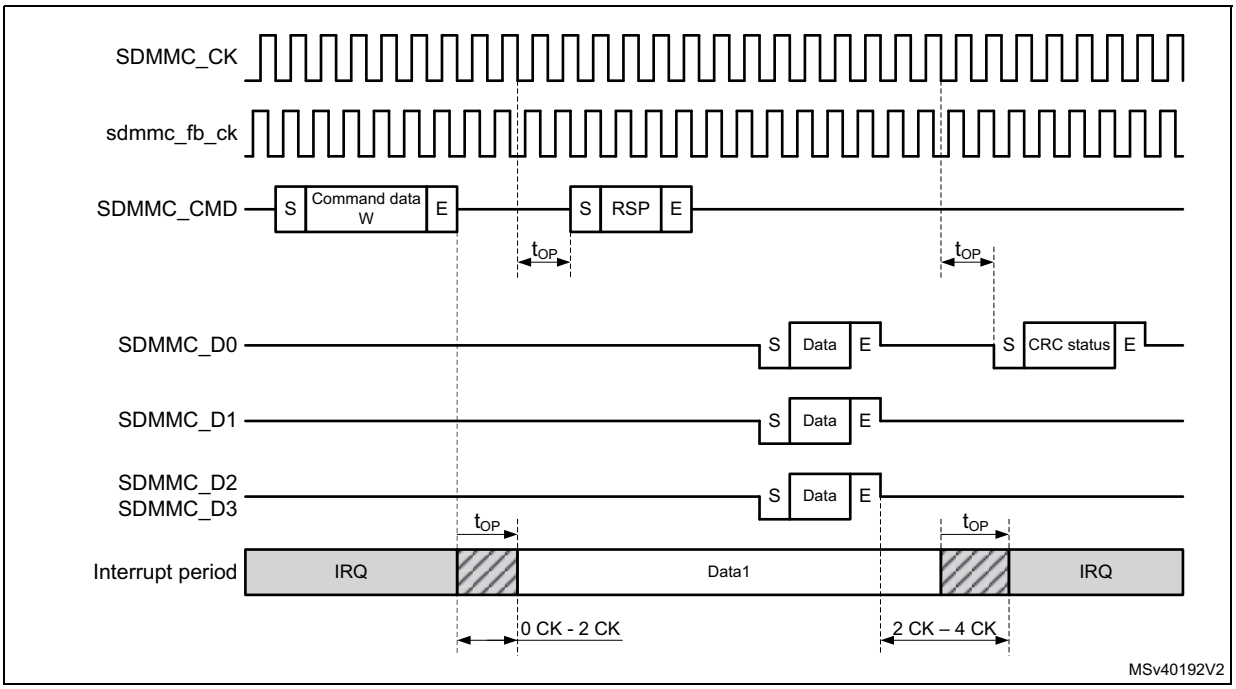


Figure 714. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC will mask the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 455*.

Table 455. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or ReadWait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s)
2. Requests the lower-priority or slower transaction to suspend
3. Waits for the transaction suspension to complete
4. Begins the higher-priority transaction
5. Waits for the completion of the higher priority transaction
6. Restores the suspended transaction

The card receiving a suspend command will respond with its current bus status. Only when the bus has been suspended by the card the bus status will indicated suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted. (data transfer has suspended)
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during ReadWait.

For the host to know if the bus has been released it shall check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation shall be saved where after an other operation can start.

The suspend command shall be sent with the CMDSUSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps shall be taken by firmware:

1. The normal receive process shall be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO shall be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO shall be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation shall be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this shall be ignored.

To resume receiving data from the card, the following steps shall be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) shall be programmed in DATALENGTH.
2. The DPSM shall be configured to receive data in the DTDIR bit.
3. The resume command shall be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which will end the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card will resume sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps shall be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM shall be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO shall be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming shall be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command shall be sent by the CPSM with the CMDSUSPEND bit set. This allows to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps shall be taken by firmware:

1. The remaining number of data bytes shall be programmed in DATALENGTH.
2. The DPSM shall be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This will end the interrupt period and start the data transfer.

The DPSM will either go to the Wait_S state when SDMMC_D0 does not signal busy, or will go to the Busy state when busy is signaled.

3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware shall start filling the FIFO with the remaining data.

SD I/O ReadWait

There are 2 methods to pause the data transfer during the Block gap:

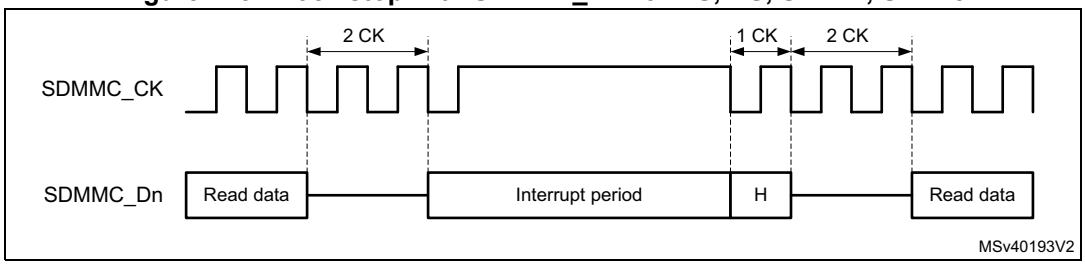
1. Stopping the SDMMC_CK.
2. Using ReadWait signaling on SDMMC_D2.

The SDMMC can perform a ReadWait with register settings according *Table 454*.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

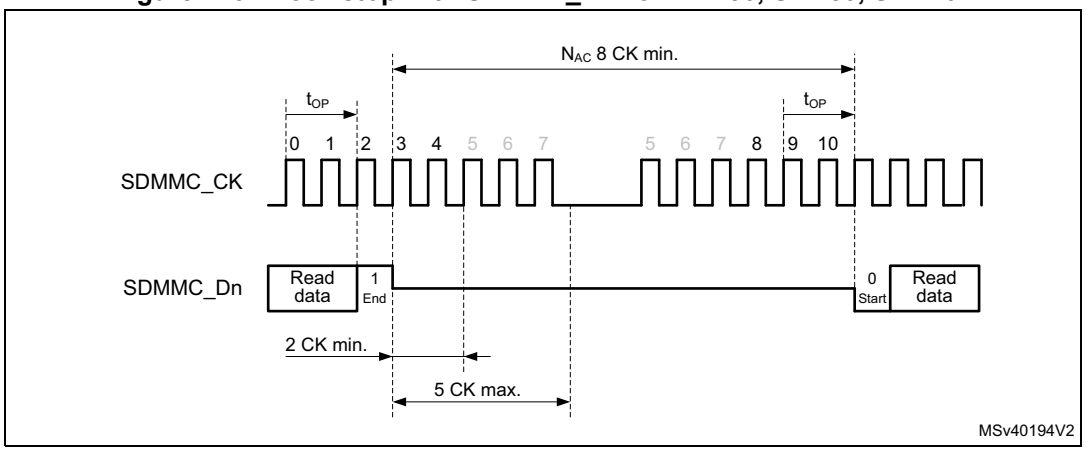
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the End bit. When ready the host resumes by restarting clock, see *Figure 715*.

Figure 715. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the End bit. When ready the host resumes by restarting clock, see *Figure 716*. (In DDR50 mode the SDMMC_CK shall only be stopped after the falling edge, when the clock line is low.)

Figure 716. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



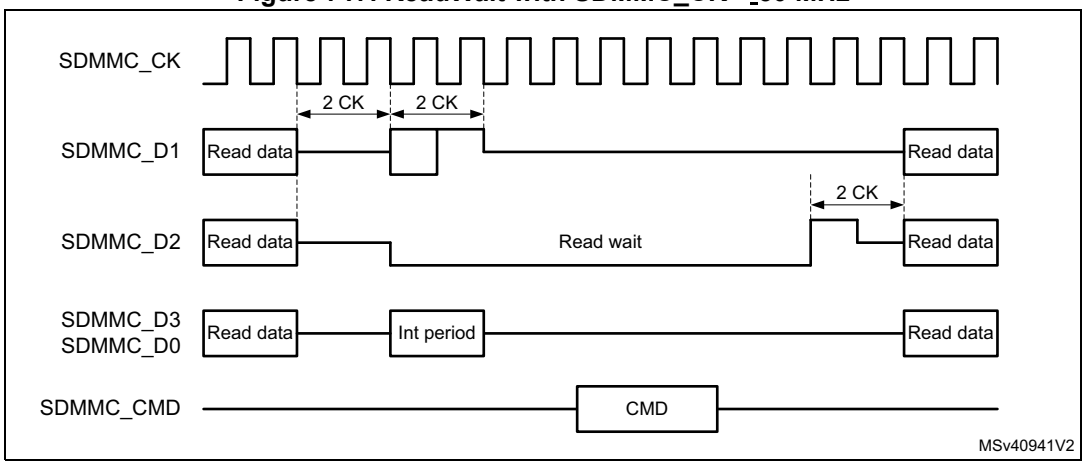
In ReadWait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the End bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a Start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a ReadWait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional ReadWait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The ReadWait operation allows the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the ReadWait protocol, the host must test capability bits in the internal card registers.

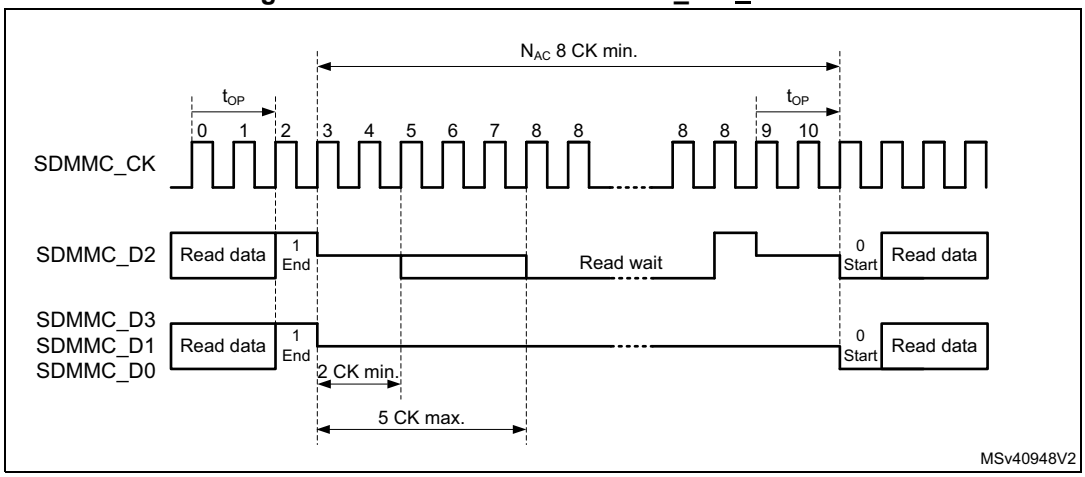
The timing for ReadWait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter ReadWait. To exit ReadWait the host shall raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 717*.

Figure 717. ReadWait with SDMMC_CK <_50 MHz



For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card will treat the ReadWait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter ReadWait. To exit ReadWait the host shall raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host shall raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 718*).

Figure 718. ReadWait with SDMMC_CK ≥ 50 MHz



In ReadWait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the End bit of the current received data block CRC. The ReadWait signaling on SDMMC_D2 will be removed when writing 1 to the RWSTOP bit. The DPSM remains in ReadWait state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a Start bit from the card.

During the ReadWait signaling on SDMMC_D2 commands can be issued to the card. During the ReadWait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

54.5.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the End bit of the Stop Transmission command.

Table 456. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.

Table 456. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

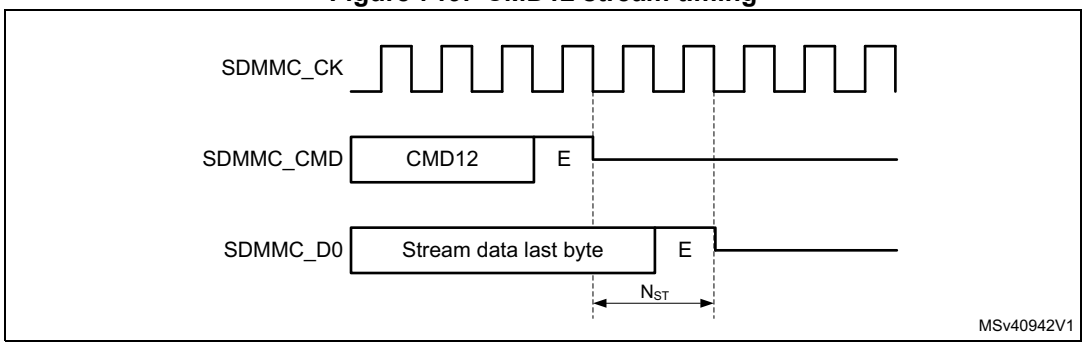
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   a) This causes the CPSM to generate the Abort signal when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   a) The card, when sending data, will stop data transfer 2 cycles after the Stop Transmission command End bit.
   The card when no data is being sent, will not start sending any new data.
   b) The host, when sending data, will send one last data bit followed by an End bit after the Stop Transmission command End bit.
   The host when not sending data, will not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   a) When writing data to the card. On the CMDREND flag FW shall stop writing data to the FIFO. Subsequently the FIFO shall be reset with FIFORST, this will flush the FIFO.
   b) When reading data from the card. On the CMDREND flag FW shall read the remaining data from the FIFO. Subsequently the FIFO shall be reset with FIFORST.
4. When IDMAEN = 1, hardware will take care of the FIFO.
   a) When writing data to the card. On the Abort signal hardware will stop the IDMA and subsequently the FIFO will be flushed.
   b) When reading data from the card. On the Abort signal hardware will instruct the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag will be generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 End bit timing shall be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit shall set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data End bit and command End bit will be aligned.
   a) When DATALENGTH > 5 bytes, Command CMD12 will be waited in the CPSM to be aligned with the data transfer End bit.
   b) When DATALENGHT < 5 bytes, Command CMD12 will be started before and the DPSM will remain in the Wait_S state to align the data transfer end with the CMD12 End bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This will cause the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 719. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 End bit timing shall occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM (DATAEND flag).
   a) The DPSM will not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   a) CMD12 will stop the card sending data.

Note: The SDMMC will not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 End bit shall be sent after the last block End bit.

When writing data to the card the CMD12 End bit shall be sent after the write data block CRC token End bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   a) The DPSM will not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   a) CMD12 will set the card to Idle mode.

When reading data from the card the CMD12 End bit shall be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing.The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM (DATAEND flag).
   a) The DPSM will not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   a) CMD12 will stop the Card sending more data and set the card to Idle mode. Any ongoing block transfer will be aborted by the Card.

Note: The SDMMC will not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

54.5.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state
4. Vcc power supply is allowed to be switched off

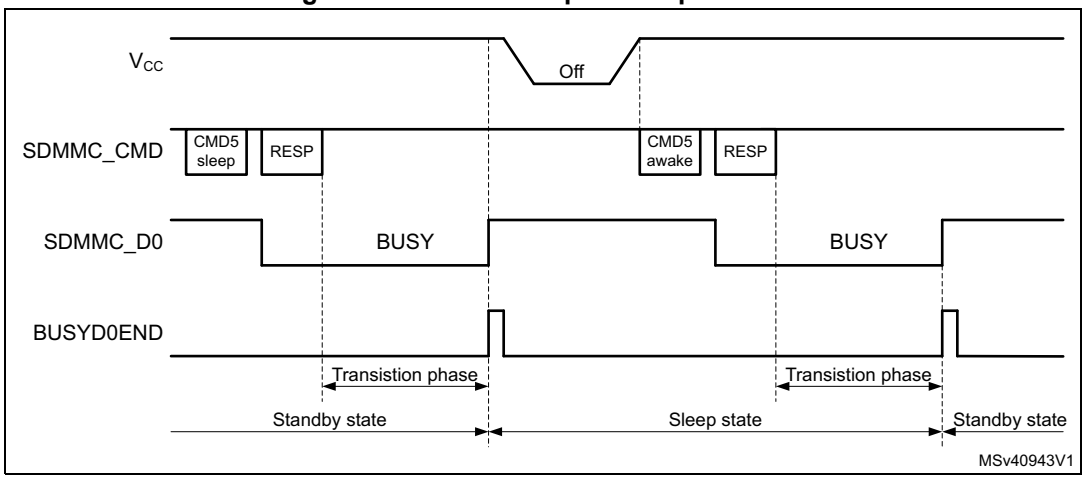
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply is allowed to be switched off only after the Sleep state has been reached. The Vcc supply shall be reinstalled before CMD5 (AWAKE) is sent.

Figure 720. CMD5 Sleep Awake procedure



54.5.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency shall be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV shall be set >1, and the SETCLKRX shall select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a Start bit. Upon detection of a Start bit the card will abort the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (Start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one will successfully send its full interrupt service request response. i.e. If the host sends simultaneously, it will lose sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV shall be set >1, and SETCLKRX shall select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   a) This will cause the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   a) Upon the detection of an interrupt service request response Start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   b) To abort the interrupt mode the host clears the WAITINT register bit, which will cause the host to send an interrupt service request response by its self. Which will move the CPSM to the Receive state.The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response Start bit collision the host will lose the bus access after the Transmission bit.

54.5.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the 2 boot operation functions:

1. Normal boot. (keeping CMD line low)
2. Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

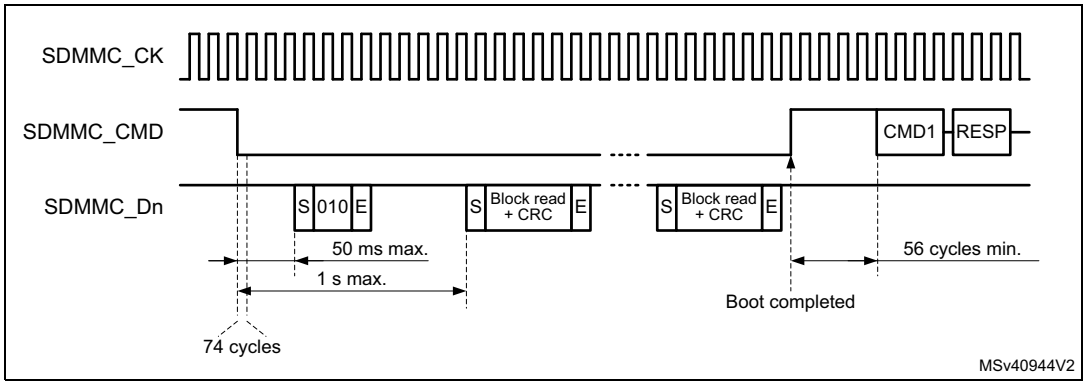
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 721. Normal boot mode operation



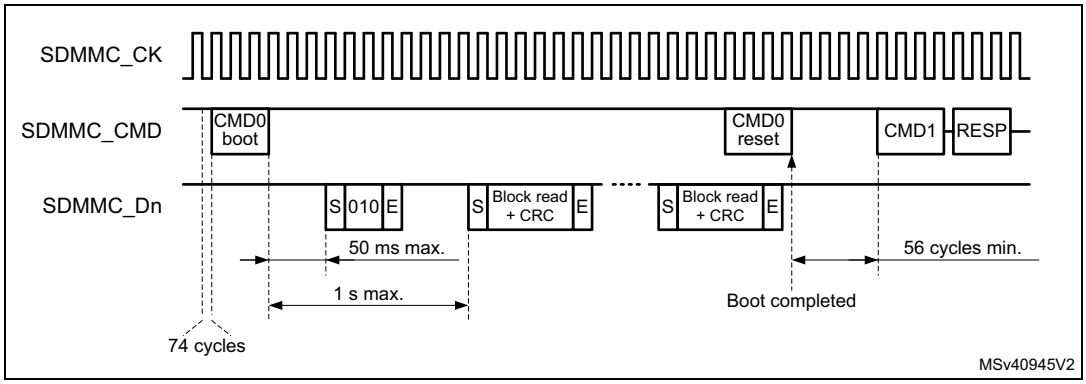
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This will cause:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag will occur.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure will be terminated by FW clearing BOOTEN, which will cause the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   a) If the boot procedure is aborted by FW before all data has been received the CPSM Abort signal will stop data reception and disable the DPSM which will generate an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 722. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

1. Move the SDMMC to power-off state, and reset the card
2. Move the SDMMC to power-on state. This will guarantee the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of

boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This will cause:

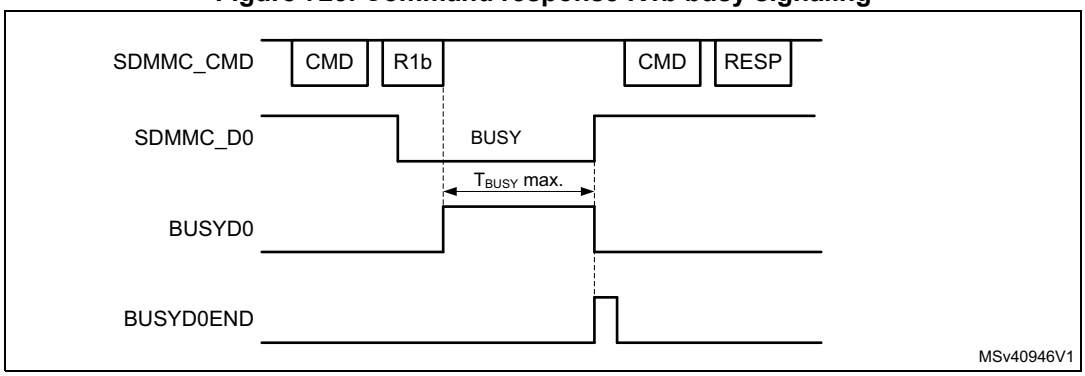
- the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
- the ACK timeout to start.
- DPSM to be enabled.

6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit shall be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag will occur.
8. when all boot data has been received the DATAEND flag will occur.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
9. When last data has been received, read data from the FIFO until FIFO is empty (RXFIFOE = 1) after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit shall be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This will cause the CMDSENT flag to occur 56 cycles after the Command.
    - if the boot procedure is aborted by FW before all data has been received the CPSM Abort signal will stop the data transfer and disable the DPSM which will generate an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

54.5.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 723. Command response R1b busy signaling



The expected maximum busy time shall be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag will be set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  - If BUSYD0 signals not busy, signal busy release to the Firmware
  - If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to the firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

54.5.7 Reset and card cycle power

Reset

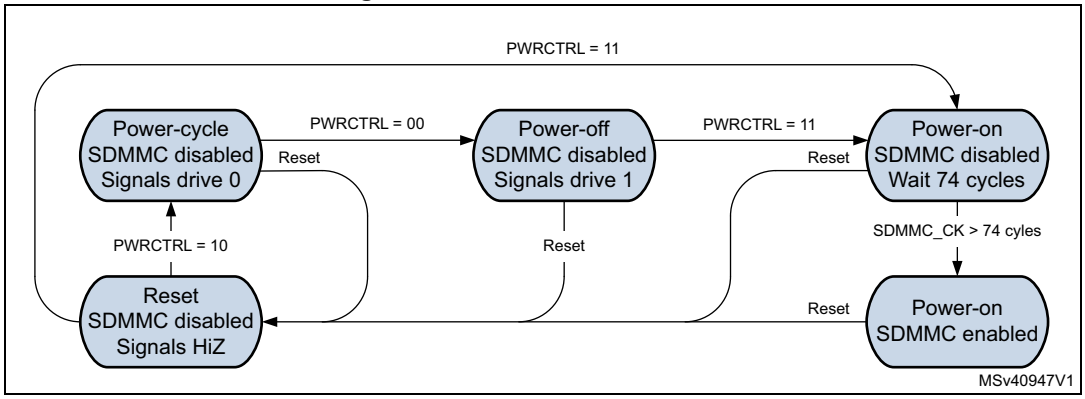
Following reset the SDMMC will be in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC shall be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles will be clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 724.*.

Figure 724. SDMMC state control

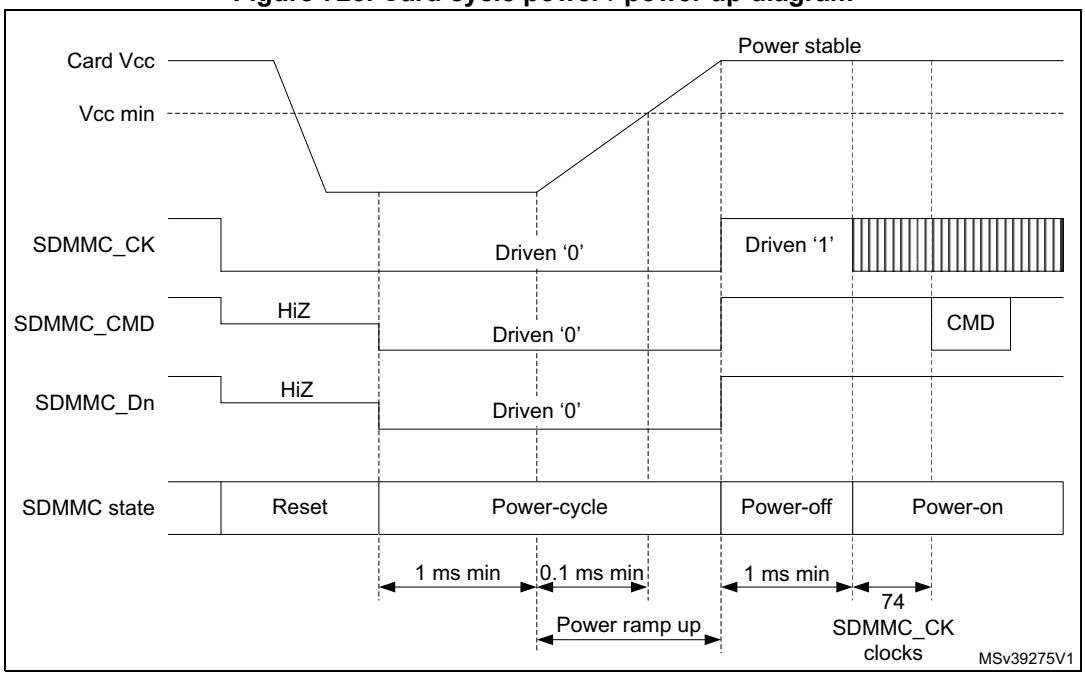


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This will reset the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This will make that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive “1”.
6. After the 1ms delay set the SDMMC to power-on state in which the SDMMC_CK clock will be enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 725. Card cycle power / power up diagram



54.6 Hardware flow control

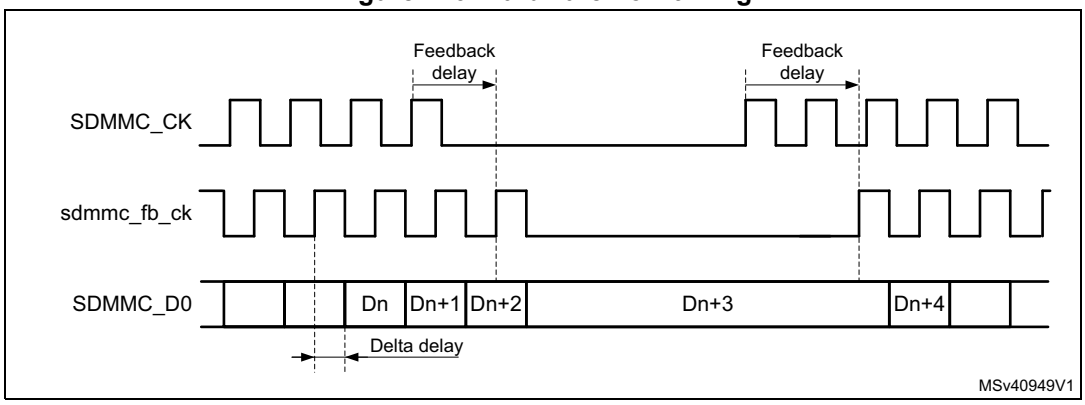
The hardware flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

To enable hardware flow control, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control shall only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control can NOT be used.

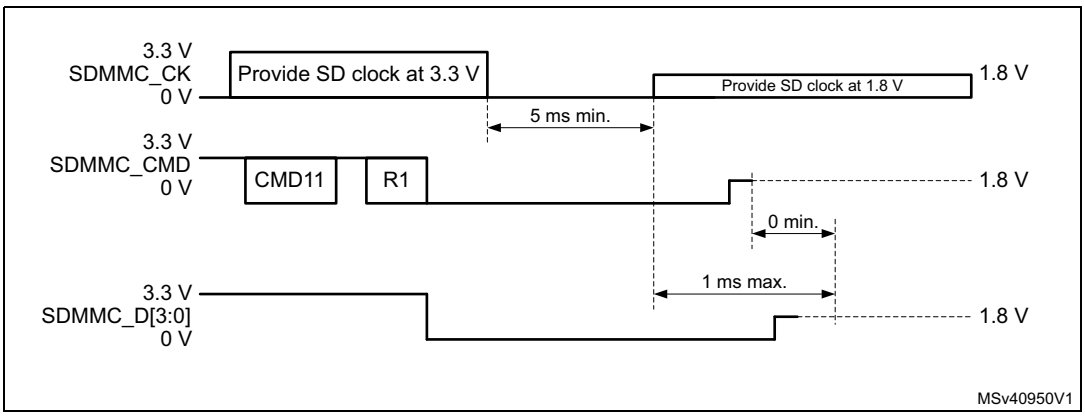
Figure 726. Hardware flow timing



54.7 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 727. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency shall be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   – if the response CRC is pass, the Voltage Switch procedure continues the host will no longer drive the CMD and SDMMC_D[3:0] signals until completion of the

voltage switch sequence. Some cycles after the response the SDMMC_CK will be stopped and the CKSTOP flag will be set.

– if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.

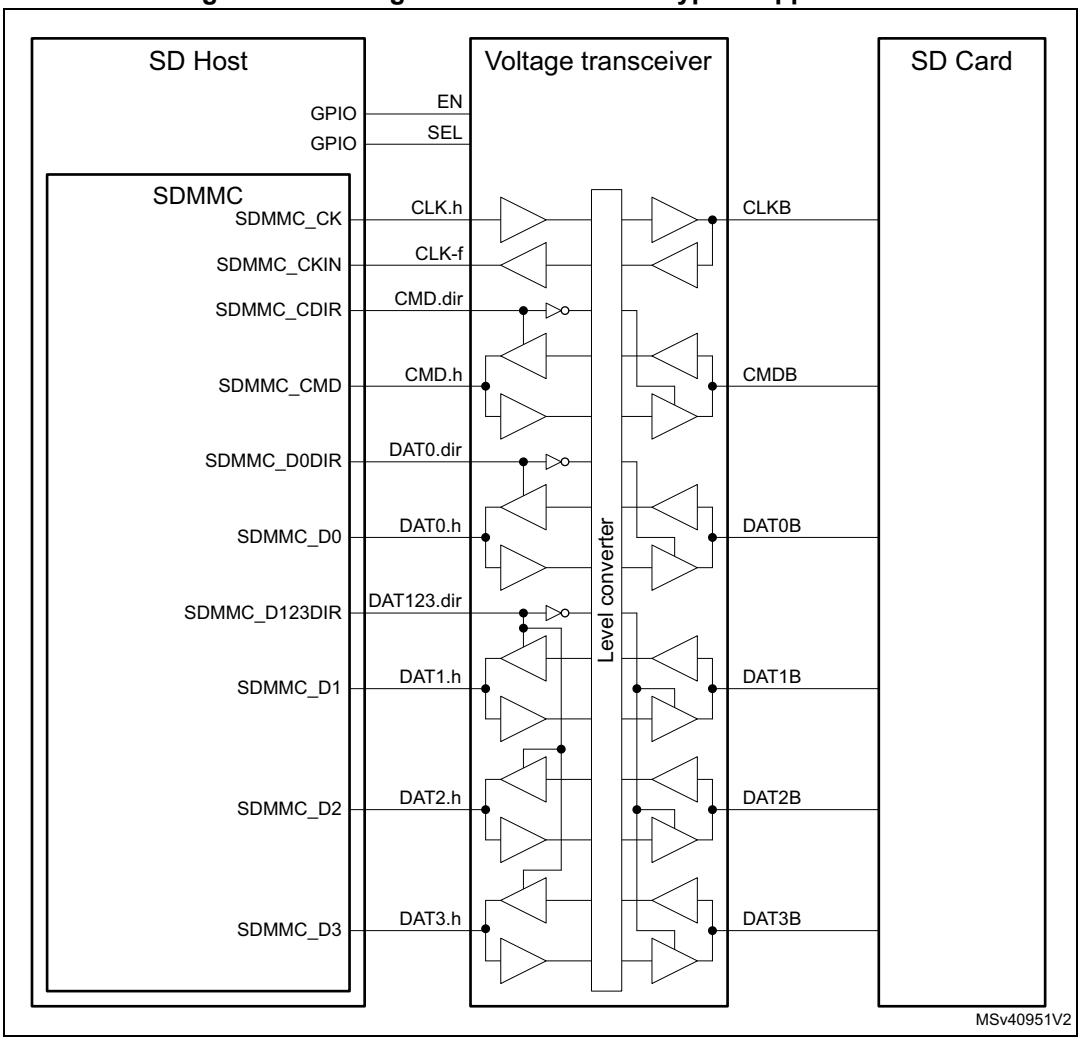
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   – When the BUSYD0 is detected low the host FW will switch the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware will continue to stop the SDMMC_CK by holding it low for at least 5ms.
   – When the BUSYD0 is detected high the host will abort the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC_CK low will begin switching signaling voltage to 1.8V.
7. The host SDMMC hardware after at least 5ms will restart the SDMMC_CK.
8. The card within 1ms from detecting SDMMC_CK transition will drive CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].
9. The host SDMMC hardware, 1ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and generate the VSWEND flag.
10. The host, on the VSWEND flag, will check SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    – When BUSYD0 is detected low, Voltage Switch has failed, the host will cycle power the card power.

The minimum 5ms time to stop the SDMMC_CK is derived from the internal ungated SDMMC_CK clock, which will have a maximum frequency of 25MHz (SD mode), as set by the clock divider CLKDIV. The >5ms time will be counted by $2^{\wedge 12}$ cycles (10.24ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped will be longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which will have a maximum frequency of 25MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC will check the lines after >1ms time which will be counted by $2^{\wedge 9}$ cycles (1.28ms @25MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines will be longer.

The signal voltage level is supported through an external voltage translation transceiver i.e. ST6G3244ME.

Figure 728. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

SDMMC_CKIN feedback input clock

SDMMC_CDIR I/O direction control for the CMD signal.

SDMMC_D0DIR I/O direction control for the SDMMC_D0 signal.

SDMMC_D123DIR I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals.

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

54.8 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and halfword (16-bit) accesses generate an AHB bus error.

54.8.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR POL	VSWI TCHEN	VSWI TCH	PWRCTRL[1:0]	
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver IOs driven as output when direction signal is low.

1: Voltage transceiver IOs driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved. (When written 01, PWRCTRL value will not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write will be ignored, PWRCTRL value will keep 11.

54.8.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

The SDMMC_CLKCR register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_ EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: sdmmc_fb_ck tuned feedback clock selected as receive clock.

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed mode selection between DS, HS, SDR12, SDR25 and SDR50, DDR50, SDR104

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25 bus speed mode selected

1: SDR50, DDR50, SDR104 bus speed mode selected.

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate shall only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate shall only be selected with clock division >1. (CLKDIV > 0)

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, please see SDMMC status register definition in *Section 54.8.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) & DDR = 0:

0: - Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: - Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) & DDR = 1:

0: - Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: - Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: etc..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: etc..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: 1 While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

2 The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

3 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the ReadWait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

54.8.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

The SDMMC_ARGR register contains a 32-bit command argument, which is sent to a card as part of a command message.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDARG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDARG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

54.8.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

The SDMMC_CMDR register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS=0.
CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF=1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled
1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
0: Normal boot mode procedure selected
1: Alternative boot mode procedure selected.

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM will not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
If this bit is set, the CPSM is enabled.
When DTEN = 1, no command will be transfered nor boot procedure will be started. CPSMEN is cleared to 0.
During ReadWait with SDMMC_CK stopped no command will be sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
If this bit is cleared in the CPSM Wait state, will cause the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: Short response, expect CMDREND flag (No CRC)
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals Abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the Abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software will distinguish the type of response according to the send command.

54.8.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

The SDMMC_RESPCMDR register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bit field. Contains the command index of the last command response received.

54.8.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

The SDMMC_RESP1/2/3/4R registers contain the status of a card, which is part of the received response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CARDSTATUSx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CARDSTATUSx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CARDSTATUSx[31:0]**: see *Table 457*

The card status size is 32 or 128 bits, depending on the response type.

Table 457. Response type and SDMMC_RESPxR registers

Register(1)	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0](2)

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

54.8.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

The SDMMC_DTIMER register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDMMC_DTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATATIME[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATATIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

54.8.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

The SDMMC_DLENR register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATALENGTH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data will be transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command will be transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

54.8.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

The SDMMC_DCTRL register control the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, will flush any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit will only take effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit automatically will be cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the READ_WAIT state to the WAIT_R or IDLE state.

0: No read wait stop.

1: Enable for read wait stop when DPSM is in the READ_WAIT state.

Bit 8 **RWSTART**: Read wait start

If this bit is set, read wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
Define the data block length when the block data transfer mode is selected:
0000: (0 decimal) lock length = 2^0 = 1 byte
0001: (1 decimal) lock length = 2^1 = 2 bytes
0010: (2 decimal) lock length = 2^2 = 4 bytes
0011: (3 decimal) lock length = 2^3 = 8 bytes
0100: (4 decimal) lock length = 2^4 = 16 bytes
0101: (5 decimal) lock length = 2^5 = 32 bytes
0110: (6 decimal) lock length = 2^6 = 64 bytes
0111: (7 decimal) lock length = 2^7 = 128 bytes
1000: (8 decimal) lock length = 2^8 = 256 bytes
1001: (9 decimal) lock length = 2^9 = 512 bytes
1010: (10 decimal) lock length = 2^{10} = 1024 bytes
1011: (11 decimal) lock length = 2^{11} = 2048 bytes
1100: (12 decimal) lock length = 2^{12} = 4096 bytes
1101: (13 decimal) lock length = 2^{13} = 8192 bytes
1110: (14 decimal) lock length = 2^{14} = 16384 bytes
1111: (15 decimal) reserved
When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (Any remain data will not be transfered.)
When DDR = 1, DBLOCKSIZE = 0000 shall not be used. (No data will be transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
00: Block data transfer ending on block count.
01: SDIO multibyte data transfer.
10: *e*•MMC Stream data transfer. (WIDBUS shall select 1-bit wide bus mode)
11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: From host to card.
1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by Hardware when data transfer completes.
This bit shall only be used to transfer data when no associated data transfer command is used, i.e. shall not be used with SD or *e*•MMC cards.
0: Do not start data transfer without CPSM data transfer command.
1: Start data transfer without CPSM data transfer command.

54.8.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

The SDMMC_DCNTR register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then

moves to the Idle state and when there has been no error, the data status end flag (DATAEND) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:16]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATACOUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register should be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

54.8.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

The SDMMC_STAR register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and de-asserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
0: card signals not busy on SDMMC_D0.
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active, i.e. not in Idle state
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active, i.e. not in Idle state
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received
(CRC check passed) and DPSM moves to the READWAIT state. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
(data counter, DATACOUNT is zero and no errors occur). Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags shall be masked in SDMMC_MASKR when using IDMA mode.

54.8.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

The SDMMC_ICR register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOP C	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	D ABOR TC	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

54.8.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

The interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			rw		rw	rw	rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVERR IE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage Switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by Voltage Switch clock stopped.
0: Voltage Switch clock stopped interrupt disabled
1: Voltage Switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment Fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment Fail.
0: Acknowledgment Fail interrupt disabled
1: Acknowledgment Fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO Mode interrupt received interrupt disabled
1: SDIO Mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

54.8.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

The SDMMC_ACKTIMER register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from the SDMMC_ACKTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACKTIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

54.8.15 SDMMC data FIFO register (SDMMC_FIFOR)

Address offset: 0x080 to 0x0BC

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFODATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data
This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).
The FIFO data occupies 16 entries of 32-bit words.

54.8.16 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMAB ACT	IDMAB MODE	IDMA EN
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IDMABACT**: Double buffer mode active buffer indication

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). When IDMA is enabled this bit is toggled by hardware.

0: When IDMA is enabled, uses buffer0 and firmware write access to IDMABASE0 is prohibited.

1: When IDMA is enabled, uses buffer1 and firmware write access to IDMABASE1 is prohibited.

Bit 1 **IDMABMODE**: Buffer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Single buffer mode.

1: Double buffer mode.

Bit 0 **IDMAEN**: IDMA enable

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: IDMA disabled

1: IDMA enabled

54.8.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

The SDMMC_IDMABSIZER register contains the buffers size when in double buffer configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IDMABNDT[7:0]								Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:5 **IDMABNDT[7:0]**: Number of bytes per buffer

This 8-bit value shall be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.

Example: IDMABNDT = 0x01: buffer size = 8 words = 32 bytes.

Example: IDMABNDT = 0x80: buffer size = 1024 words = 4 Kbyte

These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0 Reserved, must be kept at reset value.

54.8.18 SDMMC IDMA buffer 0 base address register (SDMMC_IDMABASE0R)

Address offset: 0x058

Reset value: 0x0000 0000

The SDMMC_IDMABASE0R register contains the memory buffer base address in single buffer configuration and the buffer 0 base address in double buffer configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABASE0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **IDMABASE0[31:0]**: Buffer 0 memory base address bits [31:2], shall be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 0 is inactive (IDMABACT = ‘1’).

54.8.19 SDMMC IDMA buffer 1 base address register (SDMMC_IDMABASE1R)

Address offset: 0x05C

Reset value: 0x0000 0000

The SDMMC_IDMABASE1R register contains the double buffer configuration second buffer memory base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABASE1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **IDMABASE1[31:0]**: Buffer 1 memory base address, shall be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1) and memory buffer 1 is inactive (IDMABACT = ‘0’).

54.8.20 SDMMC register map

The following table summarizes the SDMMC registers.

Table 458. SDMMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**SDMMC_ POWER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	**SDMMC_ CLKCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	**SDMMC_ ARGR**	CMDARG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**SDMMC_ CMDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																
0x010	**SDMMC_ RESPCMDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	**SDMMC_ RESP1R**	CARDSTATUS1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**SDMMC_ RESP2R**	CARDSTATUS2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**SDMMC_ RESP3R**	CARDSTATUS3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**SDMMC_ RESP4R**	CARDSTATUS4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**SDMMC_ DTIMER**	DATATIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**SDMMC_ DLENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 458. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**SDMMC_ DCTRLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]				DTMODE		DTDIR	DTEN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	**SDMMC_ DCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	**SDMMC_ STAR**	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**SDMMC_ ICR**	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWEDNDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**SDMMC_ MASKR**	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0
0x040	**SDMMC_ ACKTIMER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044 - 0x04C	**Reserved**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x050	**SDMMC_ IDMACTRLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABACT	IDMABMODE	IDMAEN
	Reset value																														0	0	0
0x054	**SDMMC_ IDMABSIZER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[7:0]								Res.	Res.	Res.	Res.	Res.
	Reset value																				0	0	0	0	0	0	0	0					
0x058	**SDMMC_ IDMABASE0R**	IDMABASE0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	**SDMMC_ IDMABASE1R**	IDMABASE1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060 - 0x07C	**Reserved**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 458. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	**SDMMC_ FIFOR**	FIFODATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084 - x3FC	**Reserved**	Res.																															

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

55 FD controller area network (FDCAN)

55.1 Introduction

The controller area network (CAN) subsystem (see *Figure 729*) consists of two CAN modules, a shared message RAM and a clock calibration unit. Refer to the memory map for the base address of each of these four parts.

Both modules (FDCAN1 and FDCAN2) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

In addition, the first CAN module FDCAN1 supports time triggered CAN (TTCAN), specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. The FDCAN1 contains additional registers, specific to the time triggered feature. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

A 10 Kbyte message RAM implements filters, receive FIFOs, receive buffers, transmit event FIFOs, transmit buffers (and triggers for TTCAN). This message RAM is shared between the FDCAN1 and FDCAN2 modules.

The common clock calibration unit is optional. It can be used to generate a calibrated clock for both FDCAN1 and FDCAN2 from the HSI internal RC oscillator and the PLL, by evaluating CAN messages received by the FDCAN1.

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 459* and *Table 460*.

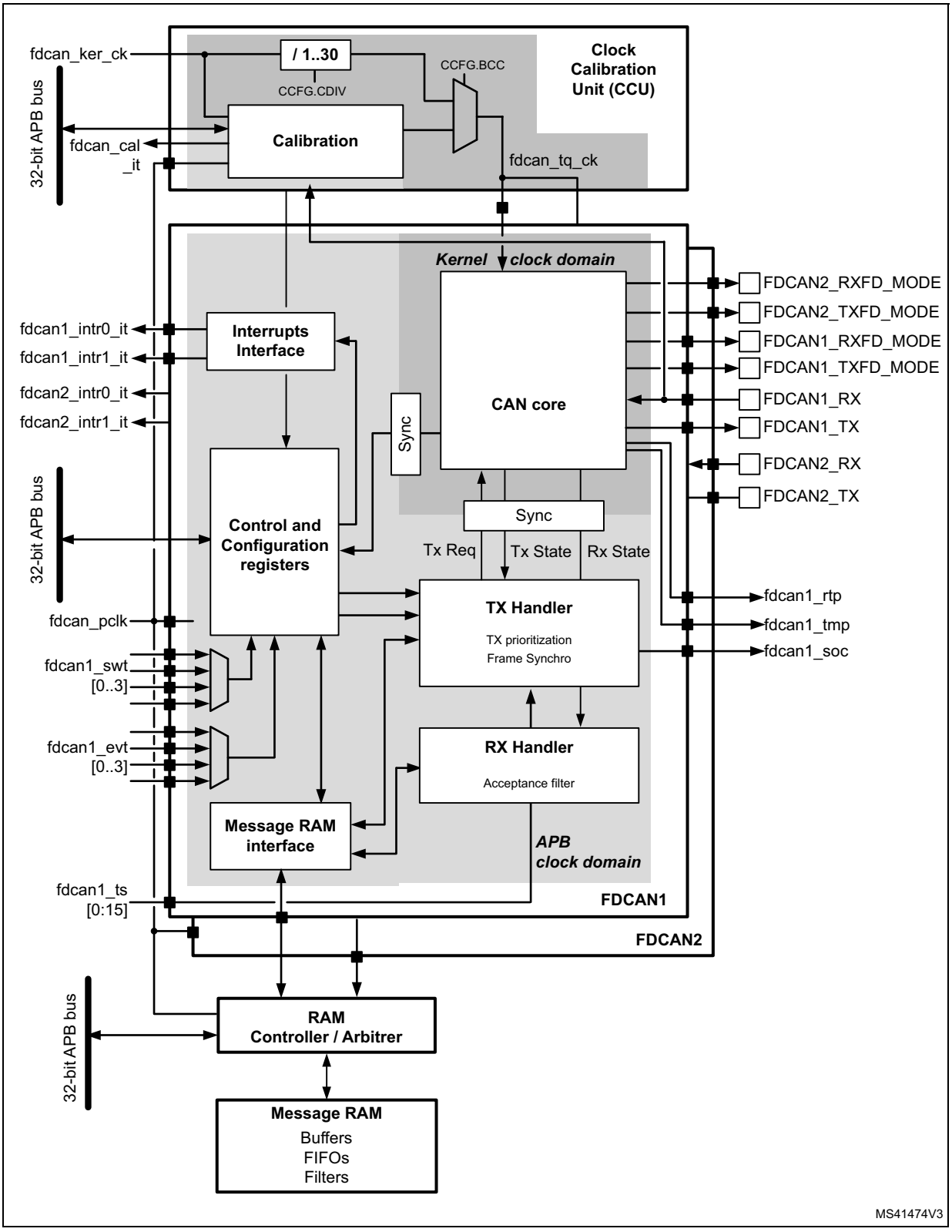
Table 459. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan_cal_it	Digital output	FDCAN calibration interrupt
fdcan1_intr0_it	Digital output	FDCAN1 interrupt0
fdcan1_intr1_it		FDCAN1 interrupt1
fdcan2_intr0_it		FDCAN2 interrupt0
fdcan2_intr1_it		FDCAN2 interrupt1
fdcan1_swt[0:3]	Digital input	Stop watch trigger input
fdcan1_evt[0:3]		Event trigger input
fdcan1_ts[0:15]		External timestamp vector
fdcan1_soc	Digital output	Start of cycle pulse
fdcan1_rtp		Register time mark pulse
fdcan1_tmp		Trigger time mark pulse

Table 460. CAN subsystem I/O pins

Name	Type	Description
FDCAN1_RX	Digital input	FDCAN1 receive pin
FDCAN1_TX	Digital output	FDCAN1 transmit pin
FDCAN2_RX	Digital input	FDCAN2 receive pin
FDCAN2_TX	Digital output	FDCAN2 transmit pin
FDCAN1_TXFD_MODE	Digital output	Indicates if the current TX CAN1 frame is in FD mode
FDCAN1_RXFD_MODE	Digital output	Indicates if the current RX CAN1 frame is in FD mode
FDCAN2_TXFD_MODE	Digital output	Indicates if the current TX CAN2 frame is in FD mode
FDCAN2_RXFD_MODE	Digital output	Indicates if the current RX CAN2 frame is in FD mode

Figure 729. CAN subsystem

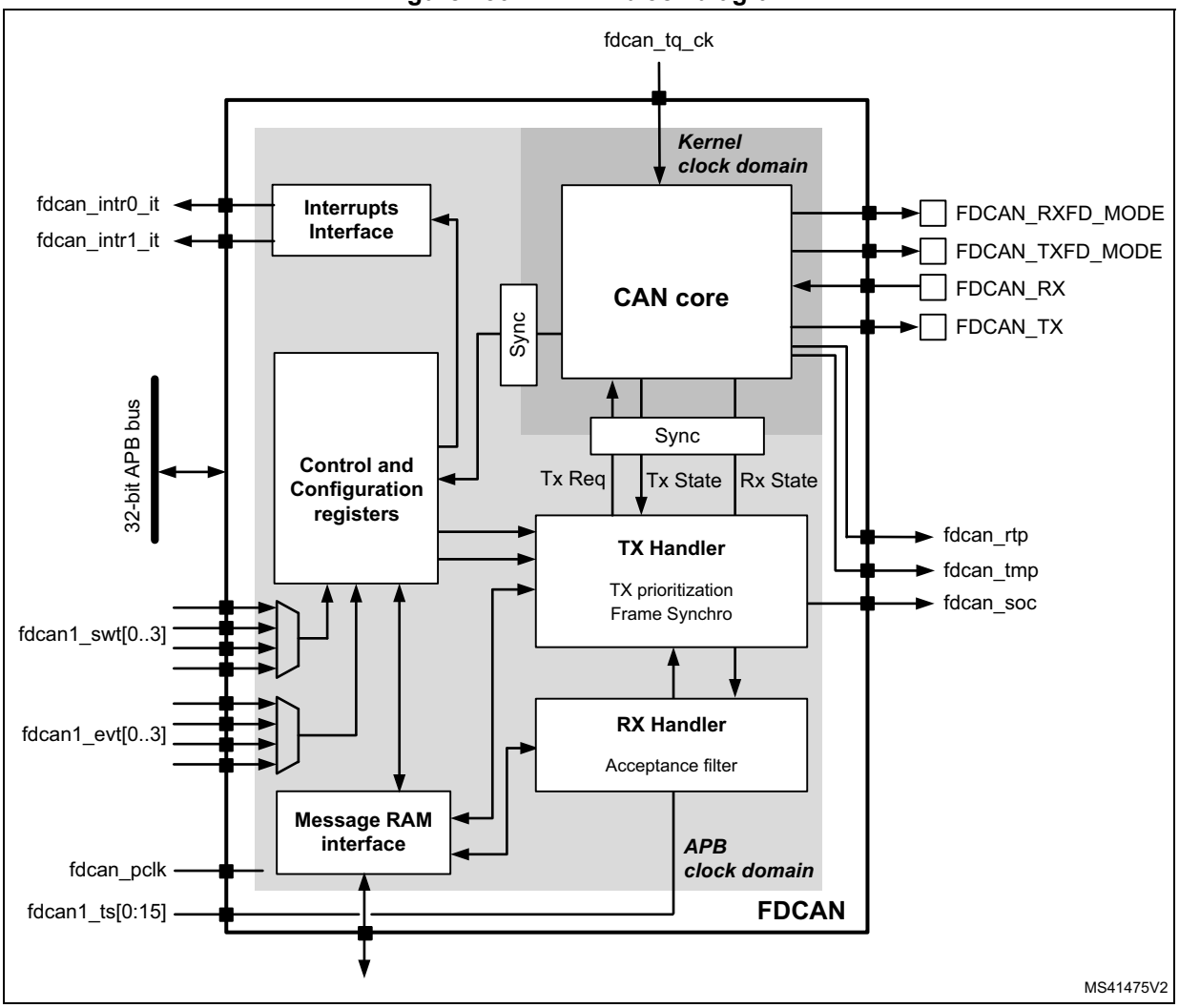


55.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015, -4
- CAN FD with max. 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware (FDCAN1 only)
- Event synchronized time-triggered communication supported (FDCAN1 only)
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two configurable receive FIFOs
- Separate signaling on reception of high priority messages
- Up to 64 dedicated receive buffers
- Up to 32 dedicated transmit buffers
- Configurable transmit FIFO /queue
- Configurable transmit event FIFO
- Both FDCAN1 and FDCAN2 modules share the same message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

55.3 FDCAN functional description

Figure 730. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines fdcan_intr0_it and fdcan_intr1_it. By programming EINT0 and EINT1 bits in FDCAN_ILE register, the interrupt lines can be enabled or disabled separately.

CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

The sync block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

Controls the message transfer from the message RAM to the CAN core. A maximum of 32 Tx buffers can be configured for transmission. Tx buffers can be used as dedicated Tx buffers, as Tx FIFO, part of a Tx queue, or as a combination of them. A Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

On FDCAN1, the Tx handler also implements the frame synchronization entity (FSE) which controls time-triggered communication according to ISO11898-4. It synchronizes itself with the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the message RAM. Stop watch trigger, event trigger, and time mark interrupt are synchronization interfaces.

Rx handler

Controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, each of configurable size, and up to 64 dedicated Rx buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx buffer, in contrast to a receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

APB Interface

Connects the FDCAN to the APB bus.

Message RAM Interface

Connects the FDCAN access to an external 10 Kbytes message RAM through a RAM controller/arbiter.

55.3.1 Operating modes

Software initialization

Software initialization is started by setting INIT bit in FDCAN_CCCR register, either by software or by a hardware reset, or by going Bus_Off. While INIT bit in FDCAN_CCCR register is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output FDCAN_TX is recessive (high). The counters of the error management logic (EML) are unchanged. Setting INIT bit in FDCAN_CCCR does not change any configuration register. Clearing INIT bit in FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when both INIT bit in FDCAN_CCCR register and CCE bit in FDCAN_CCCR register are set.

CCE bit in FDCAN_CCCR register can only be set/cleared while INIT bit in FDCAN_CCCR is set. CCE bit in FDCAN_CCCR register is automatically cleared when INIT bit in FDCAN_CCCR is cleared.

The following registers are reset when CCE bit in FDCAN_CCCR register is set:

- FDCAN_HPMS - high priority message status
- FDCAN_ RXF0S - Rx FIFO 0 status
- FDCAN_RXF1S - Rx FIFO 1 status
- FDCAN_TXFQS - Tx FIFO/queue status
- FDCAN_TXBRP - Tx buffer request pending
- FDCAN_TXBTO - Tx buffer transmission occurred
- FDCAN_TXBCF - Tx buffer cancellation finished
- FDCAN_TXEFS - Tx event FIFO status
- FDCAN_TTOST - TT operation status (FDCAN1 only)
- FDCAN_TTLGT - TT local & global time, only global time FDCAN_TTLGT.GT is reset (FDCAN1 only)
- FDCAN_TTCTC - TT cycle time & count (FDCAN1 only)
- FDCAN_TTCSM - TT cycle sync mark (FDCAN1 only)

The timeout counter value TOC bit in FDCAN_TOCV register is preset to the value configured by TOP bit in FDCAN_TOCC register when CCE bit in FDCAN_CCCR is set.

In addition the state machines of the Tx handler and Rx handler are held in idle state while CCE bit in FDCAN_CCCR is set.

The following registers can be written only when CCE bit in FDCAN_CCCR register is cleared:

- FDCAN_TXBAR - Tx buffer add request
- FDCAN_TXBCR - Tx buffer cancellation request

TEST bit in FDCAN_CCCR and MON bit in FDCAN_CCCR can only be set by software while both INIT bit and CCE bit in FDCAN_CCCR register are set. Both bits may be reset at any time. DAR bit in FDCAN_CCCR can only be set/cleared while both INIT bit in FDCAN_CCCR and CCE bit in FDCAN_CCCR are set.

Normal operation

The FDCAN1 default operating mode after hardware reset is event-driven CAN communication without time triggers (FDCAN_TTOCF.OM = 00). It is required that both INIT bit and CCE bit in FDCAN_CCCR register are set before the TT operation mode can be changed.

Once the FDCAN is initialized and INIT bit in FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol, first the long frame mode (LFM) where the data field of a CAN frame may be longer that eight bytes. The second variant is the fast frame mode (FFM) where control field, data field, and CRC field of a CAN frame are

transmitted with a higher bitrate than the beginning and the end of the frame. Fast frame mode can be used in combination with long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bitrate inside this CAN FD frame is switched. A CAN FD bitrate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the M_TTCAN receives a frame with FDF recessive and res recessive, it will signal a protocol exception event by setting bit FDCAN_PSR.PXE. When protocol exception handling is enabled (FDCAN_CCCR.PXHD = 0), this causes the operation state to change from receiver (FDCAN_PSR.ACT = 10) to Integrating (FDCAN_PSR.ACT = 00) at the next sample point. In case protocol exception handling is disabled (FDCAN_CCCR.PXHD = 1), the FDCAN will treat a recessive res bit as a form error and will respond with an error frame.

CAN FD operation is enabled by programming FDCAN_CCCR.FDOE. If FDCAN_CCCR.FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of classic CAN frames is always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via bit FDF in the respective Tx buffer element. With FDCAN_CCCR.FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted, even if bit FDF of a Tx buffer element is set. FDCAN_CCCR.FDOE and FDCAN_CCCR.BRSE can only be changed while FDCAN_CCCR.INIT and FDCAN_CCCR.CCE are both set.

With FDCAN_CCCR.FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in classic CAN format. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 0, only bit FDF of a Tx buffer element is evaluated. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 1, transmission of CAN FD frames with bitrate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bitrate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bitrate switching option for transmissions.
- During system startup all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 461*.

Table 461. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing will be switched inside the frame, after the BRS (bitrate Switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the nominal CAN bit timing is used as defined by the bit timing and prescaler register FDCAN_NBTP. In the following FDCAN data phase, the fast CAN bit timing is used as defined by the fast bit fiming and prescaler register FDCAN_DBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bitrate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with a FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (tq), the bitrate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long frames and CAN FD fast frames, the value of the bit ESI (error status Indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case a FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx buffer or Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of a FDCAN transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bitrate in the data phase of a FDCAN frame is limited by the transceivers loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bitrates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit FDCAN_DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by FDCAN_TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq (minimum time quantum, that is one period of fdcan_tq_ck clock).

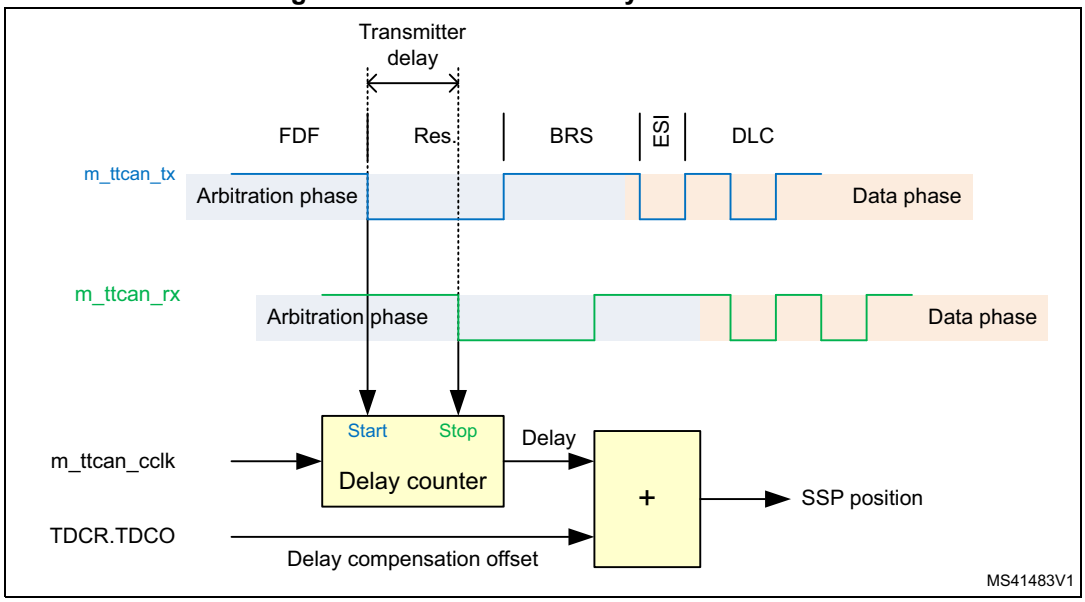
FDCAN_PSR.TDCV shows the actual transmitter delay compensation value, it is cleared when FDCAN_CCCR.INIT is set and is updated at each transmission of an FD frame while FDCAN_DBTP.TDC is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from m_ttcan_tx to m_ttcan_rx and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO has to be less than six bit times in the data phase.
- The sum of the measured delay from m_ttcan_tx to m_ttcan_rx and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value (127 mtq) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking received bits at the SSPs

If transmitter delay compensation is enabled by programming FDCAN_DBTP.TDC = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mtq.

Figure 731. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a to early SSP position) the use of a transmitter delay compensation filter window can be enabled by

programming FDCAN_TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on m_ttcan_rx, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least FDCAN_TDCR.TDCF and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (FDCAN_ECR.REC, FDCAN_ECR.TEC) are frozen while error logging (FDCAN_ECR.CEL) is active. The software can set the FDCAN into restricted operation mode by setting bit FDCAN_CCCR.ASM. The bit can only be set by software when both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set to 1. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler was not able to read data from the message RAM in time. To leave restricted operation mode, the software has to reset FDCAN_CCCR.ASM.

The restricted operation mode can be used in applications that adapt themselves to different CAN bitrates. In this case the application tests different bitrates and leaves the restricted operation mode after it has received a valid frame.

FDCAN_CCCR.ASM is also controlled by the clock calibration unit. Wen the clock calibration process is enabled, the restricted operation mode is entered and the FDCAN_CCR.ASM bit is set. Once the calibration is completed, FDCAN_CCR.ASM bit is cleared.

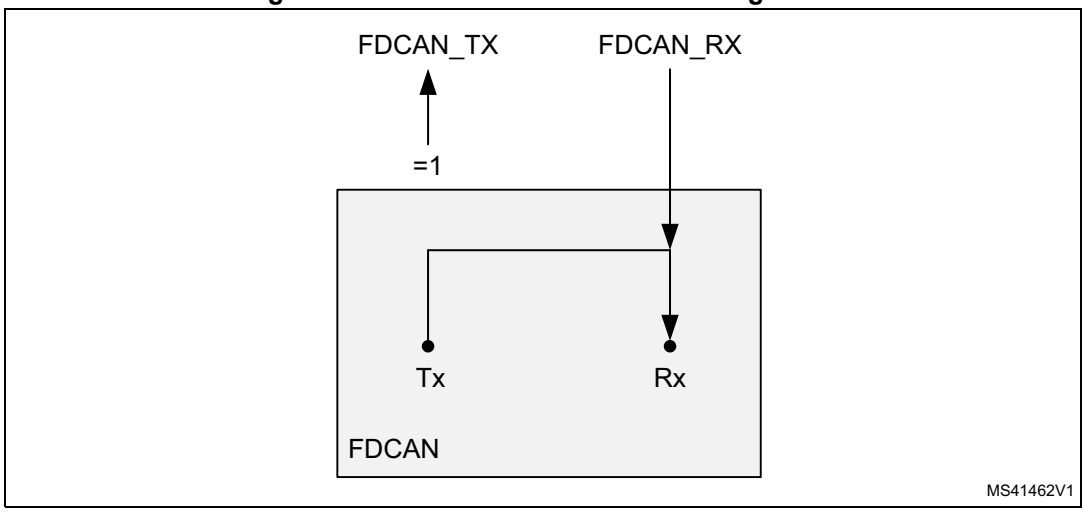
Note: The restricted operation mode must not be combined with the loop back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting FDCAN_CCCR.MON bit or when error level S3 (FDCAN_TTOST.EL = 11) is entered. In bus monitoring mode (For more details please refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode register FDCAN_TXBRP is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 732* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 732. Pin control in bus monitoring mode



Disabled automatic retransmission (DAR) mode

According to the CAN Specification (see ISO11898-1, 6.3.3 recovery management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled.

To support time-triggered communication as described in ISO 11898-1: 2015, chapter 9.2, the automatic retransmission may be disabled via FDCAN_CCCR.DAR.

Frame transmission in disabled automatic retransmission (DAR) mode

In DAR mode all transmissions are automatically canceled after they started on the CAN bus. A Tx buffer Tx request pending bit FDCAN_TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set
- Arbitration loss or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx not set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power down (Sleep mode)

The FDCAN can be set into power down mode controlled by clock stop request input via register FDCAN_CCCR.CSR. As long as the clock stop request is active, bit FDCAN_CCCR.CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until bus idle state is detected. Then the FDCAN sets FDCAN_CCCR.INIT to 1 to prevent any further CAN transfers. Now the FDCAN acknowledges that it is ready for power down by setting FDCAN_CCCR.CSA to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to FDCAN_CCCR.INIT will have no effect. Now the module clock inputs may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC control register flag FDCAN_CCCR.CSR. The FDCAN will acknowledge this by resetting FDCAN_CCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit FDCAN_CCCR.INIT.

Test modes

To enable write access to FDCAN Test register (see *Section 55.4.4 on page 2493*), bit FDCAN_CCCR.TEST has to be set to 1, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming FDCAN_TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the FDCAN bit timing and it can drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from FDCAN_TEST.RX. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there may be a delay of several APB clock periods between writing to FDCAN_TEST.TX until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via FDCAN_TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop back mode

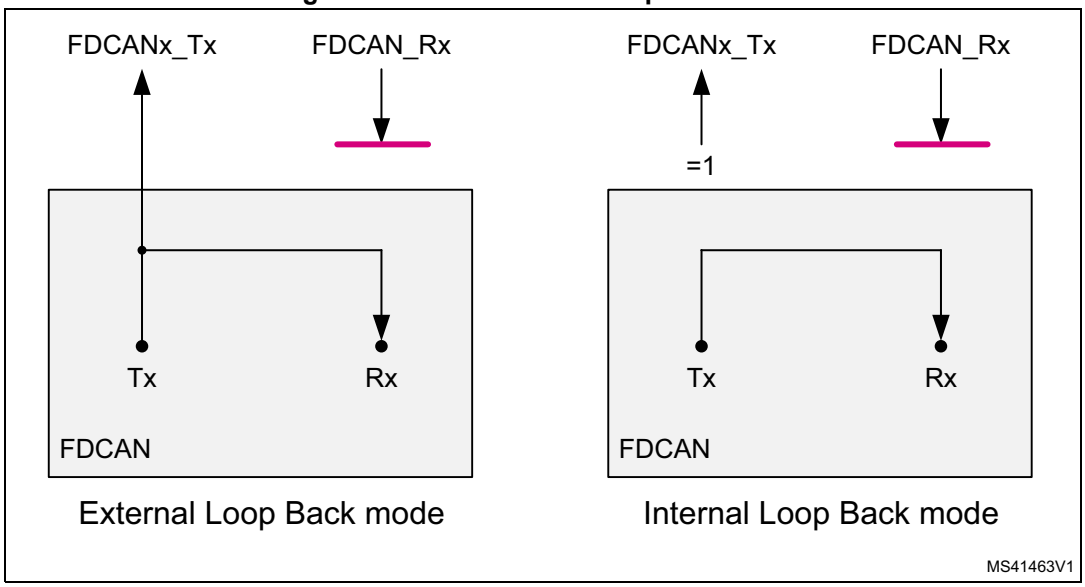
The FDCAN can be set in external loop back mode by programming FDCAN_TEST.LBCK to 1. In loop back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 733* (left side) shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop back mode

Internal loop back mode is entered by programming bits FDCAN_TEST.LBCK and FDCAN_CCCR.MON to 1. This mode can be used for a “Hot Selftest”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 733* (right side) shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop back mode.

Figure 733. Pin control in loop back mode



Application watchdog (FDCAN1 only)

The application watchdog is served by reading register FDCAN_TTOST. When the application watchdog is not served in time, bit FDCAN_TTOST.AWE is set, all TTCAN communication is stopped, and the FDCAN1 is set into bus monitoring mode.

The TT application watchdog can be disabled by programming the application watchdog limit FDCAN_TTOCF.AWL to 0x00. The TT application watchdog should not be disabled in a TTCAN application program

Timestamp generation

For timestamp generation the FDCAN supplies a 16-bit wraparound counter. A prescaler FDCAN_TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 … 16). The counter is readable via FDCAN_TSCV.TCV. A write access to register FDCAN_TSCV resets the counter to 0. When the timestamp counter wraps around interrupt flag FDCAN_IR.TSW is set.

On start of frame reception/transmission the counter value is captured and stored into the timestamp section of a Rx buffer/Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit FDCAN_TSCC.TSS, a 16-bit timestamp can be used.

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as down-counter and uses the same prescaler controlled by FDCAN_TSCC.TCP as the timestamp counter. The timeout counter is configured via register FDCAN_TOCC. The actual counter value can be read from FDCAN_TOCV.TOC. The timeout counter can only be started while FDCAN_CCCR.INIT = 0. It is stopped when FDCAN_CCCR.INIT = 1, e.g. when the FDCAN enters Bus_Off state.

The operation mode is selected by FDCAN_TOCC.TOS. When operating in Continuous mode, the counter starts when FDCAN_CCCR.INIT is reset. A write to FDCAN_TOCV

presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

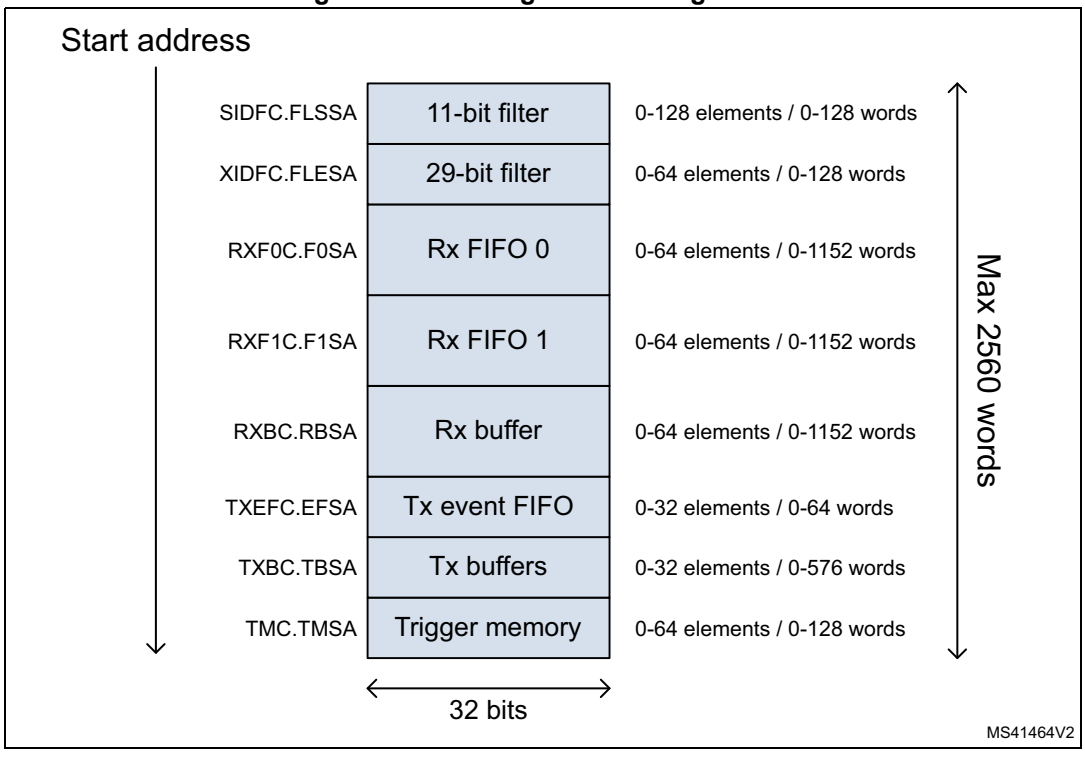
When the counter reaches 0, interrupt flag FDCAN_IR.TOO is set. In Continuous mode, the counter is immediately restarted at FDCAN_TOCC.TOP.

Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore the point in time where the timeout counter is decremented may vary due to the synchronization/re-synchronization mechanism of the CAN core. If the baudrate switch feature in FDCAN is used, the timeout counter is clocked differently in arbitration and data fields.

55.3.2 Message RAM

The message RAM has a width of 32 bits. The FDCAN module can be configured to allocate up to 2560 words in the message RAM. It is not necessary to configure each of the sections listed in *Figure 734*, nor is there any restriction with respect to the sequence of the sections.

Figure 734. Message RAM configuration



When the FDCAN addresses the message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The FDCAN does not check for erroneous configuration of the message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to Rx buffers or to 1 of the two Rx FIFOs, as well as the Rx FIFO put and get Indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx buffer, Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (FDCAN_GFC)
- Standard ID filter configuration (FDCAN_SIDFC)
- Extended ID filter configuration (FDCAN_XIDFC)
- Extended ID AND Mask (FDCAN_XIDAM)

Depending on the configuration of the filter element (SFEC / EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx buffer
- Store received frame in Rx buffer and generate pulse at filter event pin
- Reject received frame
- Set high priority message interrupt flag FDCAN_IR.HPM
- Set high priority message interrupt flag FDCAN_IR.HPM and store received frame in FIFO 0 or FIFO 1
- Set high priority message interrupt flag FDCAN_IR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in portions of 32 bit to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact:

- **Rx buffer**
  New data flag of matching Rx buffer is not set, but Rx buffer (partly) overwritten with received data. For error type see FDCAN_PSR.LEC and FDCAN_PSR.DLEC.
- **Rx FIFO**
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly)

overwritten with received data. For error type see FDCAN_PSR.LEC and FDCAN_PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID / SF2ID and EF1ID / EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT = 00: The message ID of received frames is AND-ed with the extended ID AND Mask (FDCAN_XIDAM) before the range filter is applied
- EFT = 11: The extended ID AND Mask (FDCAN_XIDAM) is not used for range filtering

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID / EF1ID is used as message ID filter, while SF2ID / EF2ID is used as filter mask.

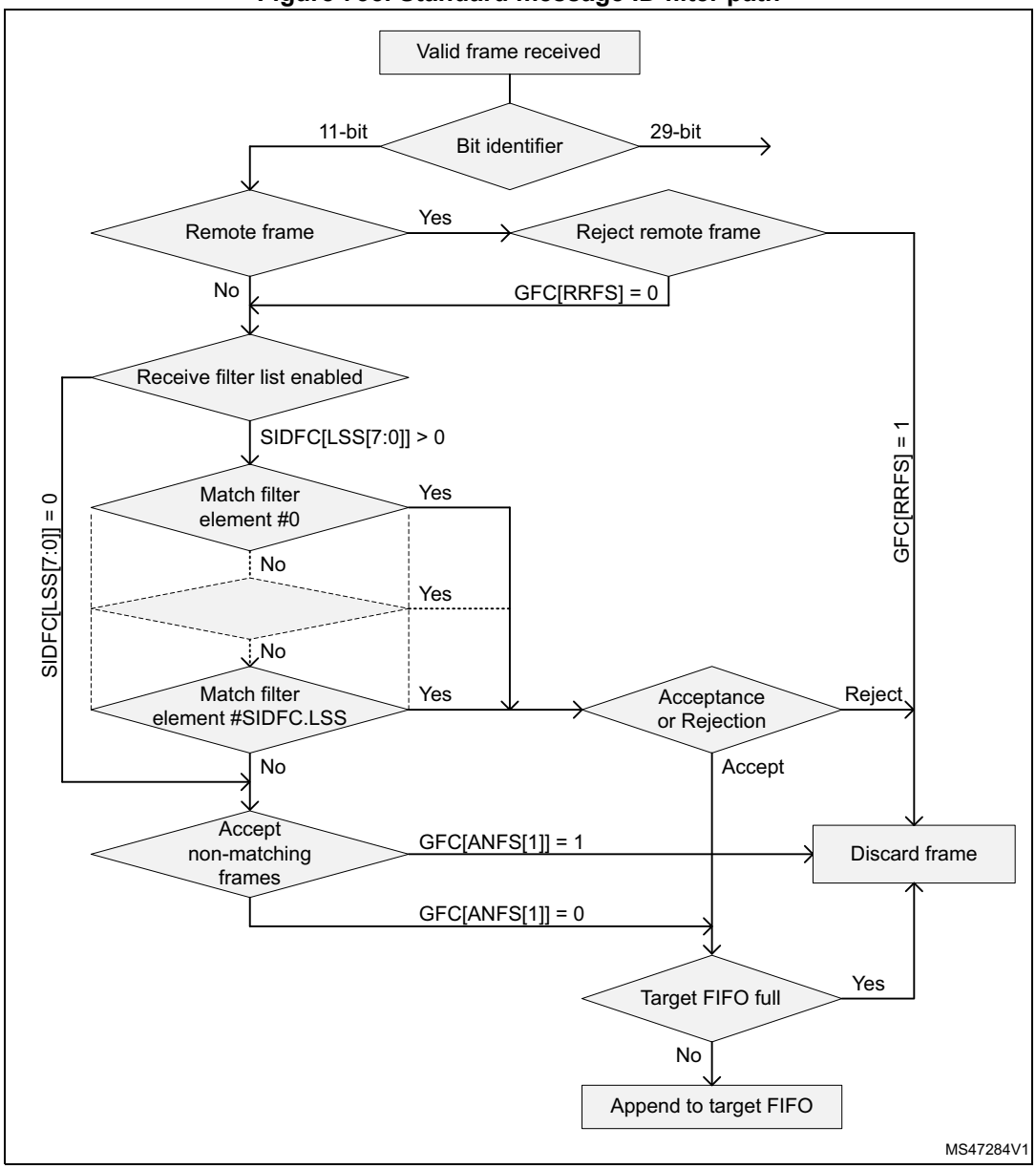
A 0 bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 735 shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 55.3.20 on page 2487*.

Figure 735. Standard message ID filter path

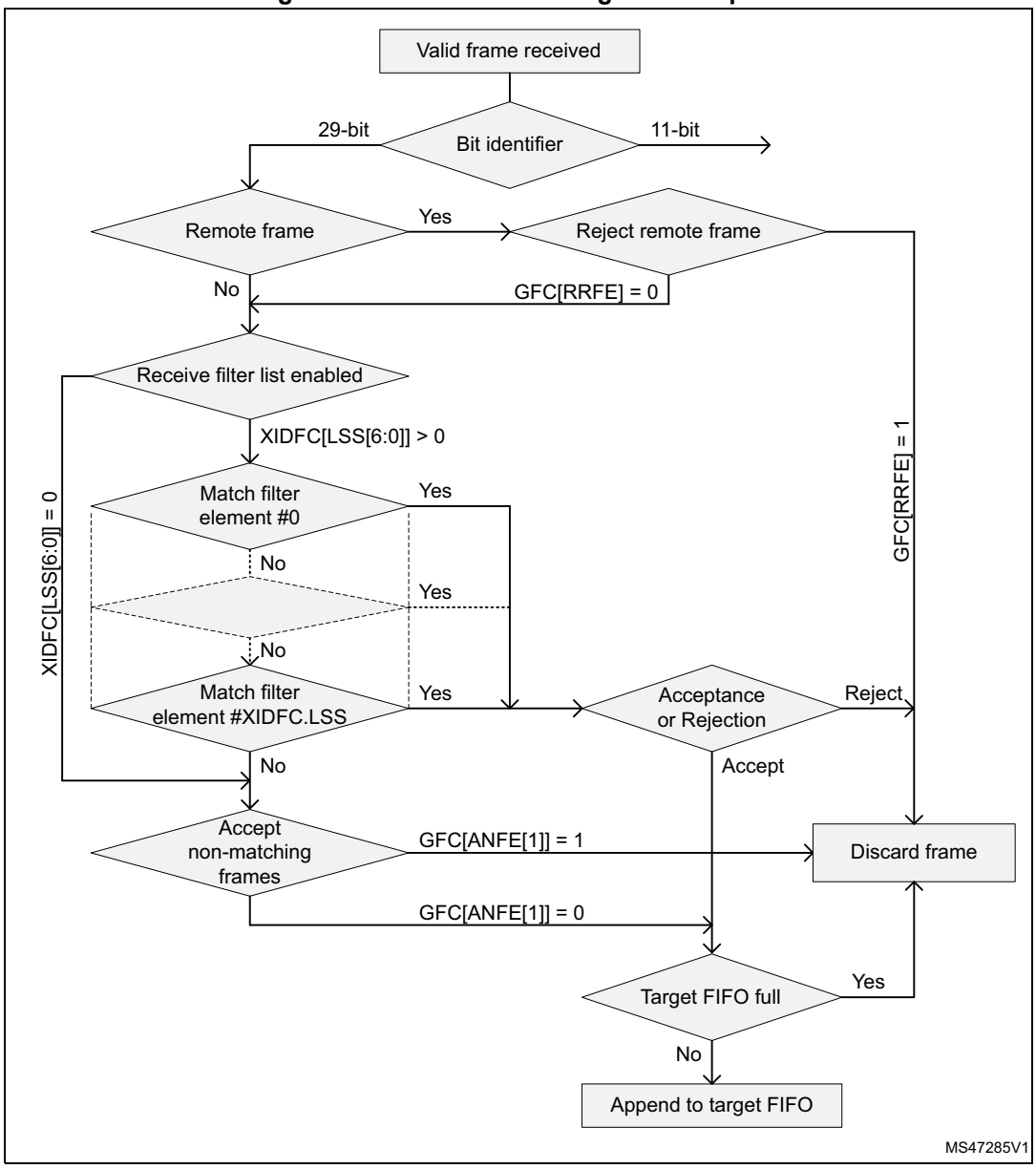


Controlled by the global filter configuration (FDCAN_GFC) and the standard ID filter configuration (FDCAN_SIDFC) message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Extended message ID filtering

Figure 736 shows the flow for extended message ID (29-bit Identifier) filtering. The extended message ID filter element is described in *Section 55.3.21 on page 2489*.

Figure 736. Extended message ID filter path



Controlled by the global filter configuration FDCAN_GFC and the extended ID filter configuration FDCAN_XIDFC message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The extended ID AND Mask (FDCAN_XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers FDCAN_RXF0C and FDCAN_RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx buffer and FIFO element is described in *Section 55.3.17: FDCAN Rx buffer and FIFO element*.

When an Rx FIFO full condition is signaled by FDCAN_IR.RFnF, no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag FDCAN_IR.RFnL is set.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by FDCAN_RXFnC.FnWM, interrupt flag FDCAN_IR.RFnW is set.

When reading from an Rx FIFO, Rx FIFO get index RXFnS[FnGI] + FIFO element size has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition interrupt flag FDCAN_IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled by RXFnS.RFnL = 1. In addition interrupt flag FDCAN_IR.RFnL is set.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is that it can happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 738* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in elements 1 and 2 are lost.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

Dedicated Rx buffers

The FDCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via FDCAN_RXBC.RBSA.

For each Rx buffer a standard or extended message ID filter element with SFEC / EFEC=111 and SFID2 / EFID2[10:9] = 00 has to be configured (see *Section 55.3.20: FDCAN standard message ID filter element* and *Section 55.3.21: FDCAN extended message ID filter element*).

After a received message has been accepted by a filter element, the message is stored into the Rx buffer in the message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag FDCAN_IR.DRX (message stored in dedicated Rx buffer) in the interrupt register is set.

Table 462. Example of filter configuration for Rx buffers

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective New data flag in register NDAT1,2 is set. As long as the New data flag is set, the respective Rx buffer is locked against updates from received matching frames. The New data flags have to be reset by the Host by writing a 1 to the respective bit position.

While an Rx buffer New data flag is set, a message ID filter element referencing this specific Rx buffer will not match, causing the acceptance filtering to continue. The following message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Rx buffer handling

- Reset interrupt flag FDCAN_IR.DRX
- Read New data registers
- Read messages from message RAM
- Reset New data flags of processed messages

Filtering for Debug messages

Filtering for debug messages is done by configuring one standard/extended message ID filter element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC/EFEC has to be programmed to 111. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New data flag nor FDCAN_IR.DRX are set. The reception of debug messages can be monitored via FDCAN_RXF1S.DMS.

Table 463. Example of filter configuration for Debug messages

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO.Up to 32 Tx buffers can be set up for message transmission (see *Dedicated Tx buffers*). Depending on the configuration of the element size (FDCAN_RXESC), between two and sixteen 32-bit words (Rn = 3 ... 17) are used for storage of a CAN message data field.

Table 464. Possible configurations for frame transmission

FDCAN_CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bitrate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bitrate switching
1	1	1	1	FD with bitrate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register FDCAN_TXBRP is updated, or when a transmission has been started.

Transmit pause

This feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, as an example, CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having

received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by TXP bit in FDCAN_CCCR register. If the bit is set, the FDCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is disabled (FDCAN_CCCR.TXP = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmission under complete control of the CPU. Each dedicated Tx buffer is configured with a specific message ID. In case that multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an add request via FDCAN_TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates four 32-bit words in the message RAM. Therefore the start address of a dedicated Tx buffer in the message RAM is calculated by adding four times the transmit buffer index (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

Table 465. Tx buffer/FIFO - queue element size

FDCAN_TXESC.TBDS[2;0]	Data field (bytes)	Element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Tx FIFO

Tx FIFO operation is configured by programming FDCAN_TXBC.TFQM to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index FDCAN_TXFQS.TFGI. After each transmission the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level FDCAN_TXFQS.TFFL as difference between get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO Full (FDCAN_TXFQS.TFQF = 1) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via FDCAN_TXBAR. The put index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO free level remain unchanged.

A Tx FIFO element allocates four 32-bit words in the message RAM. Therefore the start address of the next available (free) Tx FIFO buffer is calculated by adding four times the put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

Tx queue

Tx queue operation is configured by programming FDCAN_TXBC.TFQM to 1. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case that multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request cyclically increments the put index to the next free Tx buffer. In case that the Tx queue is full (FDCAN_TXFQS.TFQF = 1), the put index is not valid and no further message should be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

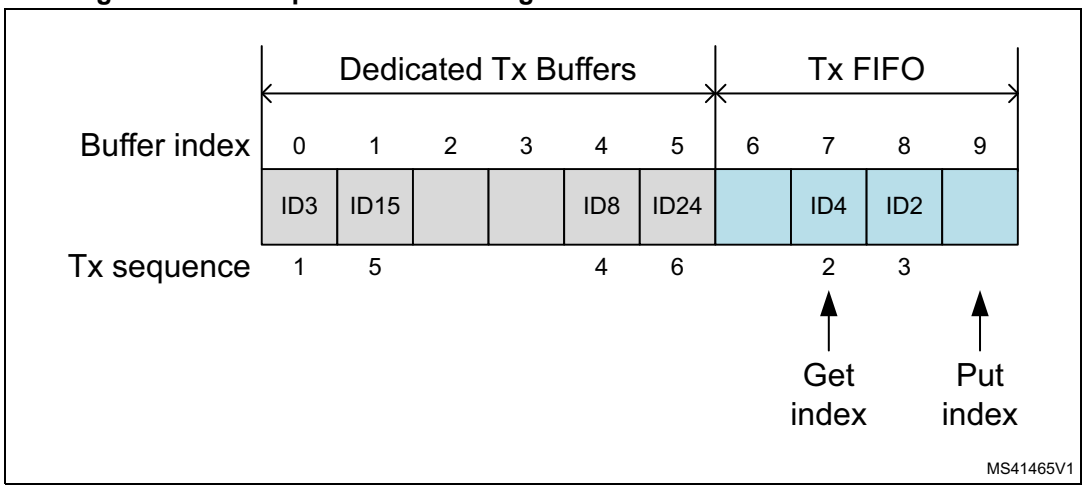
The application may use register FDCAN_TXBRP instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates four 32-bit words in the message RAM. Therefore the start address of the next available (free) Tx queue buffer is calculated by adding four times the Tx queue put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

Mixed dedicated Tx buffers / Tx FIFO

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx buffers assigned to the Tx FIFO is configured by FDCAN_TXBC.TFQS. In case, FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 737. Example of mixed configuration dedicated Tx buffers / Tx FIFO



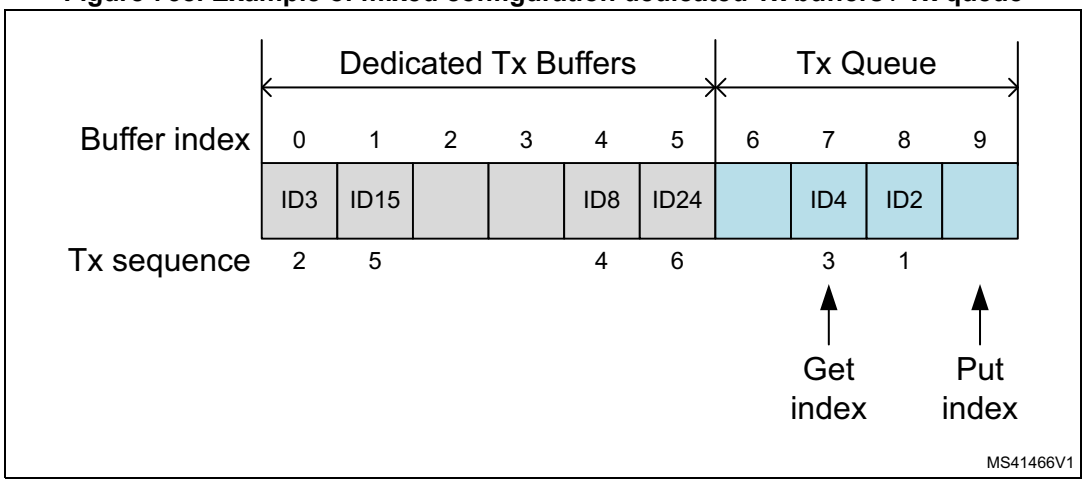
Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by FDCAN_TXFS.TFGI)
- Buffer with lowest message ID gets highest priority and is transmitted next

Mixed dedicated Tx buffers / Tx queue

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx queue buffers is configured by FDCAN_TXBC.TFQS. If FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 738. Example of mixed configuration dedicated Tx buffers / Tx queue



Tx priority setting:

- Scan all Tx buffers with activated transmission request
- Tx buffer with lowest message ID gets highest priority and is transmitted next

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer the Host has to write a 1 to the corresponding bit position (= number of Tx buffer) of register FDCAN_TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register FDCAN_TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN_TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before this transmission could have been started, there follows a short time window where no transmission is stared even if another message is pending in this node. This may enable another node to transmit a message that may have a priority lower than that of the second message in this node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in *Tx FIFO*. Depending on the configuration of the element size (FDCAN_TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message data field.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by FDCAN_IR.TEFF, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and interrupt flag FDCAN_IR.TEFL is set.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by FDCAN_TXEFC.EFWM, interrupt flag FDCAN_IR.TEFW is set.

When reading from the Tx event FIFO, two times the Tx event FIFO get index FDCAN_TXEFS.EFGI has to be added to the Tx event FIFO start address FDCAN_TXEFC.EFSA.

55.3.3 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index, see *FDCAN Rx FIFO 0 acknowledge register*

(FDCAN_RXF0A), *FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)*, and *FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)*. Writing to the FIFO acknowledge index will set the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

1. When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
2. When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: index of the last element read), to update the FIFO get index.

Due to the fact that the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the FIFO acknowledge index should not be written because this would set the get index to a wrong position and also alters the FIFO fill level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

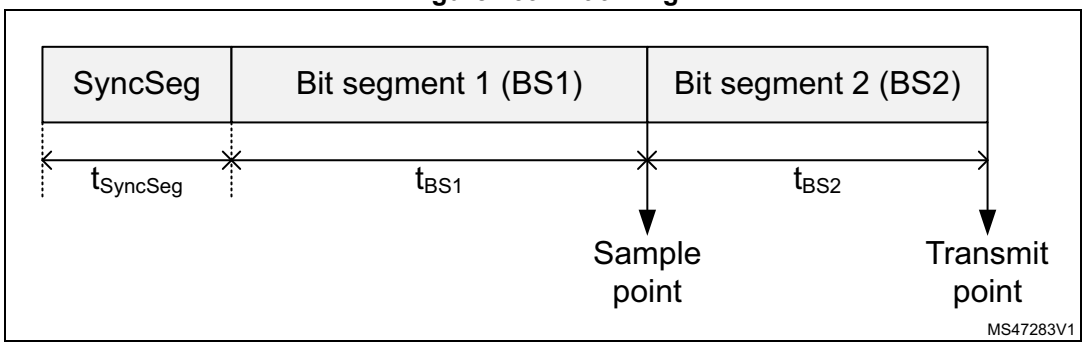
55.3.4 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 739*, its operation may be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, that has a fixed length of one time quantum (1 x tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.

Figure 739. Bit timing



The baudrate is the inverse of bit time (baudrate = 1 / bit time), which, in turn, is the sum of three components. *Figure 739* indicates that bit time = $t_{SyncSeg}$ + t_{BS1} + t_{BS2}, where:

- for the nominal bit time
  - tq = (FDCAN_NBTP.NBRP[8:0] + 1) * $t_{fdcan_tq_clk}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_NBTP.NTSEG1[7:0] + 1)
  - t_{BS2} = tq * (FDCAN_NBTP.NTSEG2[6:0] + 1)
- for the data bit time
  - tq = (FDCAN_DBTP.DBRP[4:0] + 1) * $t_{fdcan_tq_clk}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_DBTP.DTSEG1[4:0] + 1)
  - t_{BS2} = tq * (FDCAN_DBTP.DTSEG2[3:0] + 1)

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. Registers FDCAN_DBTP and FDCAN_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set.

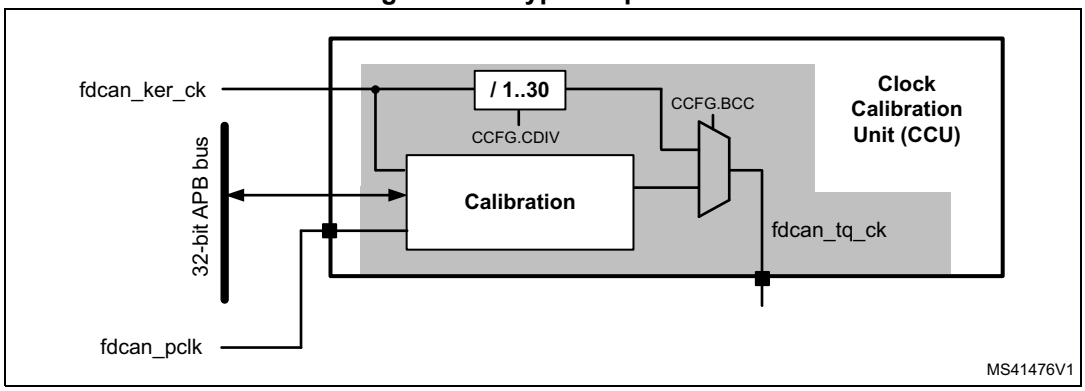
Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

55.3.5 Clock calibration on CAN

After device reset the clock calibration unit (CCU) does not provide a valid clock signal to the FDCAN1 and FDCAN2. The CCU has to be initialized via FDCAN_CCFG register. The FDCAN_CCFG register can be written only when FDCAN1 has both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set. In consequence the CCU and the FDCAN1 initialization needs to be completed before any FDCAN1 and/or FDCAN2 module can operate.

Clock calibration is bypassed when FDCAN_CCFG.BCC = 1 (see *Figure 740*).

Figure 740. Bypass operation



Operating conditions

The clock calibration on CAN unit is designed to operate under the following conditions:

- a CAN kernel clock frequency fdcan_ker_ck up to 80 MHz
- FDCAN bitrates:
  - Nominal bitrate: up to 1 Mbit/s
  - Data bitrate: between nominal bitrate and 8 Mbit/s

The clock calibration on FDCAN unit generates a calibrated time quanta clock fdcan_tq_ck in the range from 0.5 to 25 MHz.

Note: The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock (fdcan_tq_ck < fdcan_pclk). This has to be considered when the clock calibration on CAN unit is bypassed (FDCAN_CCFG.BCC = 1).

Calibration accuracy

The calibration accuracy in state Precision_Calibrated depends upon the factors listed below.

- Dynamic clock tolerance at the CAN kernel clock input fdcan_ker_ck
- Measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one fdcan_pclk period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of FDCAN_CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages has to be chosen to fit the clock tolerance requirements of the FDCAN1 module.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages e.g. caused by change of temperature or operating voltage.

Functional description

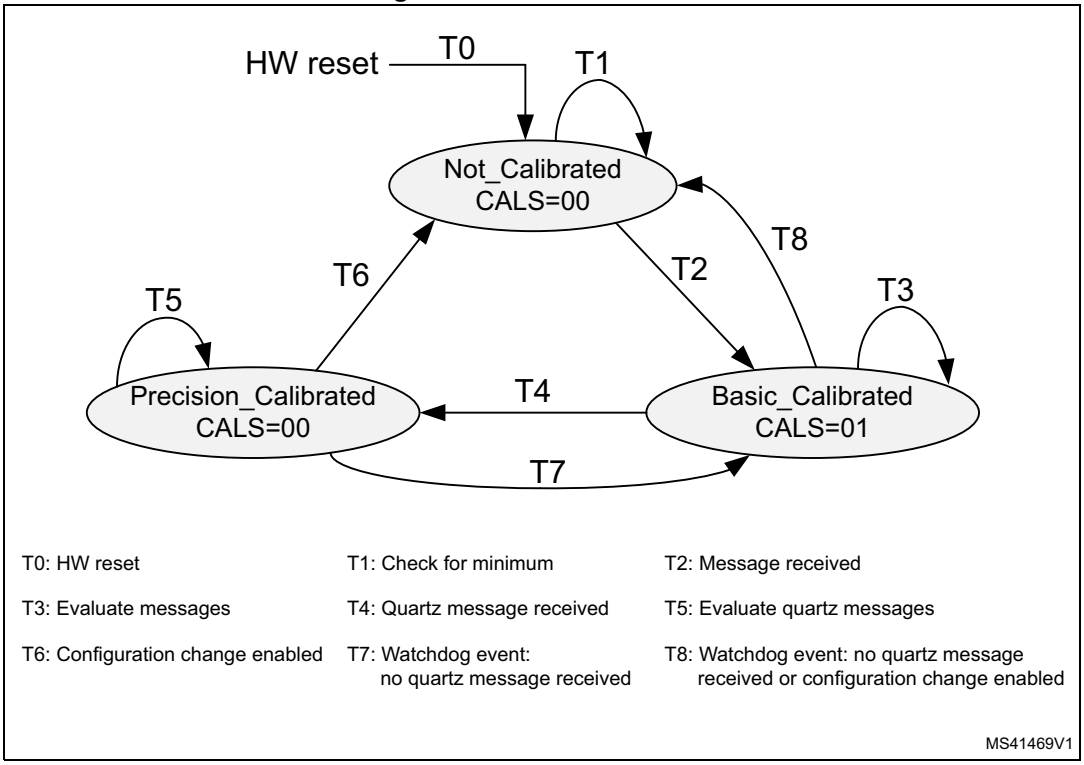
Calibration of the time quanta clock fdcan_tq_ck via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock fdcan_ker_ck.

1. First step: basic calibration
   The minimum distance between two consecutive falling edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider (FDCAN_CCFG.CDIV) is updated each time a new

measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol controller detects a valid CAN message.

2. Second step: Precision calibration
The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of fdcan_ker_ck periods. The length of this bit sequence can be configured to 32 or 64 bits via FDCAN_CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/6 byte data field is required. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 741. FSM calibration



A change in the calibration state sets interupt flag FDCAN_CCU_IR.CSC, if enabled by the interupt enable FDCAN_CCU_IE.CSCE. it remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

Until precision calibration is achieved, FDCAN1 and FDCAN2 operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the fdcan_ker_clk is done by software by evaluating the calibration status from register CCU_CSTAT, FDCAN1 and FDCAN2 have to be set to restricted operation mode (FDCAN_CCCR.ASM = 1) until the calibration on CAN unit is in state Precision_Calibrated (see *Application*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. calibration frames are detected by the FDCAN1 acceptance filtering A filter element and a Rx buffer have to be configured in the FDCAN1 to identify and store calibration messages. After reception of a calibration message the Rx buffer new data flag has to be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node has to transmit its first message after startup with at least one 1010 binary sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the clock calibration on CAN unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the FDCAN1 complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

Configuration

The clock calibration on CAN unit is configured via register FDCAN_CCFG, i.e. when FDCAN1 has FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at pin FDCAN1_RX is measured. The number of clock periods depends on the clock frequency applied at input fdcan_ker_ck. In case the measured number of clock periods is below the minimum configured by FDCAN_CCFG.OCPM (as an example, because of a glitch on FDCAN1_RX) the value is discarded and measurement continues.

It is recommended to configure FDCAN_CCFG.OCPM slightly below two CAN bit times:

FDCAN_CCFG.OCPM < ((2 x CAN bit time) / fdcan_ker_ck period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via FDCAN_CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages.

The number of time quanta per bit time configured by FDCAN_CCFG.TQBT is used together with the measured number of oscillator clock periods FDCAN_CCU_CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring FDCAN_CCFG.BCC = 1, the internal clock divider has to be configured via FDCAN_CCFG.CDIV to fulfill the condition fdcan_tq_ck < fdcan_pclk.

Note: When clock calibration on CAN is active (FDCAN_CCFG.BCC = 0), the baudrate prescalers of FDCAN1 and FDCAN2 have to be configured to inactive.

Status signaling

The status of the clock calibration on CAN unit can be monitored by reading register FDCAN_CCU_CSTAT. When in state Precision_Calibrated the oscillator clock period counter FDCAN_CCU_CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while FDCAN_CCU_CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by FDCAN_CCU_CSTAT.CALS. A change in the calibration state sets interupt flag FDCAN_CCU_IR.CSC, if enabled by the interrupt enable FDCAN_CCU_IE.CSCE. it remains set until cleared by writing 1 in CCU_IR.CSC.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE, interrupt line cu_int is activated (set to high). Interrupt line cu_int remains active until interrupt flag FDCAN_CCU_IR.CWE is reset

Application

Clock calibration bypassed

The CCU internal clock divider is configured for division by one (FDCAN_CCFG.CDIV = 0x0000). In this operation mode the input clock fdcan_ker_ck is directly routed to the clock output fdcan_tq_ck. In this case fdcan_tq_ck is independent from the configuration and status of FDCAN1and FDCAN2 connected to the CCU. With a fdcan_ker_ck of 20 / 40 / 80 MHz CAN FD operation is possible.

Software calibration

The clock calibration on CAN unit also supports software calibration of fdcan_ker_ck by trimming of an on-chip oscillator. For calculation of the trimming values the Host has to read the CCU state from CCU_CSTAT. In this operation mode the CCU output signal cu_cok is fixed to 1, the clock from fdcan_ker_ck is routed to output fdcan_tq_ck (FDCAN_CCFG.BCC = 1).

The input clock fdcan_ker_ck must be in the range between 80MHz and 100 MHz. The clock divider of CCU has to be configured via FDCAN_CCFG.CDIV to bring fdcan_tq_ck to a valid range. All other configuration parameters have to be set via FDCAN_CCFG. For correct operation of tFDCAN1and FDCAN2, the APB clock fdcan_pclk needs to be equal or higher than the time quanta clock (fdcan_tq_ck). CAN FD operation is not possible.

For startup FDCAN1 and FDCAN2 have to be both configured for restricted operation (FDCAN_CCCR.ASM = 1) before FDCAN_CCCR.INIT is reset. The input clock fdcan_ker_ck has to be adjusted until the clock calibration on CAN unit has reached state Precision_Calibrated. Now the software can reset FDCAN_CCCR.ASM and the CANFD1 and CANFD2 can start normal operation.

During operation the software has to check regularly whether the clock calibration on CAN unit is still in state Precision_Calibrated. In case the clock calibration on CAN unit has left state Precision_Calibrated due to drift of fdcan_ker_ck, FDCAN1 and FDCAN2 have to be set into restricted operation mode by programming FDCAN_CCCR.INIT, FDCAN_CCCR.CCE, and FDCAN_CCCR.ASM to 1. After fdcan_ker_ck has been adjusted successfully (clock calibration on CAN unit is in state Precision_Calibrated), FDCAN1 and FDCAN2 can resume normal operation.

Note: Trimming accuracy needs to be to sufficient to meet the CAN clock tolerance requirements for the configured bitrate.

Clock calibration active

This operation mode is entered by resetting FDCAN_CCFG.BCC to 0. In this operation mode the CCU output signals cu_cok and fdcan_ker_ck are controlled by the CCU. When the CCU is not in state Precision_Calibrated cu_cok is 0.

Generation of CCU output signals fdcan_tq_ck and cu_cok depends on the state of the FDCAN1. Input clock fdcan_ker_ck must be in the range between 80 and 100 MHz. Configuration of the CCU and FDCAN1 is required. CAN FD operation is not possible.

In case FDCAN1 turns to Bus_Off or when its INIT bit is set by Host command (FDCAN_CCCR.INIT = 1), the CCU enters state Not_Calibrated and output cu_cok is reset to 0. CANFD1 and CANFD2 enter restricted operation mode.

Note: This is the default operation mode after reset in case the reset value of FDCAN_CCFG.BCC is configured to 0 at synthesis via generic parameter.

55.3.6 TTCAN operations (FDCAN1 only)

Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the time master (sender of the reference message).

For level 1 the data length must be at least one; for level 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register FDCAN_TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

Level 1

Level 1 operation is configured via FDCAN_TTOCF.OM = 01 and FDCAN_TTOCF.GEN. external clock synchronization is not available in level 1. The information related to the reference message is stored in the first data byte as shown in *Table 466*. Cycle_Count is optional.

Table 466. First byte of level 1 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5;0]					

Level 2

Level 2 operation is configured via FDCAN_TTOCF.OM = 10 and FDCAN_TTOCF.GEN. The information related to the reference message is stored in the first four data bytes as shown in *Table 467*. Cycle_Count and the lower four bits of FDCAN_NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 467. First four bytes of level 2 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5;0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

Level 0

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. External event-synchronized time-triggered operation is not available in level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In level 0 Next_is_Gap is always 0. Cycle_Count and the lower four bits of NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 468. First four bytes of level 0 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5;0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

55.3.7 TTCAN configuration

TTCAN timing

The network time unit (NTU) is the unit in which all times are measured. The NTU is a constant of the whole network and is defined as a priority by the network system designer. In TTCAN level 1 the NTU is the nominal CAN bit time. In TTCAN level 0 and level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN level 0 and level 2 the length of the NTU is defined by the time unit Ratio TUR. The TUR is in general a non-integer number, given by
TUR = FDCAN_TURNA.NAV / FDCAN_TURCF.DC. The NTU length is given by
NTU = CAN clock period x TUR.

The TUR numerator configuration NC is an 18-bit number, FDCAN_TURCF[NCL[15:0]] can be programmed in the range 0x0000–0xFFFF. FDCAN_TURCF[NCH[17:16]] is hard wired to 0b01. When 0xnnnn is written to FDCAN_TURCF[NCL[15:0]], FDCAN_TURNA.NAV starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR denominator configuration FDCAN_TURCF.DC is a 14-bit number. FDCAN_TURCF.DC may be programmed in the range 0x0001 - 0x3FFF (0x0000 is an illegal value).

In level 1, NC must be equal or higher than 4 x FDCAN_TURCF.DC. In level 0 and level 2 NC must be equal or higher than 8 x FDCAN_TURCF.DC to get the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets FDCAN_TURCF.DC to 0x1000 and FDCAN_TURCF.NCL to 0x10000, resulting in an NTU consisting of sixteen CAN clock periods. Local time and application watchdog are not started before either the FDCAN_CCCR.INIT is reset, or FDCAN_TURCF.ELT is set. FDCAN_TURCF.ELT may not be set before the NTU is configured. Setting FDCAN_TURCF.ELT to 1 also locks the write access to register FDCAN_TURCF.

At startup FDCAN_TURNA.NAV is updated from NC (= FDCAN_TURCF.NCL + 0x10000) when FDCAN_TURCF.ELT is set. In TTCAN level 1 there is no drift compensation. FDCAN_TURNA.NAV does not change during operation, it is always equal to NC.

In TTCAN level 0 and level 2 there are two possibilities for FDCAN_TURNA.NAV to change. When operating as time slave or backup time master, and when FDCAN_TTOCF.ECC is set, FDCAN_TURNA.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization, it returns to NC. When operating as the actual time master, and when FDCAN_TTOCF.EECS is set, the Host may update FDCAN_TURCF.NCL. When the Host sets FDCAN_TTOCN.ECS, FDCAN_TURNA.NAV will be updated from the new value of NC at the next reference message. The status flag FDCAN_TTOST.WECS as is set when FDCAN_TTOCN.ECS is set and is cleared when FDCAN_TURNA.NAV is updated. FDCAN_TURCF.NCL is write locked while FDCAN_TTOST.WECS is set.

In TTCAN level 0 and level 2 the clock calibration process adapts FDCAN_TURNA.NAV in the range of the synchronization deviation limit SDL of NC ± 2(FDCAN_TTOCF.LDSDL + 5). FDCAN_TURCF.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of FDCAN_TURNA.NAV.

The synchronization deviation SD is the difference between NC and FDCAN_TURNA.NAV (SD = | NC - FDCAN_TURNA.NAV |). It is limited by the synchronization deviation limit SDL, which is configured by its dual logarithm FDCAN_TTOCF.LDSDL (SDL = 2 (FDCAN_TTOCF.LDSDL + 5)) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new basic cycle. When the calculated TURNA[NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and FDCAN_TTIR.GTE is set and FDCAN_TTOSC.QCS is reset, or in case of the Disc_Bit = 1, FDCAN_TTIR.GTD is set.

Table 469. TUR configuration example

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
FDCAN _TURCF.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

FDCAN_TTOCN.ECS schedules NC for activation by the next reference message. FDCAN_TTOCN.SGT schedules FDCAN_TTGTP.TP for activation by the next reference message. Setting FDCAN_TTOCN.ECS and FDCAN_TTOCN.SGT requires FDCAN_TTOCF.EECS to be set (external clock synchronization enabled) while the FDCAN is actual time master.

The TTCAN module provides an application watchdog to verity the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The application watchdog limit FDCAN_TTOCF.AWL specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The application watchdog is served by reading register FDCAN_TTOST. FDCAN_TTOST.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag FDCAN_TTIR.AW is set. For software development, the application watchdog may be disabled by programming FDCAN_TTOCF.AWL to 0x00, see *Section 55.4.49: FDCAN TT operation configuration register (FDCAN_TTOCF)*.

Timing of interface signals

The timing events which cause a pulse at output FDCAN trigger time mark interrupt pulse m_ttcan_tmp and FDCAN register time mark interrupt pulsem_ttcan_rtp are generated in the CAN clock domain. There is a clock domain crossing delay to be considered before the same event is visible in the APB clock domain (when FDCAN_TTIR.TTMI is set or FDCAN_TTIR.RTMI is set). As an example, the signals can be connected to the timing input(s) of another FDCAN node (fdcan_swt/fdcan_evt), in order to automatically synchronize two TTCAN networks.

Output FDCAN start of cyclem_ttcan_soc gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the APB clock domain.

55.3.8 Message scheduling

FDCAN_TTOCF.TM controls whether the TTCAN operates as potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message identifier FDCAN_TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest. There cannot be two nodes in the network using the same master priority. FDCAN_TTRMC.RID is used for recognition of reference messages. FDCAN_TTRMC.RMPS is not relevant for time slaves.

The initial reference trigger offset FDCAN_TTOCF.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for FDCAN_TTOCF.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

FDCAN_TTOCF.OM decides whether the node operates in TTCAN level 0, level 1, or level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on level 1 in a level 2 network, but not vice versa. The configuration of the TTCAN operation mode via FDCAN_TTOCF.OM is the last step in the setup. With FDCAN_TTOCF.OM = 00 (event-driven CAN communication), the FDCAN operates according to ISO 11898-1: 2015, without time triggers.With FDCAN_TTOCF.OM = 01 (level

1), the FDCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With FDCAN_TTOCF.OM = 10 (level 2), the TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With FDCAN_TTOCF.OM = 11 (level 0), the FDCAN operates as event-driven CAN but maintains a calibrated global time base as in level 2.

FDCAN_TTOCF.EECS enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in levels 0 and level 2 only) the global clock phase to an external reference.

FDCAN_TTMLM.ENTT in the TT matrix limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger repeat factors have to be taken into account.

An inaccurate configuration of FDCAN_TTMLM.ENTT will result either in a TxCount Underflow (FDCAN_TTIR.TXU = 1 and FDCAN_TTOST.EL = 01, severity 1), or in a Tx count Overflow (FDCAN_TTIR.TXO = 1 and FDCAN_TTOST.EL = 10, severity 2).

Note: In case the first reference message seen by a node does not have Cycle_Count 0, this node may finish its first matrix cycle with its Tx count resulting in a Tx count underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.

FDCAN_TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of eight basic cycles FDCAN_TTMLM.CCM would be 7. FDCAN_TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

FDCAN_TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If a transmission of a message cannot be started inside the Tx enable window because of for example, a slight overlap from the previous time window message, the transmission cannot be started in that time window at all. FDCAN_TTMLM.TXEW has to be chosen with respect to the network synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

Trigger memory

The trigger memory is part of the external message RAM to which the TTCAN is connected to (see *Section 55.4.26: FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C)*). It stores up to 64 trigger elements. A trigger memory element consists of time mark TM, cycle code CC, trigger type TYPE, filter type FTYPE, message Number MNR, message status count MSC, time mark event internal TMIN, time mark event external TMEX (see *Section 55.3.22: FDCAN trigger memory element*).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a

transmit trigger, the Tx handler starts to read the message from the message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx handler has to be able to prepare the transmission before the transmit trigger time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mar k is reached. If the reference message is n NTU long, a trigger with a time mark lower than n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger time mark. The FDCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialization) be seen at a value lower than n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger time mark + 1 NTU + 1 CAN bit time.

The trigger list will be different for all nodes in the FDCAN network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

Trigger types

Tx_Ref_Trigger (TYPE = 0000) and Tx_Ref_Trigger_Gap (TYPE = 0001) cause the transmission of a reference message by a time master. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized
time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Tx_Trigger_Single (TYPE = 0010), Tx_Trigger_Continuous (TYPE = 0011), Tx_Trigger_Arbitration (TYPE = 0100), and Tx_Trigger_Merged (TYPE = 0101) cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit is reset.

Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (FDCAN_TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = 0110) and Watch_Trigger_Gap (TYPE = 0111) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Rx_Trigger (TYPE = 1000) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger time mark is reached.

Time_Base_Trigger (TYPE = 1001) are used to generate internal/external events depending on the configuration of TMIN and TMEX.

End_of_List (TYPE = 1010 … 1111) is an illegal trigger type, a configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger and Watch_Trigger_Gap.

Restrictions for the node trigger list

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) must be set to trigger type End_of_List.

A typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of

a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the reference trigger offset FDCAN_TTOCF.IRTO. In case error level 2 is reached (FDCAN_TTOST.EL = 10), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- APB clock frequency
- Speed and waiting time for trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

Example for trigger handling

The example shows how the trigger list is derived from a node system matrix. Assumption is that node A is first time master and has knowledge of the section of the system matrix shown in *Table 470*.

Table 470. System matrix, Node A

Cycle count	Time mark						
	1	2	3	4	5	6	7
0	Tx7	-	-	-	-	TxRef	Error
1	Rx3	-	Tx2, Tx4		-	TxRef	Error
2	-	-	-	-	-	TxRef	Error
3	Tx7	-	Rx5	-	-	TxRef	Error
4	Tx7	-	-	Rx6	-	TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the corresponding number of basic cycles in the system matrix is, respectively, 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by FDCAN_TTMLM.CCM. The cycle code CC is composed of repeat factor (= value of most significant 1) and the number of the first basic cycle in the system matrix (= bit field after most significant 1).

As an example, with a cycle code of 0b0010011 (repeat factor = 16, first basic cycle = 3) and a maximum cycle count of FDCAN_TTMLM.CCM = 0x3F matches occur at cycle counts 3, 19, 35 and 51.

A trigger element consists of time mark TM, cycle code CC, trigger type TYPE, and message Number MNR. For transmission MNR references the Tx buffer number (0 ... 31). For reception MNR references the number of the filter element (0 ... 127) that matched during acceptance filtering. Depending on the configuration of the filter type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for generation of time mark event internal TMIN, and time mark event external TMEX. The message status count MSC holds the counter value (0 ... 7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 471. Trigger list, Node A

Trigger	Time mark TM[15:0]	Cycle code CC [6:0]	Trigger type TYPE [3:0]	Mess No. MNR [6:0]
0	Mark1	0b0000100	Tx_Trigger_Single	7
1	Mark 1	0b1000000	Rx_Trigger	3
2	Mark 1	0b1000011	Tx_Trigger_Single	7
3	Mark 3	0b1000001	Tx_Trigger_Merged	2
4	Mark 3	0b1000011	Rx_Trigger	5
5	Mark 4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark 4	0b1000100	Rx_Trigger	6
7	Mark 6	N/A	Tx_Ref_Trigger	0 (Ref)
8	Mark 7	N/A	Watch_Trigger	N/A
9	N/A	N/A	End_of_List	N/A

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from 0 until a trigger with time mark higher than cycle time and with its cycle code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the time mark TM, the action defined by trigger type TYPE and message Number MNR is started. There is an error in the configuration when End_of_List is reached.

At mark 6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the watch trigger at mark 7 is reached, the node was not able to transmit the reference message; error treatment is started.

Detection of configuration errors

A configuration error is signaled via FDCAN_TTOST.EL = 11 (severity 3) when:

- The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.
- The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a

Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

- The FSE of a node with FDCAN_TTOCF.TM=0 (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.
- Any time mark placed inside the Tx enable window (defined by FDCAN_TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.
- A time mark is placed near the time mark of a Tx_Ref_Trigger and the reference trigger offset FDCAN_TTOST.RTO causes a reversal of their sequential order measured in cycle time.

TTCAN schedule initialization

The synchronization to the TTCAN message schedule starts when FDCAN_CCCR.INIT is reset. The TTCAN can operate strictly time-triggered (FDCAN_TTOCF.GEN = 0) or external event-synchronized time-triggered (FDCAN_TTOCF.GEN = 1). All nodes start with cycle time 0 at the beginning of their trigger list with FDCAN_TTOST.SYS = 00 (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

Time slaves

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

Note: Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter that counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a fatal error (FDCAN_TTOST.EL = 11, severity 3), set interrupt flag FDCAN_TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (FDCAN_CCCR.MON set to 1). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

Note: To leave the fatal error state, the Host has to set FDCAN_CCCR.INIT = 1. After reset of FDCAN_CCCR.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronization, the first reference message sets FDCAN_TTOST.SYS = 01 (Synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

Potential time masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches

Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting FDCAN_TTIR.IWT will re-enable the transmission of reference messages until next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE will not be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it will assume a fatal error (FDCAN_TTOST.EL = 11, severity 3), set interrupt flag FDCAN_TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (FDCAN_CCCR.MON set to 1). In bus monitoring mode, it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets FDCAN_TTOST.SYS = 01 (synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (FDCAN_TTOST.MS = 11) when it was the transmitter of the last reference message, else it is backup time master (FDCAN_TTOST.MS = 10).

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

55.3.9 TTCAN gap control

All functions related to Gap control apply only when the FDCAN is operated in external event synchronized time-triggered mode (FDCAN_TTOCF.GEN = 1). In this operation mode the FDCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event- synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = 1 e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes FDCAN_TTOCN.NIG = 1. The Next_is_Gap bit will be transmitted as 1 with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input pin fdcan_evt by writing FDCAN_TTOCN.GCS = 1. When a reference message is started and FDCAN_TTOCN.GCS is set, a HIGH level at event trigger pin fdcan_evt will set Next_is_Gap = 1.

As soon as that reference message is completed, the FDCAN_TTOST.WFE bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, FDCAN_TTOST.GSI will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit FDCAN_TTOST.GSI will remain at 0.

When a potential time master is in synchronization state In_Gap (FDCAN_TTOST.SYS = 10), it has four options to intentionally finish a Gap:

1. Under software control by writing FDCAN_TTOCN.FGP = 1.
2. Under hardware control (FDCAN_TTOCN.GCS = 1) an edge from HIGH to LOW at the event-trigger input pin fdcan_evt sets FDCAN_TTOCN.FGP and restarts the schedule.
3. The third option is a time-triggered restart. When FDCAN_TTOCN.TMG = 1, the next register time mark interrupt (FDCAN_TTIR.RTMI = 1) will set FDCAN_TTOCN.FGP and start the reference message.
4. Finally any potential time master will finish a Gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

None of these options can cause a basic cycle to be interrupted with a reference message.

Setting of FDCAN_TTOCN.FGP after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When FDCAN_TTOCN.FGP is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = 1 in the reference message will be ignored, as well as the event-trigger input pin fdcan_evt and the bits FDCAN_TTOCN.NIG, FDCAN_TTOCN.FGP, and FDCAN_TTOCN.TMG.

55.3.10 Stop watch

The stop watch function enables capturing of FDCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via FDCAN_TTOCN.SWS. When FDCAN_TTOCN.SWS is different from 00 and TT interrupt register flag FDCAN_TTIR.SWE is 0, the actual value of the time selected by FDCAN_TTTOCN.SWS will be copied into FDCAN_TTCPT.SWV on the next rising / falling edge (as configured via FDCAN_TTOCN.SWP) on stop watch trigger pin fdcan_swt. This will set interrupt flag FDCAN_TTIR.SWE. After the application program has read FDCAN_TTCPT.SWV, it may enable the next stop watch event by resetting FDCAN_TTIR.SWE to 0.

55.3.11 Local time, cycle time, global time, and external clock synchronization

There are two possible levels in time-triggered CAN:

1. Level 1 only provides time-triggered operation using cycle time.
2. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time.

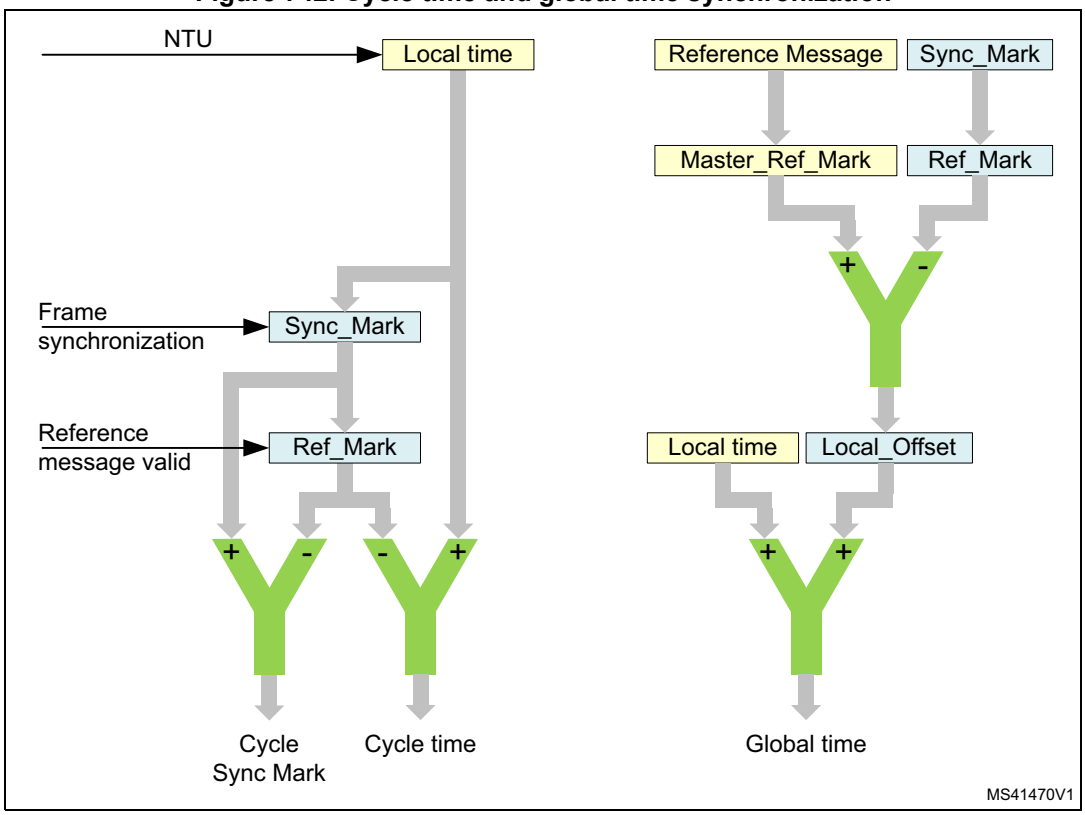
The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented eight times each NTU. If the length of the NTU is shorter than eight CAN clock periods (as may be configured in level 1, or as a result of clock calibration in level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only four times each NTU.

Figure 742 describes the synchronization of the cycle time and global time, performed in the same manner by all FDCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message is frame

synchronization event. This frame synchronization event occurs at the sample point of each start of frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the cycle sync mark (cycle sync mark = Sync_Mark - Ref_Mark) stored in register FDCAN_TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (cycle time = local time - Ref_Mark).

Figure 742. Cycle time and global time synchronization



The cycle time that can be read from FDCAN_TTCTC.CT is the difference of the node local time and Ref_Mark, both synchronized into the APB clock domain and truncated to 16 bit.

The global time exists for TTCAN level 0 and level 2 only, in level 1 it is invalid. The node view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3 and 4). The global time that can be read from FDCAN_TTLGT.GT is the sum of the node local time and its local offset, both synchronized into the APB clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see *Figure 743*). The node view of the global time is local time plus local offset. In a potential time master that has never received another time master reference message, Local_Offset will be 0. When a node becomes the current time master after first having received other reference messages,

Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signaled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register FDCAN_TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

Figure 743. TTCAN level 0 and level 2 drift compensation

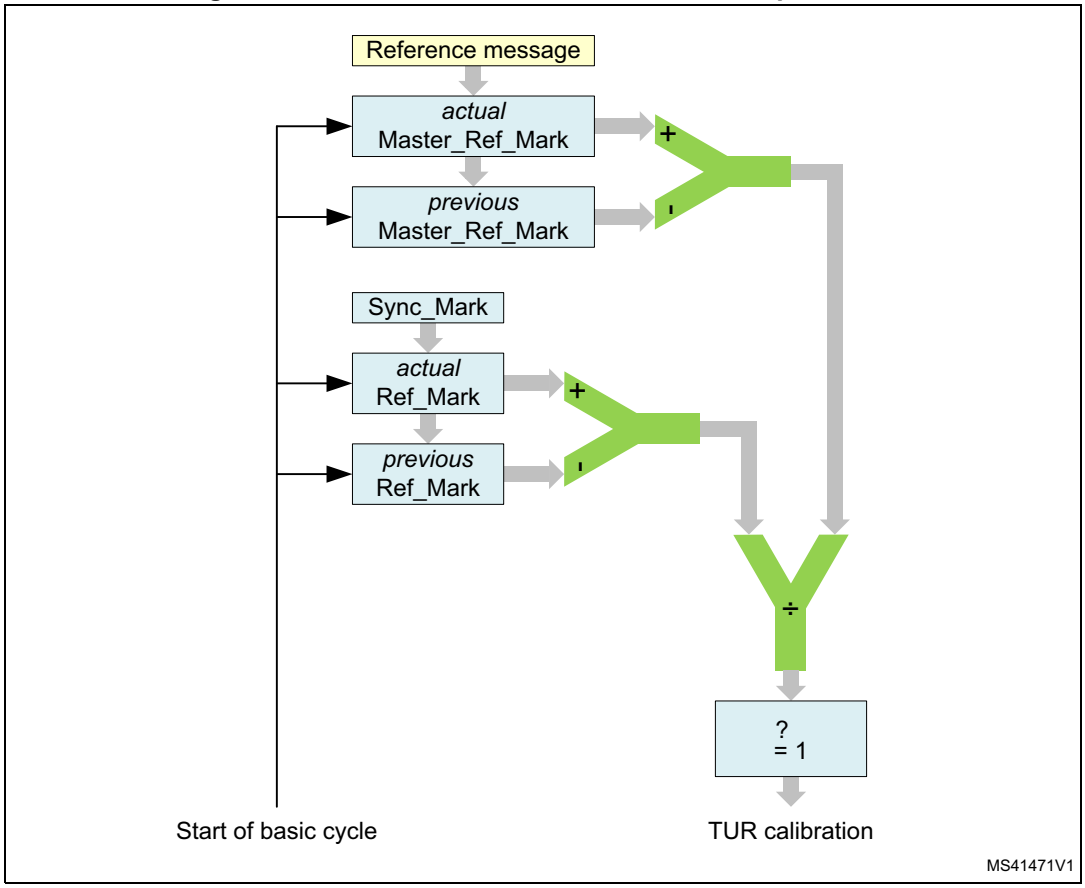


Figure 743 describes how in TTCAN levels 0 and 2 each time receiving node compensates the drift between its own local clock and the time master clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for FDCAN_TURNA.NAV is calculated. If the synchronization deviation SD = | NC - FDCAN_TURNA.NAV | ≤ SDL (synchronization deviation limit), the new value for FDCAN_TURNA.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN level 0 and level 2, FDCAN_TTOST.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN level 1, FDCAN_TOST.QCS is always 1.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit FDCAN_TTOCF.EECS.

The stop watch function (see *Section 55.3.10: Stop watch*) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated numerator configuration low to

FDCAN_TURCF.NCL (FDCAN_TURCF.DC cannot be updated during operation). The new value takes effect by writing FDCAN_TTOCN.ECS to 1.

The global time phase is adjusted by first writing the phase offset into the TT global time Preset register FDCAN_TTGTP. The new value takes effect by writing FDCAN_TTOCN.SGT to 1. The first reference message transmitted after the global time phase adjustment will have the Disc_Bit set to 1.

FDCAN_TTOST.QGTP shows whether the node global time is in phase with the time master global time. FDCAN_TTOST.QGTP is permanently 0 in TTCAN level 1 and when the synchronization deviation limit is exceeded in TTCAN level 0, 2 (FDCAN_TTOST.QCS = 0). It is temporarily 0 while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = 1 or when FDCAN_TTOST.QCS = 0.

55.3.12 TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

1. S0 - No error
2. S1 - Warning - Only notification of application, reaction application-specific.
3. S2 error - Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the reference trigger offset FDCAN_TTOST.RTO set to the maximum value of 127.
4. S3 - Severe error - Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and FDCAN_CCCR.MON is set. The S3 error condition remains active until the application updates the configuration (set FDCAN_CCCR.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by FDCAN_TTIR.ELC. The error level is monitored by FDCAN_TTOST.EL.

The TTCAN signals the following error conditions as required by ISO 11898-4:

- Config_error (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.
- Watch_Trigger_Reached (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when a watch trigger was reached because the reference message is missing.
- Application_Watchdog (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when the application failed to serve the application watchdog.The application watchdog is configured via FDCAN_TTOCF.AWL. It is served by reading register FDCAN_TTOST. When the watchdog is not served in time, bit FDCAN_TTOST.AWE and interrupt flag

FDCAN_TTIR.AW are set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode (FDCAN_CCCR.MON set to 1).

- CAN_Bus_Off (S3)
  - Entering CAN_Bus_Off state sets error level FDCAN_TTOST.EL to 11. CAN_Bus_Off state is signaled by FDCAN_PSR.BO = 1 and FDCAN_CCCR.INIT = 1.
- Scheduling_Error_2 (S2)
  - Sets error level FDCAN_TTOST.EL to 10 if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag FDCAN_TTIR.SE2 is set. The error level FDCAN_TTOST.EL is reset to 00 at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.
- Tx_Overflow (S2)
  - Sets error level FDCAN_TTOST.EL to 10 when the Tx count is equal or higher than the expected number of Tx_T riggers FDCAN_TTMLM.ENTT and a Tx_Trigger event occurs. In addition interrupt flag FDCAN_TTIR.TXO is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is no more than FDCAN_TTMLM.ENTT at the start of a new matrix cycle.
- Scheduling_Error_1 (S1)
  - Sets error level FDCAN_TTOST.EL to 01 if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than two, or if one of the MSCs of an exclusive Rx_Trigger has reached seven. In addition interrupt flag FDCAN_TTIR.SE1 is set. If within one matrix cycle none of these conditions is valid, the error level FDCAN_TTOST.EL is reset to 00.
- Tx_Underflow (S1)
  - Sets error level FDCAN_TTOST.EL to 01 when the Tx count is less than the expected number of Tx_Triggers FDCAN_TTMLM.ENTT at the start of a new matrix cycle. In addition interrupt flag FDCAN_TTIR.TXU is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is at least FDCAN_TTMLM.ENTT at the start of a new matrix cycle.

55.3.13 TTCAN message handling

Reference message

For potential time masters the identifier of the reference message is configured via FDCAN_TTRMC.RID. No dedicated Tx buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte for TTCAN level 1 (that is, the first four data bytes for TTCAN level 0 and the first four data bytes for TTCAN level 2) will be provided by the FSE.

In case the reference message Payload select FDCAN_TTRMC.RMPS is set, the rest of the reference message payload (level 1: bytes 2-8, level 0, 2: bytes 5-6) is taken from Tx buffer 0. In this case the data length DLC code from message buffer 0 is used.

Table 472. Number of data bytes transmitted with a reference message

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4

Table 472. Number of data bytes transmitted with a reference message (continued)

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
1	0	4	1	4
1	1	4 + MBO	1 + MBO	4 + MBO

To send additional payload with the reference message in level 1 a DLC > 1 has to be configured, for level 0 and level 2 a DLC > 4 is required. In addition the transmission request pending bit FDCAN_TXBRP.TRP0 of message buffer 0 must be set (see *Table 472*). In case bit FDCAN_TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the reference identifier FDCAN_TTRMC.RID is used.

Message reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see *Rx handler*).

The message status count MSC is part of the corresponding trigger memory element and has to be initialized to 0 during configuration. It is updated while the TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

Message transmission

For time-triggered message transmission the TTCAN supplies 32 dedicated Tx buffers (see *Transmit pause*). A Tx FIFO or Tx queue is not available when the FDCAN is configured for time-triggered operation (FDCAN_TTOCF.OM = 01 or 10).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading FDCAN_TXBTO
- Update the Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

Tx_Trigger_Continous

- Issue a cancellation request to reset the Tx buffer request pending bit
- Check whether the cancellation has finished by reading FDCAN_TXBCF
- Update Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

The message MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (TTCAN in error level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking FDCAN_TXBRP.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading FDCAN_TXBCF before it starts updating.

The Tx handler will transfer a message from the message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its FDCAN_TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx handler can read four words from the message RAM even at high RAM access load from other modules.

Transmission in exclusive time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx buffer request pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx buffer request pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in arbitrating time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in merged arbitrating time windows

The purpose of a merged arbitrating time window is, to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding transmission request pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or canceled by the Host), the four the request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx buffer needs to be updated.

55.3.14 TTCAN interrupt and error handling

The TT interrupt register FDCAN_TTIR consists off our segments. Each interrupt can be enabled separately by the corresponding bit in the TT interrupt enable register FDCAN_TTIE. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the FDCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the FDCAN protocol failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (level 0, 2 only) that require a reaction by the application

program. With a stop watch event triggered by a rising edge on pin fdcan_swt internal time values are captured. The trigger time mark interrupt notifies the application that a specific Time_Base_Trigger is reached. The register time mark interrupt signals that the time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) equals time mark FDCAN_TTTMK.TM. It can also be used to finish a Gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

55.3.15 Level 0

TTCAN level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. In this mode the FDCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of FDCAN_TTOCF.GEN is ignored. External event-synchronized operation is not available in level 0. A synchronized time base is maintained by transmission of reference messages.

In level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (FDCAN_TTIR.TTMI) is set when the cycle time has reached FDCAN_TTOCF.IRTO ´ 0x200, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (FDCAN_TTIR.WT) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (FDCAN_TTIR.RTMI) are also possible.

The reference message is configured as for level 2 operation. Received reference messages are recognized by the identifier configured in register FDCAN_TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- FDCAN_TTRMC
- FDCAN_TTOCF except EVTP, AWL, GEN
- FDCAN_TTMLM except ENTT, TXEW
- FDCAN_TURCF

Level 0 operation is controlled via:

- FDCAN_TTOCN except NIG, TMG, FGP, GCS, TTMIE
- FDCAN_TTGTP
- FDCAN_TTTMK
- FDCAN_TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- FDCAN_TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time FDCAN_TTOCF.IRTO 0x200
  - WT not defined by trigger memory - activated at cycle time 0xFF00

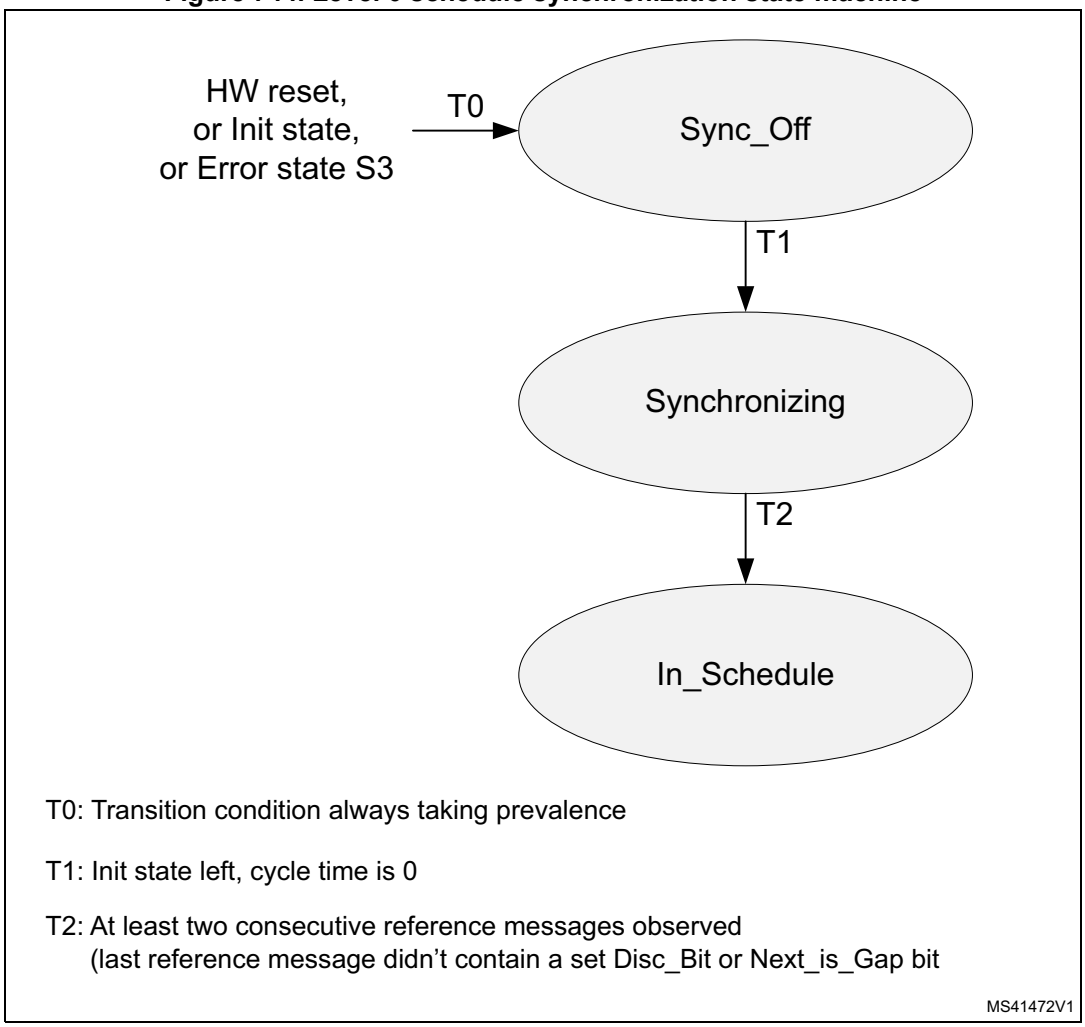
Level 0 operation is signaled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

Synchronizing

Figure 744 describes the states and the state transitions in TTCAN level 0 operation. level 0 has no In_Gap state.

Figure 744. Level 0 schedule synchronization state machine



Handling of error levels

During level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

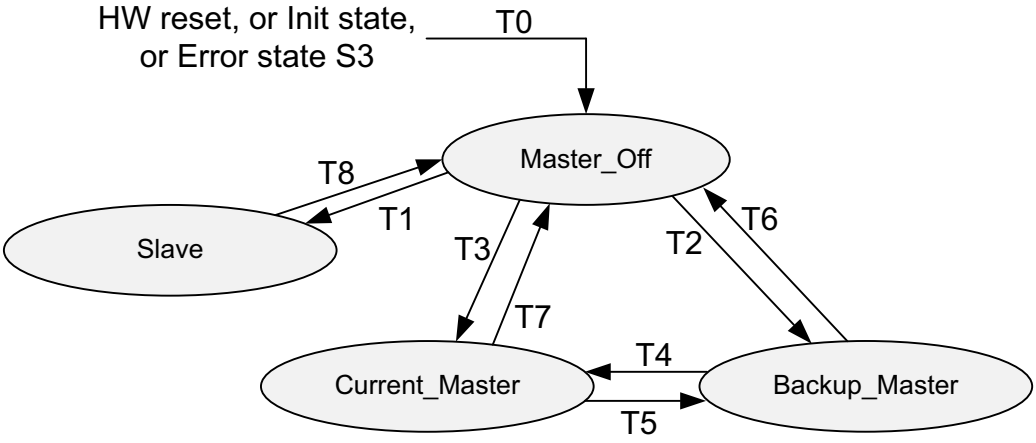
Since no S1 and S2 errors are possible, the error level can only switch between S0 (No error) and S3 (Severe error). In TTCAN level 0 an S3 error is handled differently. When error level S3 is reached, both FDCAN_TTOST.SYS and FDCAN_TTOST.MS are reset, and interrupt flags FDCAN_TTIR.GTE and FDCAN_TTIR.GTD are set.

When error level S3 (FDCAN_TTOST.EL = 11) is entered, bus monitoring mode is (contrary to TTCAN level 1 and level 2) not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

Master slave relation

Figure 745 describes the master slave relation in TTCAN level 0. In case of an S3 error the FDCAN returns to state Master_Off.

Figure 745. Level 0 master to slave relation



T0: Transition condition, always taking prevalence

T1: Reference message observed when not potential time master

T2: Reference message observed with master priority different from own master priority, Error state ≠ S3

T3: Reference message observed with master priority equal to own master priority, Error state ≠ S3

T4: Reference message observed with own master priority

T5: Reference message observed with master priority higher than own master priority

T6: Error state S3

T7: Error state S3

T8: Error state S3

MS41473V1

55.3.16 Synchronization to external time schedule

This feature can be used to synchronize the phase of the FDCAN schedule to an external schedule (e.g. that of a second TTCAN network or FlexRay network). It is applicable only when the FDCAN is current time master (FDCAN_TTOST.MS = 11).

External synchronization is controlled by event trigger input pin fdcan_evt. If bit FDCAN_TTOCN.ESCN is set, a rising edge at event trigger pin fdcan_evt the FDCAN compares its actual cycle time with the target phase value configured by FDCAN_TTGTP.CTP.

Before setting FDCAN_TTOCN.ESCN the Host has to adapt the phases of the two time schedules e.g. by using the FDCAN gap control (see *Section 55.3.9: TTCAN gap control*). When the Host sets FDCAN_TTOCN.ESCN, FDCAN_TTOSTSPL is set.

If the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is greater than 9 NTU, the phase lock bit FDCAN_TTOST.SPL is reset, and interrupt flag FDCAN_TTIR.CSM is set.

FDCAN_TTOST.SPL is also reset (and FDCAN_TTIR.CSM is set), when another node becomes time master.

If both FDCAN_TTOST.SPL and FDCAN_TTOCN.ESCN are set, and if the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is lower or equal to nine NTU, the phase lock bit FDCAN_TTOST.SPL remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at event trigger pin fdcan_evt is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with FDCAN_*TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.*

55.3.17 FDCAN Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx buffer / FIFO element is shown in *Table 473*, the description is provided in *Table 474*.

Table 473. Rx buffer and FIFO element

Bit	31			24	23			16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]								
R1	ANMF	FIDX[6:0]			Res.	FDF	BRS	DLC[3:0]	RXTS[15:0]			
R2	DB3[7:0]				DB2[7:0]				DB1[7:0]		DB0[7:0]	
R3	DB7[7:0]				DB6[7:0]				DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮				⋮			
Rn	DBm[7:0]				DBm-1[7:0]				DBm-2[7:0]		DBm-3[7:0]	

The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_RXESC.

Table 474. Rx buffer and FIFO element description

Field	Description
R0 bit 31 ESI	Error state indicator 0: Transmitting node is error active 1: Transmitting node is error passive
R0 bit 30 XTD	Extended identifier Signals to the Host whether the received frame has a standard or extended identifier. 0: 11-bit standard identifier 1: 29-bit extended identifier
R0 bit 29 RTR	Remote transmission request Signals to the Host whether the received frame is a data frame or a remote frame. 0: Received frame is a data frame 1: Received frame is a remote frame

Table 474. Rx buffer and FIFO element description (continued)

Field	Description
R0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames may be enabled via FDCAN_GFC.ANFS and FDCAN_GFC.ANFE. 0: Received frame matching filter index FIDX 1: Received frame did not match any Rx filter element
R1 bits 30:24 FIDX[6:0]	Filter index 0-127 = index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to FDCAN_SIDFC.LSS. - 1 or FDCAN_XIDFC.LSE. - 1.
R1 bit 21 FDF	FD Format 0: Standard frame format 1: FDCAN frame format (new DLC-coding and CRC)
R1 bit 20 BRS	Bitrate Switch 0: Frame received without bitrate switching 1: Frame received with bitrate switching
R1 bits 19:16 DLC[3:0]	Data length code 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 bits 15:0 RXTS[15:0]	Rx timestamp Timestamp counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.
R2 bits 31:24 DB3[7:0]	Data Byte 3
R2 bits 23:16 DB2[7:0]	Data Byte 2
R2 bits 15:8 DB1[7:0]	Data Byte 1
R2 bits 7:0 DB0[7:0]	Data Byte 0
R3 bits 31:24 DB7[7:0]	Data Byte 7
R3 bits 23:16 DB6[7:0]	Data Byte 6
R3 bits 15:8 DB5[7:0]	Data Byte 5
R3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮

Table 474. Rx buffer and FIFO element description (continued)

Field	Description
Rn bits 31:24 DBm[7:0]	Data Byte m
Rn bits 23:16 DBm-1[7:0]	Data Byte m-1
Rn bits 15:8 DBm-2[7:0]	Data Byte m-2
Rn bits 7:0 DBm-3[7:0]	Data Byte m-3

55.3.18 FDCAN Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx queue. The Tx handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx queue by evaluating the Tx buffer configuration FDCAN_TXBC.TFQS and FDCAN_TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_TXESC.

Table 475. Tx buffer and FIFO element

Bit	31 24	23 16	15 8	7 0
T0	ESI, XTD, RTR, ID[28:0]			
T1	MM[7:0]	EFC, Res., FDF, BPS, DLC[3:0]	Reserved	
T2	DB3[7:0]	DB2[7:0]	DB1[7:0]	DB0[7:0]
T3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
⋮	⋮	⋮	⋮	
Tn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Table 476. Tx buffer element description

Field	Description
T0 bit 31 ESI(1)	Error state indicator 0: ESI bit in CAN FD format depends only on error passive flag 1: ESI bit in CAN FD format transmitted recessive
T0 bit 30 XTD	Extended identifier 0: 11-bit standard identifier 1: 29-bit extended identifier
T0 bit 29 RTR(2)	Remote transmission request 0: Transmit data frame 1: Transmit remote frame

Table 476. Tx buffer element description (continued)

Field	Description
T0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 bit 23 EFC	Event FIFO control 0: Don't store Tx events 1: Store Tx events
T1 bit 21 FDF	FD Format 0: Frame transmitted in classic CAN format 1: Frame transmitted in CAN FD format
T1 bit 20 BRS(3)	Bitrate switching 0: CAN FD frames transmitted without bitrate switching 1: CAN FD frames transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 bits 31:24 DB3[7:0]	Data Byte 3
T2 bits 23:16 DB2[7:0]	Data Byte 2
T2 bits 15:8 DB1[7:0]	Data Byte 1
T2 bits 7:0 DB0[7:0]	Data Byte 0
T3 bits 31:24 DB7[7:0]	Data Byte 7
T3 bits 23:16 DB6[7:0]	Data Byte 6
T3 bits 15:8 DB5[7:0]	Data Byte 5
T3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮
Tn bits 31:24 DBm[7:0]	Data Byte m
Tn bits 23:16 DBm-1[7:0]	Data Byte m-1

Table 476. Tx buffer element description (continued)

Field	Description
Tn bits 15:8 DBm-2[7:0]	Data Byte m-2
Tn bits 7:0 DBm-3[7:0]	Data Byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if FDCAN_CCCR.FDOE enables the transmission in CAN FD format.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled FDCAN_CCCR.FDOE = 1. Bit BRS is only evaluated when in addition FDCAN_CCCR.BRSE = 1.

55.3.19 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register FDCAN_TXEFS.

Table 477. Tx Event FIFO element

Bit	31			24	23					16	15		8	7		0
E0	ESI	XTD	RTR		ID[28:0]											
E1	MM[7:0]				ET[1:0]		EDL	BRS	DLC[3:0]		TXTS[15:0]					

Table 478. Tx Event FIFO element description

Field	Description
E0 bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 bit 30 XTD	Extended Identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.

Table 478. Tx Event FIFO element description (continued)

Field	Description
E1 bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 bit 20 BRS	Bitrate switching – 0: Frame transmitted without bitrate switching – 1: Frame transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code – 0 - 8: Frame with 0-8 data bytes transmitted – 9 - 15: Frame with eight data bytes transmitted
E1 bits 15:0 TXTS[15:0]	Tx timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.

55.3.20 FDCAN standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FDCAN_SIDFC.FLSSA plus the index of the filter element (0 … 127).

Table 479. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 480. Standard message ID filter element field description

Field		Description
Bit 31:30 SFT[1:0](1)		Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]		Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer or as debug message, configuration of FDCAN_SFT[1:0] ignored
Bits 26:16 SFID1[10:0]		Standard filter ID 1 First ID of standard ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 15:0	SFID2[15:10]	Standard filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: – SFEC = 001 ... 110 Second ID of standard ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	SFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	SFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_ttcan_hclk period in case the filter matches. SFID2[8] is used by the calibration unit.
	SFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

1. With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = “000”).

Note: In case a reserved value is configured, the filter element is considered disabled.

55.3.21 FDCAN extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended message ID filter element, its address is the filter list extended start address FDCAN_XIDFC.FLESA plus two times the index of the filter element (0 … 63).

Table 481. Extended message ID filter element

Bit	31	30	29	28	0
F0	EFEC[2:0]			EFID1[28:0]	
F1	EFTI[1:0]		Reserved	EFID2[28:0]	

Table 482. Extended message ID filter element field description

Field	Description
F0 bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer, configuration of EFT[1:0] ignored
F0 bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only FDCAN_XIDAM masking mechanism.
F1 bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), FDCAN_XIDAM mask not applied

Table 482. Extended message ID filter element field description (continued)

Field		Description
F1 bits 28:0	EFID2[10:0]	Extended filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: – SFEC = 001 ... 110 Second ID of extended ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	EFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	EFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_ttcan_hclk period in case the filter matches. EFID2[8] interface is used by the calibration unit.
	EFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

55.3.22 FDCAN trigger memory element

Up to 64 trigger memory elements can be configured. When accessing a trigger memory element, its address is the trigger memory start address FDCAN_TTTMC.TMSA plus the index of the trigger memory element (0 … 63).

Table 483. Trigger memory element

Bit	31 24	23 16	15 8	7 0
T0	TM[15:0]	Res. CC[6:0]	Res. TMIN TMEX	TYPE[3:0]
T1	Res.	FTYPE MNR[6:0]	Res.	MSC[2:0]

Table 484. Trigger memory element description

Field	Description
T0 bits 31:16 TM[15:0]	Time mark Cycle time for which the trigger becomes active.
T0 bit 14:8 CC[6:0]	Cycle code Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. – 0b000000x valid for all cycles – 0b000001c valid every 2^{nd} cycle at cycle count mod2 = c – 0b00001cc valid every 4^{th} cycle at cycle count mod4 = cc – 0b0001ccc valid every 8^{th} cycle at cycle count mod8 = ccc – 0b001cccc valid every 16^{th} cycle at cycle count mod16 = cccc – 0b01ccccc valid every 32^{nd} cycle at cycle count mod32 = ccccc – 0b1cccccc valid every 64^{th} cycle at cycle count mod64 = cccccc

Table 484. Trigger memory element description (continued)

Field	Description
T0 bit 5 TMIN	Time mark event internal – 0: No action – 1: FDCAN_TTIR.TTMI is set when trigger memory element becomes active
T0 bit 4 TMEX	Time mark event external – 0: No action – 1: Pulse at output m_ttcan_tmp with the length of one period is generated when the time ark of the trigger memory element becomes active and FDCAN_TTOCN.TTMIE = 1
T0 bit 3:0 TYPE[3:0]	Trigger type – 0000 Tx_Ref_Trigger - valid when not in Gap – 0001 Tx_Ref_Trigger_Gap - valid when in Gap – 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window – 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window – 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window – 0101 Tx_Trigger_Merged - starts a merged arbitration window – 0110 Watch_Trigger - valid when not in Gap – 0111 Watch_Trigger_Gap - valid when in Gap – 1000 Rx_Trigger - check for reception – 1001 Time_Base_Trigger - only control TMIN, TMEX – 1010 … 1111=End_of_List - illegal type, causes configuration error
T1 bit 23FTYPE	Filter type – 0: 11-bit standard message ID – 1: 29-bit extended message ID
T1 bit 22:16 MNR[6:0](1)	Message Number – Transmission: trigger is valid for configured Tx buffer number. Valid values are 0 to 31. – Reception: trigger is valid for standard/extended message ID filter element number. Valid values are, respectively 0 to 63 and 0 to 127.
T1 bits 2:0 MSC[2:0]	Message status count Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). – 0-7= Actual status

1. The trigger memory elements have to be written when the FDCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual reference trigger offset (TTOST[RTO]).

55.4 FDCAN registers

55.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 3

Bits 27:24 **STEP[3:0]**: Step of core release = 2

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Timestamp Year = 4

Bits 15:8 **MON[7:0]**: Timestamp Month = 12

Bits 7:0 **DAY[7:0]**: Timestamp Day =18

55.4.2 FDCAN Endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endiannes Test value
The endianness test value is 0x8765 4321.

55.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is dedicated to data bit timing phase and only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN time quantum may be programmed in the range from 1 to 32 FDCAN clock periods. tq = (DBRP + 1) FDCAN clock periods.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2. Therefore the length of the bit time is (DTSEG1 + DTSEG2 + 3) tq for programmed values, or (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) tq for functional values.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver Delay Compensation
0: Transceiver Delay Compensation disabled
1: Transceiver Delay Compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bItrate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of these quanta. Valid values for the baudrate prescaler are 0 to 31. The hardware interpreters this value as the programmed value plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS1} = (DTSEG1 + 1) x tq.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2 + 1) x tq.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Should always be smaller than DTSEG2, valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{SJW} = (DSJW + 1) x tq.

Note: With a FDCAN clock of 8 MHz, the reset value 0x00000A33 configures the FDCAN for a fast bitrate of 500 kbit/s.

55.4.4 FDCAN test register (FDCAN_TEST)

Write access to this register has to be enabled by setting bit FDCAN_CCCR.TEST to 1. All register functions are set to their reset values when bit FDCAN_CCCR.TEST is reset.

Loop back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX differently from 00 may disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin
Monitors the actual value of transmit pin FDCANx_RX
0: The CAN bus is dominant (FDCANx_RX = 0)
1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin
00: Reset value , FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time
01: Sample point can be monitored at pin FDCANx_TX
10: Dominant (0) level at pin FDCANx_TX
11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop back mode
0: Reset value, loop back mode is disabled
1: Loop back mode is enabled (see *Test modes*)

Bits 3:0 Reserved, must be kept at reset value.

55.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured by the FDCAN_RWD.WDC bits.

The counter is reloaded with FDCAN_RWD.WDC bits when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops and interrupt flag FDCAN_IR.WDI bit is set. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00 the counter is disabled.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

55.4.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and resetting of single bits see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP**:
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: disabled
1: enabled

Bit 13 **EFBI**: Edge Filtering during bus Integration
0: Edge filtering disabled
1: Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bitrate Switching
0: Bitrate switching for transmissions disabled
1: Bitrate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, register TEST holds reset values
1: Test mode, write access to register TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
Bit MON can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the Host at any time.
0: Bus monitoring mode is disabled
1: Bus monitoring mode is enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop is requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN may be set in power down by stopping APB clock and kernel clock

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bitrates. The application tests different bitrates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the software at any time.
If the FDCAN is connected to a clock calibration on CAN unit, ASM bit is set by hardware as long as the calibration is not completed.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers
1: The CPU has write access to the protected configuration registers
(while FDCAN_CCCR.INIT = 1)

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization is started

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

55.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0000 0A33

This register is dedicated to the nominal bit timing used during the arbitration phase, and is only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 81 tq. The CAN time quantum may be programmed in the range of [1 … 1024] FDCAN kernel clock periods.

tq = (BRP + 1) FDCAN clock period fdcan_tq_ck

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Should be smaller than NTSEG2, valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{SJW} = (NSJW + 1) x tq.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 24:16 **NBRP[8:0]**: Bitrate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values are 0 to 511. The value used by the hardware is the one programmed, incremented by 1.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS1} = (NTSEG1 + 1) x tq.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (NTSEG2 + 1) x tq.

Note: With a CAN kernel clock of 8 MHz, the reset value of 0x00000A33 configures the FDCAN for a bitrate of 125 kbit/s.

55.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 … 16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS = 10).

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value used (tim3_cnt[0:15])

11: Same as 00.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

55.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When FDCAN_TSCC.TSS = 01, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. A wrap around sets interrupt flag FDCAN_IR.TSW. Write access resets the counter to 0. When FDCAN_TSCC.TSS = 10, TSC reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 not caused by write access to FDCAN_TSCV.

55.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in Continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored.

00: Continuous operation

01: Timeout controlled by Tx event FIFO

10: Timeout controlled by Rx FIFO 0

11: Timeout controlled by Rx FIFO 1

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **ETOC**: Enable timeout counter

0: Timeout counter disabled

1: Timeout counter enabled

This is a protected write bit, write access is possible only when the bit 1 (CCE) and bit 0 (INIT) of FDCAN_CCCR register are set to 1.

For more details see *Timeout counter*.

55.4.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. When decremented to 0, interrupt flag FDCAN_IR.TOO is set and the timeout counter is stopped. Start and reset/restart conditions are configured via FDCAN_TOCC.TOS.

55.4.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	TREC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag FDCAN_IR.ELO.

Bit 15 **RP**: Receive error passive
0: The receive error counter is below the error passive level of 128
1: The receive error counter has reached the error passive level of 128

Bits 14:8 **TREC[6:0]:** Receive error counter
Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter
Actual state of the transmit error counter, values between 0 and 255.
When FDCAN_CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

55.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs_r	rs_r	rs_r	r	r	r	r	r	rs_r	rs_r	rs_r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter Delay Compensation value
Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and FDCAN_TDCR.TDCO. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event
0: No protocol exception event occurred since last read access
1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message
This bit is set independent of acceptance filtering.
0: Since this bit was reset by the CPU, no FDCAN message has been received
1: Message in FDCAN format with EDL flag set has been received
Access type is RX: reset on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message
This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not have its BRS flag set
1: Last received FDCAN message had its BRS flag set
Access type is RX: reset on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message
This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not ha ve its ESI flag set
1: Last received FDCAN message had its ESI flag set
Access type is RX: reset on read.

Bits 10:8 **DLEC[2:0]**: Data Last error code
Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to 0 when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.
Access type is RS: set on read.

Bit 7 **BO**: Bus_Off status
0: The FDCAN is not Bus_Off
1: The FDCAN is in Bus_Off state

Bit 6 **EW**: Warning status
0: Both error counters are below the Error_Warning limit of 96
1: At least one of error counter has reached the Error_Warning limit of 96

Bit 5 **EP**: Error Passive

0: The FDCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected

1: The FDCAN is in the Error_Passive state

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication

01: Idle: node is neither receiver nor transmitter

10: Receiver: node is operating as receiver

11: Transmitter: node is operating as transmitter

Bits 2:0 **LEC[2:0]:** Last error code

The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error.

000: No error: No error occurred since LEC has been reset by successful reception or transmission.

001: Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error: A fixed format part of a received frame has the wrong format.

011: AckError: The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: NoChange: Any read access to the protocol status register re-initializes the LEC to ‘7. When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is RS: set on read.

Note: When a frame in FDCAN format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a FDCAN CRC sequence will be shown as a Form error, not Stuff error

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting FDCAN_CCCR.INIT. If the device goes Bus_Off, it will set FDCAN_CCCR.INIT of its own, stopping all bus activities. Once FDCAN_CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the error management counters will be reset. During the waiting time after the reset of FDCAN_CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0 error code is written to FDCAN_PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. FDCAN_ECR.REC is used to count these sequences.

55.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 mtq.

These are protected write bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

These are protected write bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

55.4.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position.

Writing a 0 has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.	Res.	DRX	TOO	MRAF	TSW
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARA**: Access to Reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 28 **PED**: Protocol error in data phase (data Bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (PSR.DLEC different from 0,7)

Bit 27 **PEA**: Protocol error in Arbitration phase (Nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (PSR.LEC different from 0,7)

Bit 26 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 25 **BO**: Bus_Off status
0: Bus_Off status unchanged
1: Bus_Off status changed

Bit 24 **EW**: Warning status
0: Error_Warning status unchanged
1: Error_Warning status changed

Bit 23 **EP**: Error Passive
0: Error_Passive status unchanged
1: Error_Passive status changed

Bit 22 **ELO**: Error logging Overflow
0: CAN error logging counter did not overflow
1: Overflow of CAN error logging counter occurred

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRX**: Message stored to dedicated Rx buffer
The flag is set whenever a received message has been stored into a dedicated Rx buffer.
0: No Rx buffer updated
1: At least one received message stored into a Rx buffer

Bit 18 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 17 **MRAF**: Message RAM Access Failure

The flag is set when the Rx handler

- Has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- Was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated or the New data flag for a dedicated Rx buffer is not set. The partly stored message is overwritten when the next message is stored to this location.

The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure the M_TTCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the Host CPU has to reset FDCAN_CCCR.ASM.

0: No message RAM access failure occurred

1: Message RAM access failure occurred

Bit 16 **TSW**: Timestamp wraparound

0: No timestamp counter wraparound

1: Timestamp counter wraparound

Bit 15 **TEFL**: Tx event FIFO element Lost

0: No Tx event FIFO element lost

1: Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero

Bit 14 **TEFF**: Tx event FIFO Full

0: Tx event FIFO not full

1: Tx event FIFO full

Bit 13 **TEFW**: Tx event FIFO Watermark Reached

0: Tx event FIFO fill level below watermark

1: Tx event FIFO fill level reached watermark

Bit 12 **TEFN**: Tx event FIFO New Entry

0: Tx event FIFO unchanged

1: Tx handler wrote Tx event FIFO element

Bit 11 **TFE**: Tx FIFO Empty

0: Tx FIFO non-empty

1: Tx FIFO empty

Bit 10 **TCF**: Transmission cancellation finished

0: No transmission cancellation finished

1: Transmission cancellation finished

Bit 9 **TC**: Transmission Completed

0: No transmission completed

1: Transmission completed

Bit 8 **HPM**: High priority message

0: No high priority message received

1: High priority message received

Bit 7 **RF1L**: Rx FIFO 1 message Lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero

Bit 6 **RF1F**: Rx FIFO 1 Full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 5 **RF1W**: Rx FIFO 1 Watermark Reached
0: Rx FIFO 1 fill level below watermark
1: Rx FIFO 1 fill level reached watermark

Bit 4 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 3 **RF0L**: Rx FIFO 0 message Lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero

Bit 2 **RF0F**: Rx FIFO 0 Full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 1 **RF0W**: Rx FIFO 0 Watermark Reached
0: Rx FIFO 0 fill level below watermark
1: Rx FIFO 0 fill level reached watermark

Bit 0 **RF0N**: Rx FIFO 0 New message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

55.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register will be signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAE**: Access to Reserved address enable

Bit 28 **PEDE**: Protocol error in data phase enable

Bit 27 **PEAE**: Protocol error in Arbitration phase enable

Bit 26 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 25 **BOE**: Bus_Off status
0: Interrupt disabled
1: Interrupt enabled

Bit 24 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 21 **BEUE**: Bit error uncorrected interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 20 **BECE**: Bit error corrected interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **DRXE**: Message stored to dedicated Rx buffer interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TEFWE**: Tx event FIFO watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **HPME**: High priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1WE**: Rx FIFO 1 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0WE**: Rx FIFO 0 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

55.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific interrupt flag from the interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAL**: Access to reserved address line

Bit 28 **PEDL**: Protocol error in data phase line

Bit 27 **PEAL**: Protocol error in arbitration phase line

Bit 26 **WDIL**: Watchdog interrupt Line

Bit 25 **BOL**: Bus_Off status

Bit 24 **EWL**: Warning status interrupt Line

Bit 23 **EPL**: Error passive interrupt line

Bit 22 **ELOL**: Error logging overflow interrupt line

Bit 21 **BEUL**: Bit error uncorrected interrupt line

Bit 20 **BECL**: Bit error corrected interrupt line

Bit 19 **DRXL**: Message stored to dedicated Rx buffer interrupt line

Bit 18 **TOOL**: Timeout occurred interrupt Line

Bit 17 **MRAFL**: Message RAM access failure interrupt line

Bit 16 **TSWL**: Timestamp wraparound interrupt line

Bit 15 **TEFLL**: Tx event FIFO element Lost interrupt line

Bit 14 **TEFFL**: Tx event FIFO full interrupt line

Bit 13 **TEFWL**: Tx event FIFO watermark reached interrupt line

Bit 12 **TEFNL**: Tx event FIFO new entry interrupt line

Bit 11 **TFEL**: Tx FIFO empty interrupt Line

Bit 10 **TCFL**: Transmission cancellation finished interrupt line

Bit 9 **TCL**: Transmission completed interrupt line

Bit 8 **HPML**: High priority message interrupt line

Bit 7 **RF1LL**: Rx FIFO 1 message lost interrupt line

Bit 6 **RF1FL**: Rx FIFO 1 full interrupt line

Bit 5 **RF1WL**: Rx FIFO 1 watermark reached interrupt line

Bit 4 **RF1NL**: Rx FIFO 1 new message interrupt line

Bit 3 **RF0LL**: Rx FIFO 0 message lost interrupt line

Bit 2 **RF0FL**: Rx FIFO 0 full interrupt line

Bit 1 **RF0WL**: Rx FIFO 0 watermark reached interrupt line

Bit 0 **RF0NL**: Rx FIFO 0 new message interrupt line

55.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming bits EINT0 and EINT1.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt Line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt Line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

55.4.19 FDCAN global filter configuration register (FDCAN_GFC)

Global settings for message ID filtering. The global filter configuration register controls the filter path for standard and extended messages as described in *Figure 735: Standard message ID filter path* and *Figure 736: Extended message ID filter path*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept non-matching frames standard

Defines how received messages with 11-bit ID that do not match any element of the filter list are treated.

00: Accept in Rx FIFO 0

01: Accept in Rx FIFO 1

10: Reject

11: Reject

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended

Defines how received messages with 29-bit ID that do not match any element of the filter list are treated.

00: Accept in Rx FIFO 0

01: Accept in Rx FIFO 1

10: Reject

11: Reject

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 1 **RRFS**: Reject remote frames standard

0: Filter remote frames with 11-bit standard ID

1: Reject all remote frames with 11-bit standard ID

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **RRFE**: Reject remote frames extended

0: Filter remote frames with 29-bit standard ID

1: Reject all remote frames with 29-bit standard ID

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

55.4.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC)

Settings for 11-bit standard message ID filtering.The standard ID filter configuration register controls the filter path for standard messages as described in *Figure 735*.

Address offset: 0x0084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLSSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSS[7:0]**: List size standard

0: No standard message ID filter

1-128: Number of standard message ID filter elements

>128: Values greater than 128 are interpreted as 128.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLSSA[13:0]**: Filter list standard start address

Start address of standard message ID filter list (32-bit word address, see *Table 479: Standard message ID filter element*).These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

55.4.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC)

Settings for 29-bit extended message ID filtering. The FDCAN extended ID filter configuration register controls the filter path for standard messages as described in *Figure 736: Extended message ID filter path*.

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLESA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSE[7:0]**: List size extended

0: No standard message ID filter

1-128: Number of standard message ID filter elements

>128: Values greater than 128 are interpreted as 128.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLESA[13:0]**: Filter list standard start address

Start address of standard message ID filter list (32-bit word address, see *Table 481: Extended message ID filter element*).

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

55.4.22 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0090

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID Mask

For acceptance filtering of extended frames the extended ID AND Mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to 1 the mask is not active.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

55.4.23 FDCAN high priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLST	FIDX[6:0]							MSI[1:0]		BIDX[5:0]					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element.
0: Standard filter list
1: Extended filter list

Bits 14:8 **FIDX[6:0]**: Filter index
Index of matching filter element. Range is 0 to FDCAN_SIDFC[LSS] - 1 or FDCAN_XIDFC[LSE] - 1.

Bits 7:6 **MSI[1:0]**: Message Storage Indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:0 **BIDX[5:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

55.4.24 FDCAN new data 1 register (FDCAN_NDAT1)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[31:0]**: New data[31:0]

The register holds the New data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

55.4.25 FDCAN new data 2 register (FDCAN_NDAT2)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[63:32]**: New data[63:32]

The register holds the New data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

55.4.26 FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C)

Address offset: 0x00A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F0OM	F0WM[6:0]							Res.	F0S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F0OM**: FIFO 0 operation mode
FIFO 0 can be operated in blocking or in overwrite mode.
0: FIFO 0 blocking mode
1: FIFO 0 overwrite mode

Bits 30:24 **F0WM[6:0]**: FIFO 0 Watermark
0: Watermark interrupt disabled
1-64: Level for Rx FIFO 0 watermark interrupt (FDCAN_IR.RF0W)
>64: Watermark interrupt disabled
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F0S[6:0]**: Rx FIFO 0 size
0: No Rx FIFO 0
1-64: Number of Rx FIFO 0 elements
>64: Values greater than 64 are interpreted as 64
The Rx FIFO 0 elements are indexed from 0 to F0S-1.

Bits 15:2 **F0SA[13:0]**: Rx FIFO 0 start address
Start address of Rx FIFO 0 in message RAM (32-bit word address, see *Figure 734: Message RAM configuration*).

Bits 1:0 Reserved, must be kept at reset value.

55.4.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x00A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	F0PI[5:0]					
						rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F0GI[5:0]						Res.	F0FL[6:0]						
		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message Lost
This bit is a copy of interrupt flag FDCAN_IR.RF0L. When FDCAN_IR.RF0L is reset, this bit is also reset.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero

Bit 24 **F0F**: Rx FIFO 0 Full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F0PI[5:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F0GI[5:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F0FL**[6:0]: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0, range 0 to 64.

55.4.28 FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x00A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F0AI[5:0]**: Rx FIFO 0 acknowledge index
After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 get index FDCAN_RXF0S.F0GI to F0AI + 1 and update the FIFO 0 fill level FDCAN_RXF0S.F0FL.

55.4.29 FDCAN Rx buffer configuration register (FDCAN_RXBC)

Address offset: 0x00AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:2 **RBSA[13:0]**: Rx buffer start address
Configures the start address of the Rx buffers section in the message RAM (32-bit word address). Also used to reference debug messages A, B, C.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

55.4.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C)

Address offset: 0x00B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F1OM	F1WM[6:0]							Res.	F1S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F1SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F1OM**: FIFO 1Operation mode
FIFO 1 can be operated in blocking or in overwrite mode.
0: FIFO 1 blocking mode
1: FIFO 1 overwrite mode

Bits 30:24 **F1WM[6:0]**: Rx FIFO 1 Watermark
0: Watermark interrupt disabled
1-64: Level for Rx FIFO 1 watermark interrupt (FDCAN_IR.RF1W)
>64: Watermark interrupt disabled.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F1S[6:0]**: Rx FIFO 1 size
0: No Rx FIFO 1
1-64: Number of Rx FIFO 1 elements
>64: Values greater than 64 are interpreted as 64
The Rx FIFO 1 elements are indexed from 0 to F1S - 1.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **F1SA[13:0]**: Rx FIFO 1 start address
start address of Rx FIFO 1 in message RAM (32-bit word address, see *Figure 734: Message RAM configuration*).
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

55.4.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x00B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMS[1:0]		Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	F1PI[5:0]					
r	r					r	r			r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F1GI[5:0]						Res.	F1FL[6:0]						
		r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:30 **DMS[1:0]**: Debug message status
00: Idle state, wait for reception of debug messages, DMA request is cleared
01: Debug message A received
10: Debug messages A, B received
11: Debug messages A, B, C received, DMA request is set

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message Lost
This bit is a copy of interrupt flag FDCAN_IR.RF1L. When FDCAN_IR.RF1L is reset, this bit is also reset.
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.

Bit 24 **F1F**: Rx FIFO 1 Full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F1PI[5:0]**: Rx FIFO 1 put index
Rx FIFO 1 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F1GI[5:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F1FL[6:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1, range 0 to 64

55.4.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x00B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F1AI[5:0]**: Rx FIFO 1 acknowledge index

After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 get index FDCAN_RXF1S.F1GI. to F1AI + 1 and update the FIFO 1 fill level FDCAN_RXF1S.F1FL.

55.4.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC)

Configures the number of data bytes belonging to an Rx buffer / Rx FIFO element. Data field sizes higher than 8 bytes are intended for CAN FD operation only.

Address offset: 0x00BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RBDS[2:0]			Res.	F1DS[2:0]			Res.	F0DS[2:0]		
					r	r	r		r	r	r		r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **RBDS[2:0]**: Rx buffer data Field size:

000: 8 byte data field

001: 12 byte data field

010: 16 byte data field

011: 20 byte data field

100: 24 byte data field

101: 32 byte data field

110: 48 byte data field

111: 64 byte data field

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **F1DS[2:0]**: Rx FIFO 0 data Field size:
000: 8 byte data field
001: 12 byte data field
010: 16 byte data field
011: 20 byte data field
100: 24 byte data field
101: 32 byte data field
110: 48 byte data field
111: 64 byte data field

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **F0DS[2:0]**: Rx FIFO 1 data Field size:
000: 8 byte data field
001: 12 byte data field
010: 16 byte data field
011: 20 byte data field
100: 24 byte data field
101: 32 byte data field
110: 48 byte data field
111: 64 byte data field

55.4.34 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TFQM**: Tx FIFO/queue mode.
0: Tx FIFO operation
1: Tx queue operation.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 29:24 **TFQS[5:0]**: Transmit FIFO/queue size.
0: No Tx FIFO/queue
1-32: Number of Tx buffers used for Tx FIFO/queue
>32: Values greater than 32 are interpreted as 32.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **NDTB[5:0]**: Number of dedicated transmit buffers.
0: No dedicated Tx buffers
1-32: Number of dedicated Tx buffers
>32: Values greater than 32 are interpreted as 32.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TBSA[13:0]**: Tx buffers start address.
Start address of Tx buffers section in message RAM (32-bit word address, see *Figure 734*).
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

Note: The sum of TFQS and NDTB cannot be larger than 32. There is no check for erroneous configurations. The Tx buffers section in the message RAM starts with the dedicated Tx buffers.

55.4.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in register FDCAN_TXBRP. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]				
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue Full
0 Tx FIFO/queue not full
1 Tx FIFO/queue full

Bits 20:16 **TFQPI[4:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 31

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **TFGI[4:0]**:
Tx FIFO get index.
Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TFFL[5:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1).

Note: In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get index indicate the number of the Tx buffer starting with the first dedicated Tx buffers. For example: For a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

55.4.36 FDCAN Tx buffer element size configuration register (FDCAN_TXESC)

Configures the number of data bytes belonging to a Tx buffer element. Data field sizes >8 bytes are intended for CAN FD operation only.

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TBDS[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TBDS[2:0]**: Tx buffer data Field size:
000: 8 byte data field
001: 12 byte data field
010: 16 byte data field
011: 20 byte data field
100: 24 byte data field
101: 32 byte data field
110: 48 byte data field
111: 64 byte data field

55.4.37 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TRP[31:0]**:Transmission request pending.

Each Tx buffer has its own transmission request pending bit. The bits are set via register FDCAN_TXBAR. The bits are reset after a requested transmission has completed or has been canceled via register FDCAN_TXBCR.

FDCAN_TXBRP bits are set only for those Tx buffers configured via FDCAN_TXBC. After a FDCAN_TXBRP bit has been set, a Tx scan (see *Filtering for Debug messages*) is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of register FDCAN_TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding FDCAN_TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled via FDCAN_TXBCF

– after successful transmission together with the corresponding FDCAN_TXBTO bit
– when the transmission has not yet been started at the point of cancellation
– when the transmission has been aborted due to lost arbitration
– when an error occurred during frame transmission

In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately, the corresponding FDCAN_TXBRP bit is reset.

55.4.38 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AR[31:0]**:Add request

Each Tx buffer has its own add request bit. Writing a 1 will set the corresponding add request bit; writing a 0 has no impact. This enables the Host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. FDCAN_TXBAR bits are set only for those Tx buffers configured via FDCAN_TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

55.4.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CR[31:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 will set the corresponding cancellation request bit; writing a 0 has no impact.

This enables the Host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. FDCAN_TXBCR bits are set only for those Tx buffers configured via FDCAN_TXBC. The bits remain set until the corresponding FDCAN_TXBRP bit is reset.

0: No cancellation pending

1: Cancellation pending

55.4.40 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TO[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TO[31:0]**: Transmission occurred.

Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

55.4.41 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CF[31:0]**: Cancellation finished

Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

55.4.42 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TIE[31:0]**: Transmission interrupt enable
Each Tx buffer has its own transmission interrupt enable bit.
0: Transmission interrupt disabled
1: Transmission interrupt enable

55.4.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_ TXBCIE)

Address offset: 0x00E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFIE[31:0]**: Cancellation finished interrupt enable.
Each Tx buffer has its own cancellation finished interrupt enable bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

55.4.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)

Address offset: 0x00F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **EFWM[5:0]**: Event FIFO Watermark
0: Watermark interrupt disabled
1-32: Level for Tx event FIFO watermark interrupt (FDCAN_IR.TEFW)
>32: Watermark interrupt disabled
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **EFS[5:0]**: Event FIFO size.
0: Tx event FIFO disabled
1-32: Number of Tx event FIFO elements
>32: Values greater than 32 are interpreted as 32
The Tx event FIFO elements are indexed from 0 to EFS-1.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **EFSA[13:0]**: Event FIFO start address
Start address of Tx event FIFO in message RAM (32-bit word address, see *Figure 734: Message RAM configuration*).
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

55.4.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	EFPI[4:0]				
						r	r				r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EFGI[4:0]					Res.	Res.	EFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element Lost.
This bit is a copy of interrupt flag FDCAN_IR.TEFL. When FDCAN_IR.TEFL is reset, this bit is also reset.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero.

Bit 24 **EFF**: Event FIFO Full.
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **EFPI[4:0]**: Event FIFO put index.
Tx event FIFO write index pointer, range 0 to 31.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **EFGI[4:0]**: Event FIFO get index.
Tx event FIFO read index pointer, range 0 to 31.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **EFFL[5:0]**: Event FIFO fill level.
Number of elements stored in Tx event FIFO, range 0 to 31.

55.4.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **EFAI[4:0]**: Event FIFO acknowledge index.
After the Host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI. This will set the Tx event FIFO get index FDCAN_TXEFS.EFGI to EFAI + 1 and update the FIFO 0 fill level FDCAN_TXEFS.EFFL.

55.4.47 FDCAN TT trigger memory configuration register (FDCAN_TTTMC)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TME[6:0]**: Trigger memory elements

0: No trigger memory

1-64: Number of trigger memory elements

>64: Values greater than 64 are interpreted as 64

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TMSA[13:0]**: Trigger memory start address.

Start address of trigger memory in message RAM (32-bit word address, see *Figure 734: Message RAM configuration*).

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

55.4.48 FDCAN TT reference message configuration register (FDCAN_TTRMC)

Address offset: 0x0104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMPS	XTD	Res.	RID[28:16]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RMPS**: Reference message Payload select

Ignored in case of time slaves.

0: Reference message has no additional payload

1: The following elements are taken from Tx buffer 0:

Message marker MM,
Event FIFO control EFC,
Data length code DLC,
Data Bytes DB (level 1: bytes 2-8, level 0, 2: bytes 5-8)
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 30 **XTD**: Extended identifier

0: 11-bit standard identifier

1: 29-bit extended identifier

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 29 Reserved, must be kept at reset value.

Bits 28:0 **RID[28:0]**: Reference identifier.

Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

55.4.49 FDCAN TT operation configuration register (FDCAN_TTOCF)

Address offset: 0x0108

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EECS	IRTO[6:0]							LDSDL[2:0]			TM	GEN	Res.	OM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **EVTP**: Event trigger polarity.

0: Rising edge trigger

1: Falling edge trigger

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 25 **ECC**: Enable clock calibration.

0: Automatic clock calibration in FDCAN level 0, 2 is disabled

1: Automatic clock calibration in FDCAN level 0, 2 is enabled

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 24 **EGTF**: Enable global time Filtering.

0: Global time filtering in FDCAN level 0, 2 is disabled

1: Global time filtering in FDCAN level 0, 2 is enabled

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:16 **AWL[7:0]**: Application watchdog limit.

The application watchdog can be disabled by programming AWL to 0x00.

0x00–FF: Maximum time after which the application has to serve the application watchdog. The application watchdog is incremented once each 256 NTUs.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 15 **EECS**: Enable external clock synchronization
If enabled, TUR configuration (FDCAN_TURCF.NCL only) may be updated during FDCAN operation.
0: External clock synchronization in FDCAN level 0, 2 disabled
1: External clock synchronization in FDCAN level 0, 2 enabled
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 14:8 **IRTO[6:0]**: Initial reference trigger offset.
0x00–7F Positive offset, range from 0 to 127
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:5 **LDSDL[2:0]**: LD of synchronization deviation limit.
The synchronization deviation limit SDL is configured by its dual logarithm LDSDL with SDL = 2 * (LDSDL + 5). SDL is comprised between 32 and 4096. It should not exceed the clock tolerance given by the CAN bit timing configuration.
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 4 **TM**: Time master.
0: Time master function disabled
1: Potential time master
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 3 **GEN**: Gap enable.
0: Strictly time-triggered operation
1: External event-synchronized time-triggered operation
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **OM[1:0]**: Operation mode.
00: Event-driven CAN communication, default
01: TTCAN level 1
10: TTCAN level 2
11: TTCAN level 0
These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

55.4.50 FDCAN TT matrix limits register (FDCAN_TTMLM)

Address offset: 0x010C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ENTT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TXEW[3:0]				CSS[1:0]		CCM[5:0]					
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ENTT[11:0]**: Expected Number of Tx triggers

0x000–FFF Expected number of Tx triggers in one matrix cycle.

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TXEW[3:0]**: Tx enable Window

0x0–F Length of Tx enable window, 1-16 NTU cycles

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:6 **CSS[1:0]**: Cycle start synchronization

Enables sync pulse output .

00: No sync pulse

01: Sync pulse at start of basic cycle

10: Sync pulse at start of matrix cycle

11: Reserved

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 5:0 **CCM[5:0]:** Cycle count Max

0x00: 1 basic cycle per matrix cycle

0x01: 2 basic cycles per matrix cycle

0x03: 4 basic cycles per matrix cycle

0x07: 8 basic cycles per matrix cycle

0x0F: 16 basic cycles per matrix cycle

0x1F: 32 basic cycles per matrix cycle

0x3F: 64 basic cycles per matrix cycle

Others: Reserved

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: ISO 11898-4, Section 5.2.1 requires that only the listed cycle count values are configured. Other values are possible, but may lead to inconsistent matrix cycles.

55.4.51 FDCAN TUR configuration register (FDCAN_TURCF)

The length of the NTU is given by: NTU = CAN clock period x NC/DC.

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC extends from 0x10000 to 0x1FFFF. The value configured by NCL is the initial value for

FDCAN_TURNA.NAV[15:0]. DC is set to 0x1000 by hardware reset and it may not be written to 0x0000.

- Level 1: NC 4 × DC and NTU = CAN bit time
- Levels 0 and 2: NC 8 × DC

The actual value of FDCAN_TUR may be changed by the clock drift compensation function of TTCAN level 0 and level 2 in order to adjust the node local view of the NTU to the time master view of the NTU. DC will not be changed by the automatic drift compensation, FDCAN_TURNA.NAV may be adjusted around NC in the range of the synchronization deviation limit given by FDCAN_TTOCF.LDSDL. NC and DC should be programmed to the largest suitable values in achieve the best computational accuracy for the drift compensation process.

Address offset: 0x0110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ELT	Res.	DC[13:0]													
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NCL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ELT**: Enable local time.

0: Local time is stopped, default

1: Local time is enabled

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: The local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. FDCAN_TURCF.DC is locked when FDCAN_TURCF.ELT = 1. If ELT is written to 0, the readable value will stay at 1 until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked.

Bit 30 Reserved, must be kept at reset value.

Bits 29:16 **DC[13:0]**: Denominator configuration.

0x0000: Illegal value

0x0001 to 3FFF: Denominator configuration

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:0 **NCL[15:0]**: Numerator configuration low.

Write access to the TUR numerator configuration low is only possible during configuration with FDCAN_TURCF.ELT = 0 or if FDCAN_TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT configuration mode, the new value takes effect when FDCAN_TTOST.WECS is cleared to 0. NCL is locked FDCAN_TTOST.WECS is 1.

0x0000–FFFF Numerator configuration low

These are protected write bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: If NC < 7 × DC in TTCAN level 1, it is required that subsequent time marks in the trigger memory must differ by at least two NTUs.

55.4.52 FDCAN TT operation control register (FDCAN_TTOCN)

Address offset: 0x0114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SGT
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **LCKC**: TT operation control register Locked.
Set by a write access to register FDCAN_TTOCN. Reset when the updated configuration has been
synchronized into the CAN clock domain.
0: Write access to FDCAN_TTOCN enabled
1: Write access to FDCAN_TTOCN locked

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ESCN**: External synchronization control
If enabled the FDCAN synchronizes its cycle time phase to an external event signaled by a rising edge at event trigger pin (see *Section 55.3.16: Synchronization to external time schedule*).
0: External synchronization disabled
1: External synchronization enabled

Bit 12 **NIG**: Next is Gap.
This bit can only be set when the FDCAN is the actual time master and when it is configured for external event-synchronized time-triggered operation (FDCAN_TTOCF.GEN = 1)
0: No action, reset by reception of any reference message
1: Transmit next reference message with Next_is_Gap = 1

Bit 11 **TMG**: Time mark Gap.
0: Reset by each reference message
1: Next reference message started when register time mark interrupt FDCAN_TTIR.RTMI is activated

Bit 10 **FGP**: Finish Gap.
Set by the CPU, reset by each reference message
0: No reference message requested
1: Application requested start of reference message

Bit 9 **GCS**: Gap control select
0: Gap control independent from event trigger
1: Gap control by input event trigger pin

Bit 8 **TTIE**: Trigger time mark interrupt pulse enable
External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Schedule or In_Gap.
0: Trigger time mark interrupt output m_ttcan_tmp disabled
1: Trigger time mark interrupt output m_ttcan_tmp enabled

Bits 7:6 **TMC[1:0]**: Register time mark Compare.
00: No Register time mark interrupt generated
01: Register time mark interrupt if time mark = cycle time
10: Register time mark interrupt if time mark = local time
11: Register time mark interrupt if time mark = global time
Note: When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = 00, then reconfigure FDCAN_TTTMK, and finally set FDCAN_TMC to the intended time reference.

Bit 5 **RTIE**: Register time mark interrupt pulse enable.
Register time mark interrupts are configured by register FDCAN_TTTMK. A register time mark interrupt pulse with the length of one m_ttcan_clk period is generated when time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) equals FDCAN_TTTMK.TM, independent of the synchronization state.
0: Register time mark interrupt output disabled
1: Register time mark interrupt output enabled

Bits 4:3 **SWS[1:0]**: Stop watch source.
00: Stop watch disabled
01: Actual value of cycle time is copied to FDCAN_TTCPT.SWV
10: Actual value of local time is copied to FDCAN_TTCPT.SWV
11: Actual value of global time is copied to FDCAN_TTCPT.SWV

Bit 2 **SWP**: Stop watch polarity.
0: Rising edge trigger
1: Falling edge trigger

Bit 1 **ECS**: External clock synchronization.
Writing a 1 to ECS sets FDCAN_TTOST.WECS if the node is the actual time master. ECS is reset after one APB clock period. The external clock synchronization takes effect at the start of the next basic cycle.

Bit 0 **SGT**: Set global time.
Writing a 1 to SGT sets FDCAN_TTOST.WGDT if the node is the actual time master. SGT is reset after one APB clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to FDCAN_TTGTP.

55.4.53 FDCAN TT global time preset register (FDCAN_TTGTP)

If TTOST.WGDT is set, the next reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = 1, presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a reference message with Disc_Bit = 1 becomes valid or if the node is not the current time master. TP is locked while FDCAN_TTOST.WGTD = 1 after setting FDCAN_TTOCN.SGT until the reference message with Disc_Bit = 1 becomes valid or until the node is no longer the current time master.

Address offset: 0x0118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CTP[15:0]**: Cycle time target phase
CTP is write-protected while FDCAN_TTOCN.ESCN or FDCAN_TTOST.SPL are set (see *Section 55.3.16: Synchronization to external time schedule*).
0x0000–FFFF Defines the target value of cycle time when a rising edge of event trigger is expected

Bits 15:0 **TP[15:0]**: Time Preset.
TP is write-protected while FDCAN_TTOST.WGTD is set.
0x0000–7FFF next master reference mark = master reference mark + TP
0x8000 reserved
0x8001–FFFF Next master reference mark = master reference mark - (0x10000 - TP).

55.4.54 FDCAN TT time mark register (FDCAN_TTTMK)

A time mark interrupt (FDCAN_TTIR.TMI = 1) is generated when the time base indicated by FDCAN_TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

Address offset: 0x011C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]						
r									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LCKM**: TT time mark register Locked.
Always set by a write access to registers FDCAN_TTOCN. Set by write access to register FDCAN_TTTMK when FDCAN_TTOCN.TMC 00. Reset when the registers have been synchronized into the CAN clock domain.
0: Write access to FDCAN_TTTMK enabled
1: Write access to FDCAN_TTTMK locked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **TICC[6:0]**: Time mark cycle code.
Cycle count for which the time mark is valid.
0b000000x valid for all cycles
0b000001c valid every second cycle at cycle count mod2 = c
0b00001cc valid every fourth cycle at cycle count mod4 = cc
0b0001ccc valid every eighth cycle at cycle count mod8 = ccc
0b001cccc valid every sixteenth cycle at cycle count mod16 = cccc
0b01ccccc valid every thirty-second cycle at cycle count mod32 = ccccc
0b1cccccc valid every sixty-fourth cycle at cycle count mod64 = cccccc

Bits 15:0 **TM[15:0]**: Time mark.
0x0000–FFFF time mark

Note: When using byte access to register FDCAN_TTTMK it is recommended to first disable the time mark compare function (FDCAN_TTOCN.TMC = 00) to avoid comparisons on inconsistent register values.

55.4.55 FDCAN TT interrupt register (FDCAN_TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

Address offset: 0x0120

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTG	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CER**: Configuration error.
Trigger out of order.
0: No error found in trigger list
1: Error found in trigger list

Bit 17 **AW**: Application watchdog.
0: Application watchdog served in time
1: Application watchdog not served in time

Bit 16 **WT**: Watch trigger.
0: No missing reference message
1: Missing reference message (level 0: cycle time 0xFF00)

Bit 15 **IWTG**: Initialization watch trigger.
The initialization is restarted by resetting IWT.
0 No missing reference message during system startup
1 No system startup due to missing reference message

Bit 14 **ELC**: Error level Changed.
Not set when error level changed during initialization.
0: No change in error level
1: Error level changed

Bit 13 **SE2**: Scheduling error 2.
0: No scheduling error 2
1: Scheduling error 2 occurred

Bit 12 **SE1**: Scheduling error 1.
0: No scheduling error 1
1: Scheduling error 1 occurred

Bit 11 **TXO**: Tx count overflow.
0: Number of Tx trigger as expected
1: More Tx trigger than expected in one cycle

Bit 10 **TXU**: Tx count underflow.
0: Number of Tx trigger as expected
1: Less Tx trigger than expected in one cycle

Bit 9 **GTE**: Global time error.
Synchronization deviation SD exceeds limit specified by FDCAN_TTOCF.LDSDL, TTCAN level 0, 2 only.
0: Synchronization deviation within limit
1: Synchronization deviation exceeded limit

Bit 8 **GTD**: Global time discontinuity.
0: No discontinuity of global time
1: Discontinuity of global time

Bit 7 **GTW**: Global time wrap
0: No global time wrap occurred
1: Global time wrap from 0xFFFF to 0x0000 occurred

Bit 6 **SWE**: Stop watch event
0: No rising/falling edge at stop watch trigger pin detected
1 :Rising/falling edge at stop watch trigger pin detected

Bit 5 **TTMI:** Trigger time mark event internal
Internal time mark events are configured by trigger memory element TMIN (see *Section 55.3.22: FDCAN trigger memory element*). Set when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Gap or In_Schedule.
0: Time mark not reached
1: Time mark reached (level 0: cycle time FDCAN_TTOCF.RTO x 0x200)

Bit 4 **RTMI**: Register time mark interrupt.
Set when time referenced by TTOCN.TMC (cycle, local, or global) equals FDCAN_TTTMK.TM, independently from the synchronization state.
0: Time mark not reached
1: Time mark reached

Bit 3 **SOG**: Start of Gap
0 No reference message seen with Next_is_Gap bit set
1 Reference message with Next_is_Gap bit set becomes valid

Bit 2 **CSM**: Change of synchronization mode.
0: No change in master to slave relation or schedule synchronization
1: Master to slave relation or schedule synchronization changed,
also set when FDCAN_TTOST.SPL is reset

Bit 1 **SMC**: Start of matrix cycle.
0: No matrix cycle started since bit has been reset
1: Matrix cycle started

Bit 0 **SBC**: Start of basic cycle.
0: No basic cycle started since bit has been reset
1: Basic cycle started

55.4.56 FDCAN TT interrupt enable register (FDCAN_TTIE)

The settings in the TT interrupt enable register determine which status changes in the TT interrupt register will result in an interrupt.

Address offset: 0x0124

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERE**: Configuration error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 17 **AWE**: Application watchdog interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 16 **WTE**: Watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 15 **IWTE**: Initialization watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 14 **ELCE**: Change error level interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 13 **SE2E**: Scheduling error 2 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 12 **SE1E**: Scheduling error 1 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 11 **TXOE**: Tx count overflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 10 **TXUE**: Tx count underflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 9 **GTEE**: Global time error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 8 **GTDE**: Global time discontinuity interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 7 **GTWE**: Global time wrap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 6 **SWEE**: Stop watch event interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 5 **TTMIE:** Trigger time mark event internal interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 4 **RTMIE**: Register time mark interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 3 **SOGE**: Start of gap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 2 **CSME**: Change of synchronization mode interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 1 **SMCE**: Start of matrix cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 0 **SBCE**: Start of basic cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

55.4.57 FDCAN TT interrupt line select register (FDCAN_TTILS)

The TT interrupt Line select register assigns an interrupt generated by a specific interrupt flag from the TT interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0128

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERL**: Configuration error interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 17 **AWL**: Application watchdog interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 16 **WTL:** Watch trigger interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 15 **IWTL:** Initialization watch trigger interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 14 **ELCL:** Change error level interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 13 **SE2L:** Scheduling error 2 interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 12 **SE1L:** Scheduling error 1 interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 11 **TXOL:** Tx count overflow interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 10 **TXUL:** Tx count underflow interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 9 **GTEL:** Global time error interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 8 **GTDL:** Global time discontinuity interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 7 **GTWL:** Global time wrap interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 6 **SWEL:** Stop watch event interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 5 **TTMIL:** Trigger time mark event internal interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 4 **RTMIL:** Register time mark interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 3 **SOGL:** Start of gap interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 2 **CSML:** Change of synchronization mode interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 1 **SMCL:** Start of matrix cycle interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 0 **SBCL:** Start of basic cycle interrupt Line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

55.4.58 FDCAN TT operation status register (FDCAN_TTOST)

Address offset: 0x012C

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPL	WECS	AWE	WFE	GSI	TMP[2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[7:0]								QCS	QGTP	SYS[1:0]		MS[1:0]		EL[1:0]	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **SPL**: Schedule phase lock.

The bit is valid only when external synchronization is enabled (FDCAN_TTOCN.ESCN = 1). In this case it signals that the difference between cycle time configured by FDCAN_TTGTP.CTP and the cycle time at the rising edge at event trigger pin is less or equal 9 NTU (see *Section 55.3.16: Synchronization to external time schedule*).

0: Phase outside range

1: Phase inside range

Bit 30 **WECS**: Wait for external clock synchronization.

0: No external clock synchronization pending

1: Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.

Bit 29 **AWE**: Application watchdog event.

The application watchdog is served by reading FDCAN_TTOST. When the watchdog is not served in time, bit AWE is set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode.

0: Application watchdog served in time

1: Failed to serve application watchdog in time

Bit 28 **WFE**: Wait for event.

0: No Gap announced, reset by a reference message with Next_is_Gap = 0

1: Reference message with Next_is_Gap = 1 received

Bit 27 **GSI**: Gap started Indicator.

0: No Gap in schedule, reset by each reference message and for all time slaves

1: Gap time after basic cycle has started

Bits 26:24 **TMP[2:0]**: Time master priority.

0x0-7 Priority of actual time master

Bit 23 **GFI**: Gap finished Indicator.

Set when the CPU writes FDCAN_TTOCN.FGP, or by a time mark interrupt if TMG = 1, or via input pin (event trigger) if FDCAN_TTOCN.GCS = 1. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message.

0: Reset at the end of each reference message

1: Gap finished by FDCAN

Bit 22 **WGTD**: Wait for global time discontinuity.

0: No global time preset pending

1: Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = 1 or after it received a reference message.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:8 **RTO[7:0]**: Reference trigger offset.
The reference trigger offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of-127 is reached. In case the FDCAN becomes time master (MS[1:0] = 11), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by FDCAN_TTOCF.IRTO is read.
0x00-FF Actual reference trigger offset value

Bit 7 **QCS**: Quality of clock Speed.
Only relevant in TTCAN level 0 and level 2, otherwise fixed to 1.
0: Local clock speed not synchronized to time master clock speed
1: Synchronization deviation ≤ SDL

Bit 6 **QGTP**: Quality of global time phase.
Only relevant in TTCAN level 0 and level 2, otherwise fixed to 0.
0: Global time not valid
1: Global time in phase with time master

Bits 5:4 **SYS[1:0]**: Synchronization state
00: Out of Synchronization
01: Synchronizing to FDCAN communication
10: Schedule suspended by Gap (In_Gap)
11: Synchronized to schedule (In_Schedule)

Bits 3:2 **MS[1:0]**: Master state.
00: Master_Off, no master properties relevant
01: Operating as time Slave
10: Operating as backup time master
11: Operating as current time master

Bits 1:0 **EL[1:0]**: Error level.
00: Severity 0 - No error
01: Severity 1 - Warning
10: Severity 2 - error
11: Severity 3 - Severe error

55.4.59 FDCAN TUR numerator actual register (FDCAN_TURNA)

There is no drift compensation in TTCAN level 1 (NAV = NC). In TTCAN level 0 and level 2, the drift between the node local clock and the time master local clock is calculated. The drift is compensated when the synchronization deviation (difference between NC and the calculated NAV) is lower than 2 * (FDCAN_TTOCF.LDSDL + 5). With FDCAN_TTOCF.LDSDL < 7, this results in a maximum range for NAV of (NC - 0x1000) ≤ NAV ≤ (NC + 0x1000).

Address offset: 0x0130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:16]	
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **NAV[17:0]**: Numerator actual value.
0x0EFFF Illegal value
0x0F000–20FFF Actual numerator value
0x21000 Illegal value

55.4.60 FDCAN TT local and global time register (FDCAN_TTLGT)

Address offset: 0x0134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **GT[15:0]**: Global time.
Non-fractional part of the sum of the node local time and its local offset (see *Section 55.3.11: Local time, cycle time, global time, and external clock synchronization*).
0x0000–FFFF Global time value of FDCAN network

Bits 15:0 **LT[15:0]**: Local time.
Non-fractional part of local time, incremented once each local NTU (see *Section 55.3.11: Local time, cycle time, global time, and external clock synchronization*).
0x0000–FFFF Local time value of FDCAN node

55.4.61 FDCAN TT cycle time and count register (FDCAN_TTCTC)

Address offset: 0x0138

Reset value: 0x003F 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CC[5:0]**: Cycle count.
0x00–3F Number of actual basic cycle in the system matrix

Bits 15:0 **CT[15:0]**: Cycle time
Non-fractional part of the difference of the node local time and Ref_Mark (see *Section 55.3.11: Local time, cycle time, global time, and external clock synchronization*).
0x0000–FFFF Cycle time value of FDCAN basic cycle

55.4.62 FDCAN TT capture time register (FDCAN_TTCPT)

Address offset: 0x013C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
										r	r	r	r	r	r

Bits 31:16 **SWV[15:0]**: Stop watch value.
On a rising / falling edge (as configured via FDCAN_TTOCN.SWP) at the stop watch trigger pin, when FDCAN_TTOCN.SWS] is different from 00 and FDCAN_TTIR.SWE is 0, the actual time value as selected by FDCAN_TTOCN.SWS (cycle, local, global) is copied to SWV and TFDCAN_TIR.SWE will be set to 1.Capturing of the next stop watch value is enabled by resetting FDCAN_TTIR.SWE.
0x0000–FFFF Captured stop watch value

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CCV[5:0]**: Cycle count value
Cycle count value captured together with SWV.
0x00–3F Captured cycle count value

55.4.63 FDCAN TT cycle sync mark register (FDCAN_TTCSM)

Address offset: 0x0140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSM[15:0]**: Cycle sync mark.
The cycle sync mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.
0x0000–FFFF Captured cycle time

55.4.64 FDCAN TT trigger select register (FDCAN_TTTS)

The settings in the FDCAN_TTTS register select the input to be used as event trigger and stop watch trigger.

Address offset: 0x0300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL[1:0]		Res.	Res.	SWTDEL[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **EVTSEL[1:0]**: Event trigger input selection
These bits are used to select the input to be used as event trigger
00: fdcan1_swt0
01: fdcan1_swt1
10: fdcan1_swt2
11: fdcan1_swt3

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SWTDEL[1:0]**: Stop watch trigger input selection
These bits are used to select the input to be used as stop watch trigger
00: fdcan1_evt0
01: fdcan1_evt1
10: fdcan1_evt2
11: fdcan1_evt3

55.4.65 FDCAN register map and reset value table

Table 485. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL [3:0]				STEP [3:0]				SUBSTEP [3:0]				YEAR [3:0]				MON [7:0]								DAY [7:0]							
	Reset value	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	CAT	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN _CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBRP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030 to 0x003F	Reserved																																
	Reset value																																
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x0050	FDCAN_IR	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	FDCAN_ILS	Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060 to 0x007F	Reserved																																
	Reset value																																
0x0080	FDCAN_GFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	FDCAN _SIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]								FLSSA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0088	FDCAN _XIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[6:0]							FLESA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0090	FDCAN _XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0094	FDCAN _HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	FIDX[6:0]							MSI[1:0]	BIDX[5:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0098	FDCAN _NDAT1	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x009C	FDCAN_NDAT2	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	FDCAN_RXF0C	F0OM	F0WM[6:0]							Res.	F0S[6:0]							F0SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A4	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0	F0F	Res.	Res.	F0PI[5:0]						Res.	Res.	F0GI[5:0]						Res.	F0FL[6:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A8	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0FL[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00AC	FDCAN_RXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBSA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B0	FDCAN_RXF1C	F1OM	F1WM[6:0]							Res.	F1S[6:0]							F1SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B4	FDCAN_RXF1S	DMS[1:0]		Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res	F1PI[5:0]					Res.	Res.	Re	F1GI[5:0]						Res.	Re	F1FL[6:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B8	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1FL[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00BC	FDCAN_RXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBDS[2:0]				F1DS[2:0]			Res.	F0DS[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C0	FDCAN_TXBC	Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]						TBSA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]					Res.	Res.	Res.	TFGI[4:0]						Res.	Res.	TFFL[5:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8	FDCAN_TXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0DS[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00CC	FDCAN_TXBRP	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00D0	FDCAN_TXBAR	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D4	FDCAN_TXBCR	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	FDCAN_TXBTO	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	FDCAN_TXBCF	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E0	FDCAN_TXBTIE	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E4	FDCAN_TXBCIE	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00F0	FDCAN_TXEFC	Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]						EFSA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00F4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEF	EFF	Res.	Res.	Res.	EFPI[4:0]					Res.	Res.	Res.	EFG[4:0]					Res.	Res.	EFFL[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00F8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0100	FDCAN_TTTMC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]							TMSA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0104	FDCAN_TTRMC	RMPS	XTD	Res.	RID[28:0]																												
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0108	FDCAN_TTOCF	Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]								EECS	IRTO[6:0]							LDSDL [2:0]			TM	GEN	Res.	OM [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010C	FDCAN_TTMLM	Res.	Res.	Res.	Res.	ENTT[11:0]												Res.	Res.	Res.	Res.	TXEW[3:0]				CSS[1:0]		CCM[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0110	FDCAN_TURCF	ELT	Res.	DC[13:0]														NCL[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0114	FDCAN_TTOCN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SBC
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0118	FDCAN_TTGTP	CTP[15:0]																TP[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x011C	FDCAN_TTTMK	LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]							TM[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0120	FDCAN_TTIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT	IWT	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0124	FDCAN_TTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0128	FDCAN_TTILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x012C	FDCAN_TTOST	SPL	WECS	AWE	WFE	GSI	TMP[2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.	RTO[7:0]								QCS	QGTP	SYS[1:0]		MS[1:0]		EL[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x0130	FDCAN_TURNA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:0]																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134	FDCAN_TTLGT	GT[15:0]																LT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0138	FDCAN_TTCTC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]						CT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x013C	FDCAN_TTCPT	SWV[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0140	FDCAN_TTCSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSM[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0144 to 0x01FC	Reserved																																
	Reset value																																

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0300	FDCAN _TTTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL[1:0]		Res.	Res.	SWTSEL[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

55.5 CCU registers

55.5.1 Clock calibration unit core release register (FCCAN_CCU_CREL)

Address offset: 0x0000

Reset value: 0x1114 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 1

Bits 27:24 **STEP[3:0]**: Step of core release = 1

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Time stamp year =

Bits 15:8 **MON[7:0]**: Time stamp month = 12

Bits 7:0 **DAY[7:0]**: Time stamp day = 18

55.5.2 Calibration configuration register (FCCAN_CCU_CCFG)

Address offset: 0x0004

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]			
rw												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 **SWR**: Software Reset

Writing a 1 to this bit will reset the calibration FSM to state Not_Calibrated (CCU_CSTAT.CALS = 00). The calibration watchdog value CWD.WDV is also reset. Registers FDCAN_CCFG, CCU_CSTAT and the calibration watchdog configuration CWD.WDC are unchanged. The bit remains set until reset is completed.

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bits 30:20 Reserved, must be kept at reset value.

Bits 19:16 **CDIV[3:0]**: Clock Divider

The clock divider has to be configured when the clock calibration is bypassed (BCC = 1) to ensure that the FDCAN requirement is fulfilled.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bits 15:8 **OCPM[7:0]**: Oscillator clock periods minimum

Configures the minimum number of periods in two CAN bit times. OCPM is used in basic calibration to avoid false measurements in case of glitches on the bus line. The configured number of periods is OCPM × 32. The configuration depends on the frequency (from 80 to 100 MHz) and the bitrate configured in FDCAN1 and FDCAN2 (from 125 kbit/s up to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz fdcan_ker_ck and 1 Mbit/s CAN bitrate.

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 7 **CFL**: Calibration Field length

0: Calibration field length is 32 bits

1: Calibration field length is 64 bits

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 6 **BCC**: Bypass clock calibration

If this bit is set, the clock input fdcan_ker_ck is routed to the time quanta clock through a clock divider configurable via CDIV, cu_cok is always 1. In this case the baudrate prescaler of the connected FDCANs has to be configured to generate the FDCAN internal time quanta clock.

0: Clock calibration unit generates time quanta clock

1: Clock calibration unit bypassed (default configuration)

Note: As long as fdcan_ker_ck is equal or above 80 MHz the clock calibration on CAN unit is functional, even when BCC = 1. The calibration state can be read from register CCU_CSTAT.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **TQBT[4:0]**: Time quanta per bit time

Configures the number of time quanta per bit time. Same value as configured in FDCAN1 and FDCAN2. The range of the resulting time quanta clock fdcan_tq_ck is from 0.5 MHz (bitrate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bitrate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25. Configured values below 4 are interpreted as 4, values above 25 are interpreted as 25.

Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

55.5.3 Calibration status register (FCCAN_CCU_CSTAT)

Address offset: 0x0008

Reset value: 0x0203 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALS[1:0]		Res.	TQC[10:0]											OCPC[17:16]	
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 **CALS[1:0]**: Calibration state

00: Not_Calibrated

01: Basic_Calibrated

10: Precision_Calibrated

11: Reserved

Bit 29 Reserved, must be kept at reset value.

Bits 28:18 **TQC[10:0]**: Time quanta counter

Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

Bits 17:0 **OCPC[17:0]**: Oscillator clock period counter

Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

55.5.4 Calibration watchdog register (FCCAN_CCU_CWD)

Address offset: 0x000C

Reset value: 0x0000 0000

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (CCU_CSTAT.CALS = 00). In this state the calibration watchdog monitors the message received. In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM stays in state Not_Calibrated (CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (CCU_CSTAT.CALS = 01), the calibration watchdog is restarted with each received message . In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM returns to state Not_Calibrated (CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message is received, state Precision_Calibrated (CCU_CSTAT.CALS = 10) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the quartz message received input. In case no message from a quartz controlled node is received by the attached TTCAN until the calibration watchdog has counted down to 0, the calibration FSM transits back to state Basic_Calibrated (CCU_CSTAT.CALS = 01). The signal is active when the CAN protocol engine on the attached TTCAN is started i.e. when the INIT bit is reset.

A calibration watchdog event also sets interrupt flag CCU_IR.CWE. If enabled by CCU_IE.CWEE, interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CCU_IR.CWE is reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WDV[15:0]**: Watchdog value
Actual calibration watchdog counter value.

Bits 15:0 **WDC[15:0]**: WDC
Watchdog configuration
Start value of the calibration watchdog counter. With the reset value of 00 the counter is disabled.
Write access by the Host CPU to registers/bits marked with “P = Protected Write” is possible only when the FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

55.5.5 Clock calibration unit interrupt register (FCCAN_CCU_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. The configuration of CCU_IE controls whether an interrupt is generated or not.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSC**: Calibration state changed
0: Calibration state unchanged
1: Calibration state has changed

Bit 0 **CWE**: Calibration watchdog event
0: No calibration watchdog event
1: Calibration watchdog event occurred

55.5.6 Clock calibration unit interrupt enable register (FCCAN_CCU_IE)

Address offset: 0x0014

Reset value: 0x0000 0000

The settings in the CU interrupt enable register determine whether a status change in the CU interrupt register will be signaled on an interrupt line.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSCE**: Calibration state changed enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **CWEE**: Calibration watchdog event enable
0: Interrupt disabled
1: Interrupt enabled

55.5.7 CCU register map and reset value table

Table 486. CCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FCCAN_ CCU_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FCCAN_ CCU_CCFG	SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]				CSM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	FCCAN_ CCU_CSTAT	CALS [1:0]		Res.	TQC[10:0]													OCPC[17:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	FCCAN_ CCU_CWD	WDV[15:0]																WDC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0010	FCCAN_ CCU_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	FCCAN_ CCU_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

56 USB on-the-go high-speed (OTG_HS)

56.1 Introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

This section presents the architecture and the programming model of the OTG_HS controller.

The following acronyms are used throughout the section:

FS	Full-speed
LS	Low-speed
HS	High-speed
MAC	Media access controller
OTG	On-the-go
PFC	Packet FIFO controller
PHY	Physical layer
USB	Universal serial bus
UTMI	USB 2.0 Transceiver Macrocell interface (UTMI)
ULPI	UTMI+ Low Pin Interface
LPM	Link power management
BCD	Battery charging detector
HNP	Host negotiation protocol
SRP	Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or device-only controller, fully compliant with the *USB 2.0 Specification.* OTG_HS supports the speeds defined in the *Table 487: OTG_HS speeds supported* below.The USB OTG supports both HNP and SRP. The only external device required is a charge pump for V_{BUS} in OTG mode.

Table 487. OTG_HS speeds supported

-	HS (480 Mb/s)	FS (12 Mb/s)	LS (1.5 Mb/s)
Host mode	X	X	X
Device mode	X	X	-

56.2 OTG main features

The main features can be divided into three categories: general, host-mode and device-mode features.

56.2.1 General features

The OTG_HS interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0
- OTG_HS supports the following PHY interfaces:
  - An on-chip full-speed PHY
  - An ULPI interface for external high-speed PHY
- It includes full support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - Integrated support for host Negotiation protocol (HNP) and session request protocol (SRP)
  - It allows host to turn V_{BUS} off to conserve battery power in OTG applications
  - It supports OTG monitoring of V_{BUS} levels with internal comparators
  - It supports dynamic host-peripheral switch of role
- It is software-configurable to operate as:
  - SRP capable USB HS Peripheral (B-device)
  - SRP capable USB HS/LS host (A-device)
  - USB On-The-Go Full-Speed Dual Role device
- It supports HS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- OTG_HS embeds an internal DMA with shareholding support and software selectable AHB burst type in DMA mode.
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 4 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2 on the FS PHY transceiver only.

56.2.2 Host-mode features

The OTG_HS interface main features and requirements in host-mode are the following:

- External charge pump for V_{BUS} voltage generation.
- Up to 16 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 16 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 16 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

56.2.3 Peripheral-mode features

The OTG_HS interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 8 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 8 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 9 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.

56.3 OTG implementation

Table 488. OTG implementation(1)

USB features	OTG_HS1(2)	OTG_HS2(3)
Device bidirectional endpoints (including EP0)	9	
Host mode channels	16	
Size of dedicated SRAM	4 KB	
USB 2.0 link power management (LPM) support	X	
OTG revision supported	2.0	
Attach detection protocol (ADP) support	-	
Battery charging detection (BCD) support	X	
ULPI available to primary IOs via, muxing	X	-
Integrated PHY	FS	FS

1. “X” = supported, “-” = not supported.
2. Compatible with high speed operation.
3. Incompatible with high speed operation.

56.4 OTG functional description

56.4.1 OTG block diagram

In STM32H7, two instances of OTG_HS are present (OTG_HS1 and OTG_HS2).

Only OTG_HS1 has an accessible ULPI interface which will allow high speed operation using an external HS transceiver.

Figure 746. USB1 OTG high-speed block diagram (OTG_HS1)

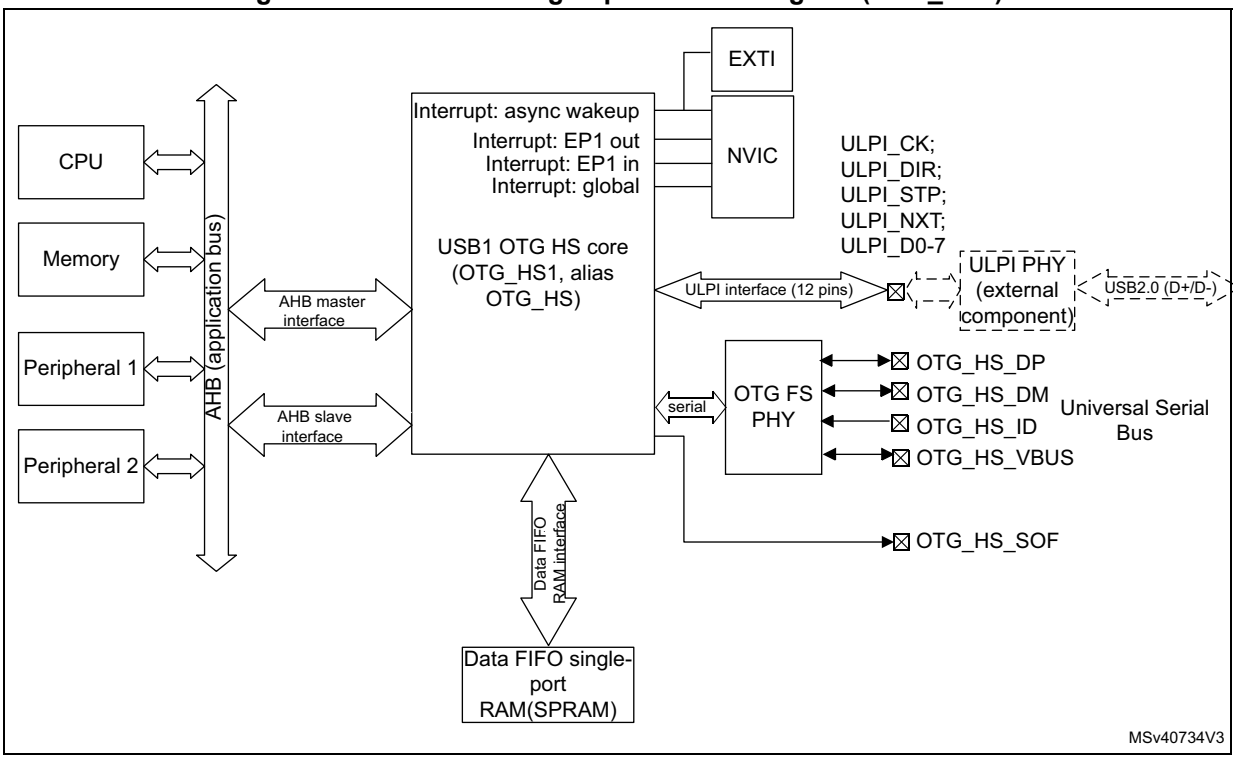
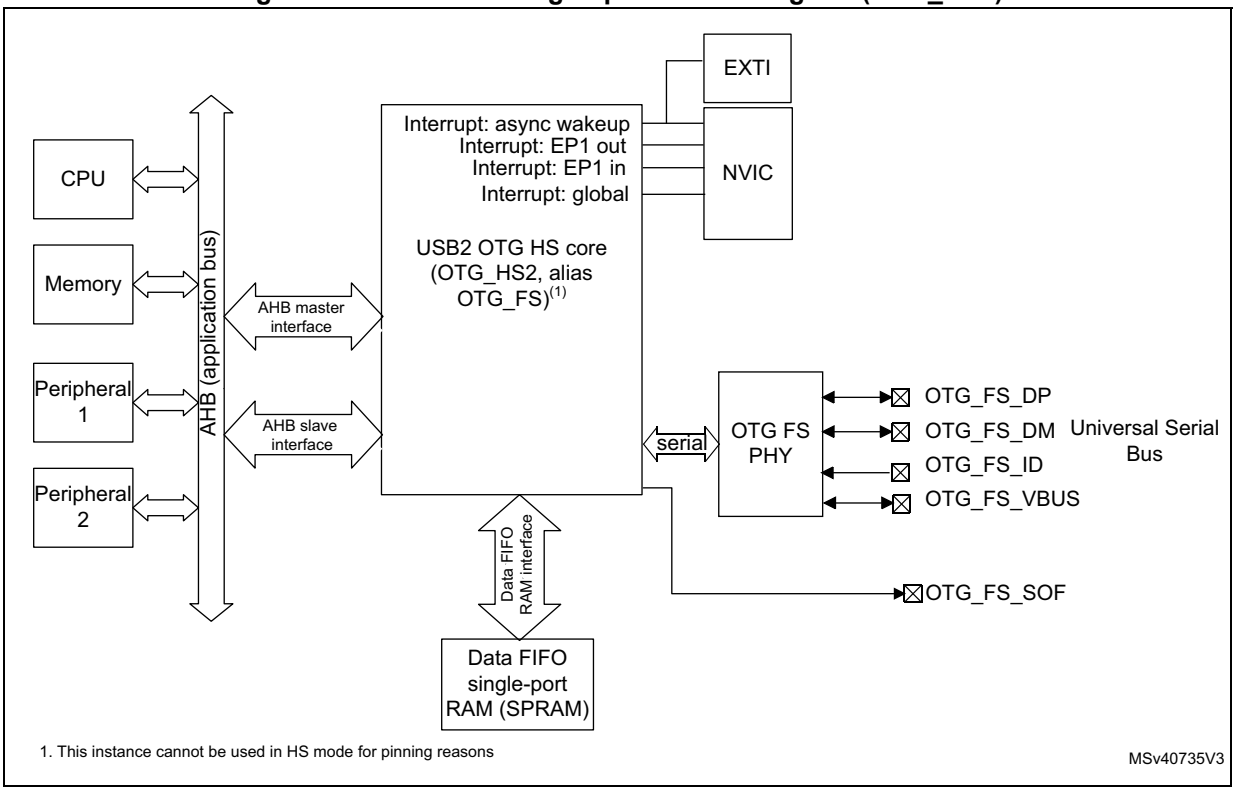


Figure 747. USB2 OTG high-speed block diagram (OTG_HS2)



56.4.2 USB OTG pin and internal signals

Table 489. OTG_FS input/output pins

Signal name	Signal type	Description
OTG_FS_DP	Digital input/output	USB OTG D+ line
OTG_FS_DM	Digital input/output	USB OTG D- line
OTG_FS_ID	Digital input	USB OTG ID
OTG_FS_VBUS	Analog input	USB OTG VBUS
OTG_FS_SOF	Digital output	USB OTG Start Of Frame (visibility)

Table 490. OTG_HS input/output pins

Signal name	Signal type	Description
OTG_HS_DP	Digital input/output	USB OTG D+ line
OTG_HS_DM	Digital input/output	USB OTG D- line
OTG_HS_ID	Digital input	USB OTG ID
OTG_HS_VBUS	Analog input	USB OTG VBUS
OTG_HS_SOF	Digital output	USB OTG Start Of Frame (visibility)
OTG_HS_ULPI_CK	Digital input	USB OTG ULPI clock
OTG_HS_ULPI_DIR	Digital input	USB OTG ULPI data bus direction control
OTG_HS_ULPI_STP	Digital output	USB OTG ULPI data stream stop
OTG_HS_ULPI_NXT	Digital input	USB OTG ULPI next data stream request
OTG_HS_ULPI_D[0..7]	Digital input/output	USB OTG ULPI 8-bit bi-directional data bus

Table 491. OTG_HS input/output signals

Signal name	Signal type	Description
usb_sof	Digital output	USB OTG start-of-frame event for on chip peripherals
usb_wkup	Digital output	USB OTG wakeup event output
usb_gbl_it	Digital output	USB OTG global interrupt
usb_ep1_in_it	Digital output	USB OTG endpoint 1 in interrupt
usb_ep1_out_it	Digital output	USB OTG endpoint 1 out interrupt

56.4.3 OTG core

The USB OTG receives the 48 MHz clock from the reset and clock controller (RCC). The USB clock is used for driving the 48 MHz domain at full-speed (12 Mbit/s) and must be enabled prior to configuring the OTG core.

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in *Section 56.12: OTG_HS interrupts*.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 4-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY) or external HS PHY.

56.4.4 Embedded full speed OTG PHY

The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and device. It directly drives transmission and reception on the single-ended USB lines.
- integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG_HS core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as V_{BUS} is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both DP/DM. Pull-up and pull-down resistors are dynamically switched when the peripheral role is changed via the host negotiation protocol (HNP).
- Pull-up/pull-down resistor ECN circuit. The DP pull-up consists of 2 resistors controlled separately from the OTG_HS as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows to achieve a better noise rejection and Tx/Rx signal quality.
- V_{BUS} sensing comparators with hysteresis used to detect V_{BUS} valid, A-B session valid and session-end voltage thresholds. They are used to drive the session request protocol (SRP), detect valid startup and end-of-session conditions, and constantly monitor the V_{BUS} supply during USB operations.

To guarantee a correct operation for the USB OTG_HS peripheral, the AHB frequency should be higher than 30 MHz.

56.4.5 High-speed OTG PHY

The USB OTG_HS core includes an ULPI interface to connect an external HS PHY.

ULPI interface is only available on OTG HS1 (see *Section 56.4.1: OTG block diagram*).

56.5 OTG dual role device (DRD)

56.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this

configuration the OTG_HS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

- If the A-side of the USB cable is connected with a grounded ID, the OTG_HS issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG_HS complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

56.5.2 HNP dual role device

The HNP capable bit in the Global USB configuration register (HNPCAP bit in OTG_ GUSBCFG) enables the OTG_HS core to dynamically change its role from A-host to A-peripheral and vice-versa, or from B-Peripheral to B-host and vice-versa according to the host negotiation protocol (HNP). The current device status can be read by the combined values of the connector ID status bit in the Global OTG control and status register (CIDSTS bit in OTG_GOTGCTL) and the current mode of operation bit in the global interrupt and status register (CMOD bit in OTG_GINTSTS).

The HNP program model is described in detail in *Section 56.15: OTG_HS programming model*.

56.5.3 SRP dual role device

The SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG_HS core to switch off the generation of V_{BUS} for the A-device to save power. Note that the A-device is always in charge of driving V_{BUS} regardless of the host or peripheral role of the OTG_HS.

The SRP A/B-device program model is described in detail in *Section 56.15: OTG_HS programming model*.

56.6 USB peripheral

This section gives the functional description of the OTG_HS in the USB peripheral mode. The OTG_HS works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- OTG A-Peripheral
  - OTG A-device state after the HNP switches the OTG_HS to its peripheral role
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable, and the HNP-capable bit in the Global USB Configuration register (HNPCAP bit in OTG_GUSBCFG) is cleared.
- Peripheral only
  - The force device mode bit (FDMOD) in the *Section 56.14.4: OTG USB configuration register (OTG_GUSBCFG)* is set to 1, forcing the OTG_HS core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from V_{BUS}.

56.6.1 SRP-capable peripheral

The SRP capable bit in the Global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG_HS to support the session request protocol (SRP). In this way, it allows the remote A-device to save power by switching off V_{BUS} while the USB session is suspended.

The SRP peripheral mode program model is described in detail in the *B-device session request protocol* section.

56.6.2 Peripheral states

Powered state

The V_{BUS} input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG_HS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG_GINTSTS) to notify the powered state.

The V_{BUS} input also ensures that valid V_{BUS} levels are supplied by the host during USB operations. If a drop in V_{BUS} below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG_HS automatically disconnects and the session end detected (SEDET bit in OTG_GOTGINT) interrupt is generated to notify that the OTG_HS has exited the powered state.

In the powered state, the OTG_HS expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG_HS enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG_HS expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG_HS then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG_HS peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG_HS enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wakeup signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

56.6.3 Peripheral endpoints

The OTG_HS core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Proper control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt (OTG_DIEPINT0/)OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints
- 8 IN endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has proper control (OTG_DIEPCTL*x*), transfer configuration (OTG_DIEPTSIZ*x*), and status-interrupt (OTG_DIEPINT*x*) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).
- 8 OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a proper control (OTG_DOEPCTL*x*), transfer configuration (OTG_DOEPTSIZ*x*) and status-interrupt (OTG_DOEPINT*x*) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

Endpoint control

- The following endpoint controls are available to the application through the device endpoint-*x* IN/OUT control register (OTG_DIEPCTL*x*/OTG_DOEPCTL*x*):
  - Endpoint enable/disable
  - Endpoint activate in current configuration
  - Program USB transfer type (isochronous, bulk, interrupt)
  - Program supported packet size
  - Program Tx FIFO number associated with the IN endpoint
  - Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - Optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - Optionally program the STALL bit to always stall host tokens to that endpoint
  - Optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

Endpoint transfer

The device endpoint-*x* transfer size registers (OTG_DIEPTSIZ*x*/OTG_DOEPTSIZ*x*) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG_HS core updates them with the current transfer status.

The following transfer parameters can be programmed:

- Transfer size in bytes
- Number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-*x* interrupt registers (OTG_DIEPINT*x*/OTG_DOPEPINT*x)* indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-*x* interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers

The peripheral core provides the following status checks and interrupt generation:

- Transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- Setup stage has been done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- Babble error condition has been detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets were received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet has been dropped, without generating an interrupt

56.7 USB host

This section gives the functional description of the OTG_HS in the USB host mode. The OTG_HS works as a USB host in the following circumstances:

- OTG A-host
  - OTG A-device default state when the A-side of the USB cable is plugged in
- OTG B-host
  - OTG B-device after HNP switching to the host role
- A-device
  - If the ID line is present, functional and connected to the A-side of the USB cable, and the HNP-capable bit is cleared in the Global USB Configuration register (HNPCAP bit in OTG_GUSBCFG). Integrated pull-down resistors are automatically set on the DP/DM lines.
- Host only
  - The force host mode bit (FHMOD) in the *OTG USB configuration register (OTG_GUSBCFG)* forces the OTG_HS core to work as a USB host-only. In this case, the ID line is ignored even if present on the USB connector. Integrated pull-down resistors are automatically set on the DP/DM lines.

Note: On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

56.7.1 SRP-capable host

SRP support is available through the SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG). With the SRP feature enabled, the host can save power by switching off the V_{BUS} power while the USB session is suspended.

The SRP host mode program model is described in detail in the *A-device session request protocol*) section.

56.7.2 USB host states

Host port power

On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output or via an I^2C interface connected to an external PMIC (power management IC). When the application decides to power on V_{BUS}, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

V_{BUS} valid

When HNP or SRP is enabled the VBUS sensing pin should be connected to V_{BUS}. The V_{BUS} input ensures that valid V_{BUS} levels are supplied by the charge pump during USB operations. Any unforeseen V_{BUS} voltage drop below the V_{BUS} valid threshold (4.4 V) leads to an OTG interrupt triggered by the session end detected bit (SEDET bit in OTG_GOTGINT). The application is then required to remove the V_{BUS} power and clear the port power bit.

When HNP and SRP are both disabled, the VBUS sensing pin does not need to be connected to V_{BUS}.

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the V_{BUS} generation and clear the port power bit.

Host detection of a peripheral connection

If SRP or HNP are enabled, even if USB peripherals or B-devices can be attached at any time, the OTG_HS will not detect any bus connection until V_{BUS} is no longer sensed at a valid level (5 V). When V_{BUS} is at a valid level and a remote B-device is attached, the OTG_HS core issues a host port interrupt triggered by the device connected bit in the host port control and status register (PCDET bit in OTG_HPRT).

When HNP and SRP are both disabled, USB peripherals or B-device are detected as soon as they are connected. The OTG_HS core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

Before starting to drive a USB reset, the application waits for the OTG interrupt triggered by the debounce done bit (DBCDNE bit in OTG_GOTGINT), which indicates that the bus is stable again after the electrical debounce caused by the attachment of a pull-up resistor on DP (FS) or DM (LS).

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for

a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

Host suspend

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG_HS core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device's initiative (remote wakeup). In this case the remote wakeup interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wakeup signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

56.7.3 Host channels

The OTG_HS core instantiates 16 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 16 transfer requests at the same time. If more than 16 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of proper control (OTG_HCCHAR*x*), transfer configuration (OTG_HCTSIZ*x*) and status/interrupt (OTG_HCINT*x*) registers with associated mask (OTG_HCINTMSK*x*) registers.

Host channel control

- The following host channel controls are available to the application through the host channel-*x* characteristics register (OTG_HCCHAR*x*):
  - Channel enable/disable
  - Program the HS/FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

Host channel transfer

The host channel transfer size registers (OTG_HCTSIZ*x*) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG_HS core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-*x* interrupt register (OTG_HCINT*x*) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers. The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

56.7.4 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the

requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG_HS core is fully responsible for the management of the periodic and nonperiodic request queues.The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

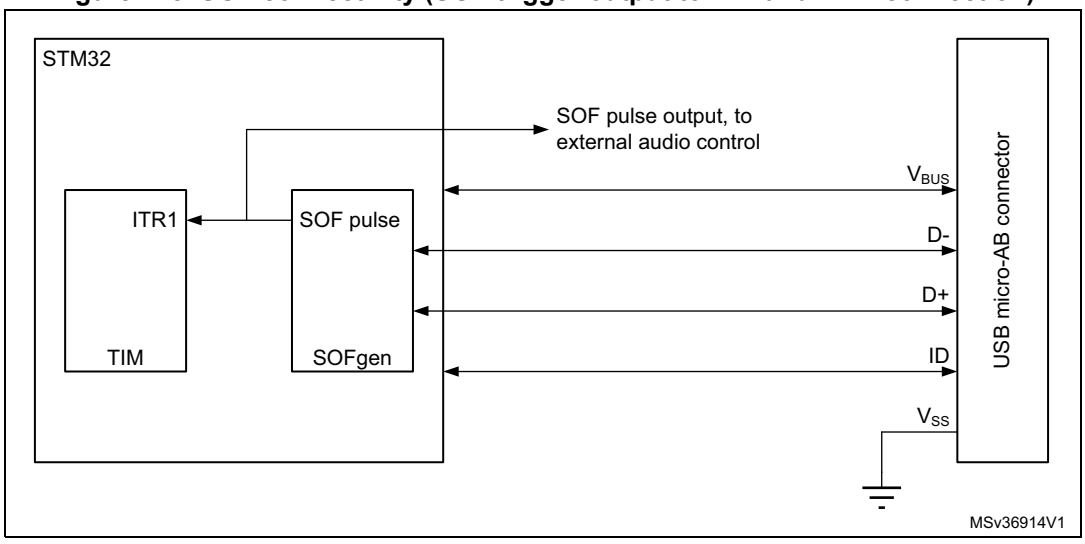
- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of 8 pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

56.8 SOF trigger

Figure 748. SOF connectivity (SOF trigger output to TIM and ITR1 connection)



The OTG_HS core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

56.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (HS/FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

56.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated.The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

56.9 OTG low-power modes

Table 492 below defines the STM32 low power modes and their compatibility with the OTG.

Table 492. Compatibility of STM32 low power modes with the OTG

Mode	Description	USB compatibility
Run	MCU fully active	Required when USB not in suspend state.
Sleep	USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.	Available while USB is in suspend state.
Stop	USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).	Available while USB is in suspend state.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.	Not compatible with USB applications.

1. Within Stop mode there are different possible settings. Some restrictions may also exist, please refer to *Section 5: Power control (PWR)* to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

The power consumption of the OTG PHY is controlled by two or three bits in the general core configuration register, depending on OTG revision supported.

- PHY power down (OTG_GCCFG/PWRDWN)

  It switches on/off the full-speed transceiver module of the PHY. It must be preliminarily set to allow any USB operation
- V_{BUS} detection enable (OTG_GCCFG/VBDEN)

  It switches on/off the V_{BUS} sensing comparators associated with OTG operations

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)

  When setting the stop PHY clock bit in the clock gating control register, most of the 48 MHz clock domain internal to the OTG full-speed core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the 48 MHz clock input is kept running by the application

  Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wakeup event is kept alive.
- Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)

  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG_HS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.
- USB system stop

  When the OTG_HS is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG_HS core automatically reactivates both system and USB clocks by asynchronous detection of remote wakeup (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG_HS core.

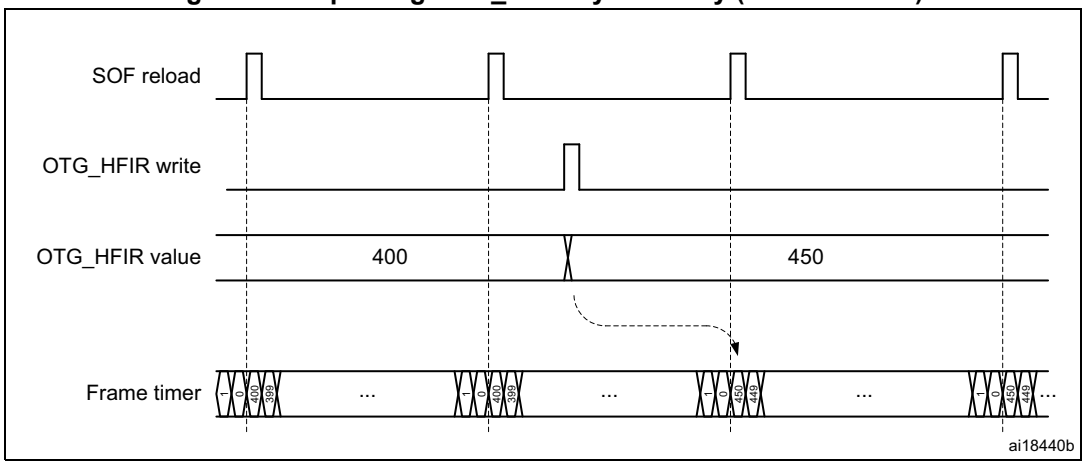
56.10 Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of micro-SOF framing period in host mode allowing to synchronize an external device with the micro-SOF frames.

When the OTG_HFIR register is changed within a current micro-SOF frame, the SOF period correction is applied in the next frame as described in *Figure 749*.

For a dynamic update, it is required to set RLDCTRL=0.

Figure 749. Updating OTG_HFIR dynamically (RLDCTRL = 0)

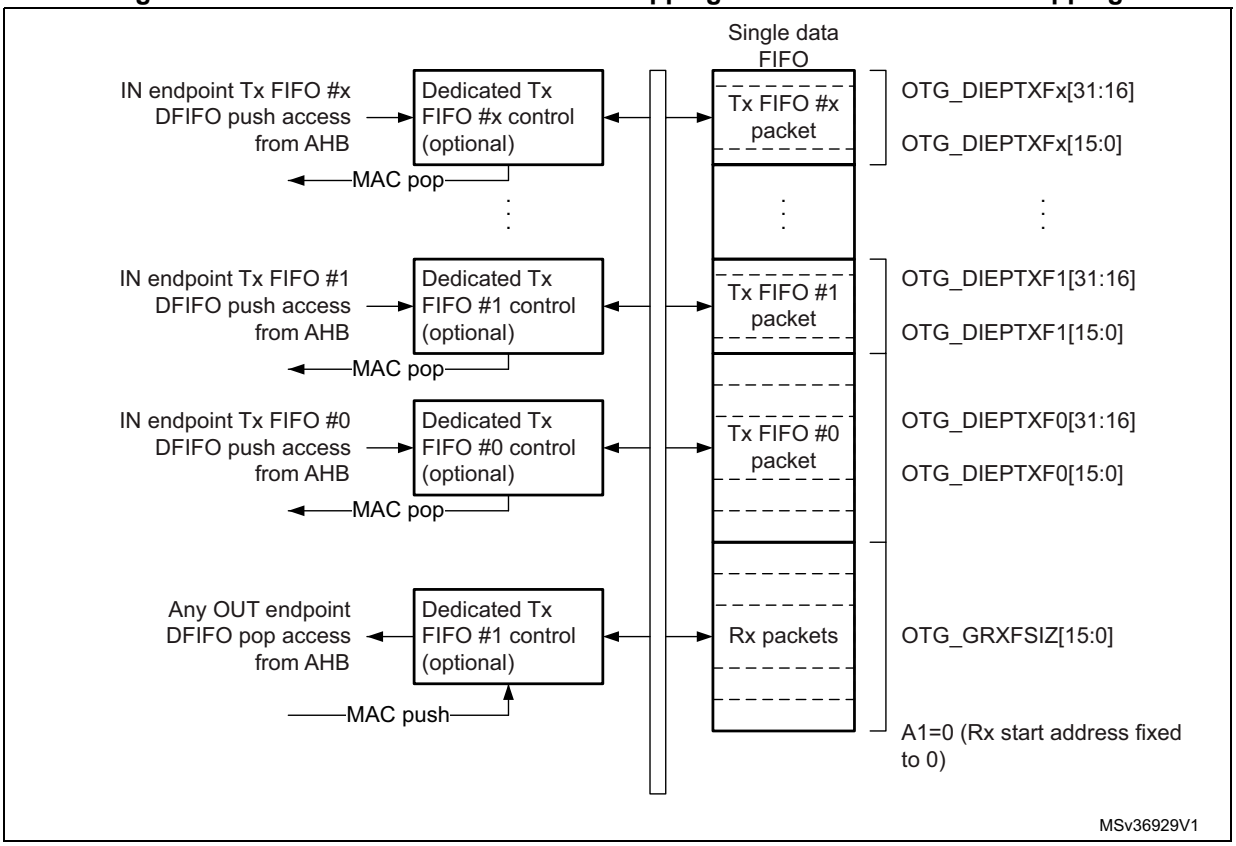


56.11 USB data FIFOs

The USB system features 4 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG_HS core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device's role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.

56.11.1 Peripheral FIFO architecture

Figure 750. Device-mode FIFO address mapping and AHB FIFO access mapping



Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG_HS core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

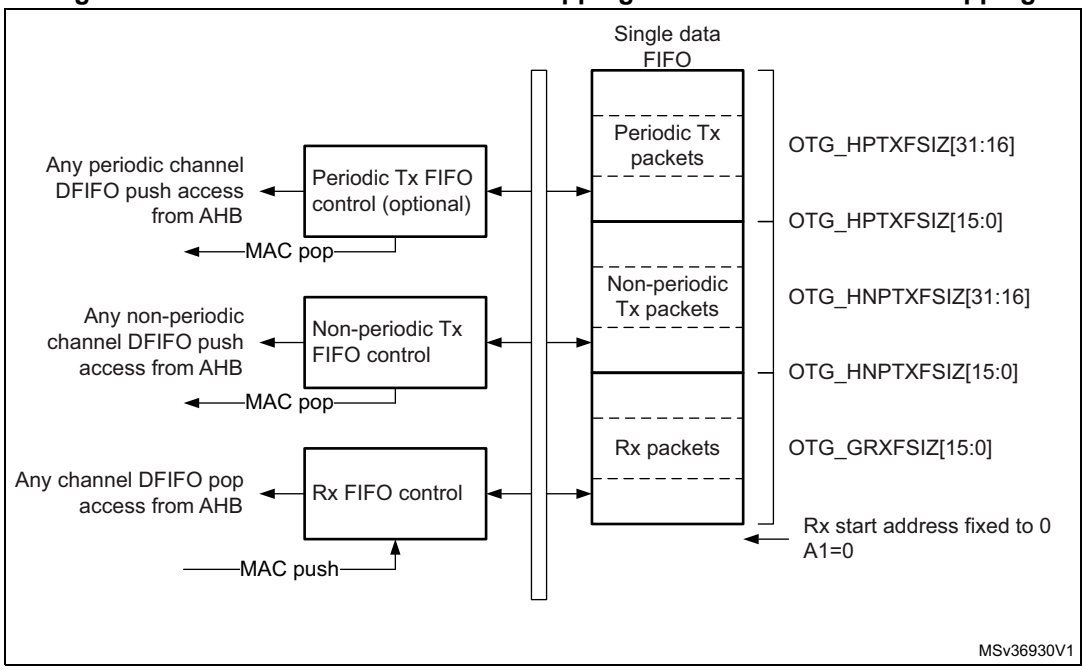
The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.

Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

56.11.2 Host FIFO architecture

Figure 751. Host-mode FIFO address mapping and AHB FIFO access mapping



Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG_HS core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.

Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG_HS core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG_HS core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG_HS core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

56.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application should allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.

Device RxFIFO =

(5 * number of control endpoints + 8) + ((largest USB packet used / 4) + 1 for status information) + (2 * number of OUT endpoints) + 1 for Global NAK

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = (5 * 1 + 8) + ((1,024 / 4) +1) + (2 * 4) + 1 = 279

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

Host mode

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = (largest USB packet used / 4) + 1 for status information + 1 transfer complete

Example: Host RxFIFO = ((1,024 / 4) + 1) + 1 = 258

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: Non-Periodic TxFIFO = (512 / 4) = 128

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = largest periodic USB packet used / 4

Example: Host Periodic TxFIFO = (1,024 / 4) = 256

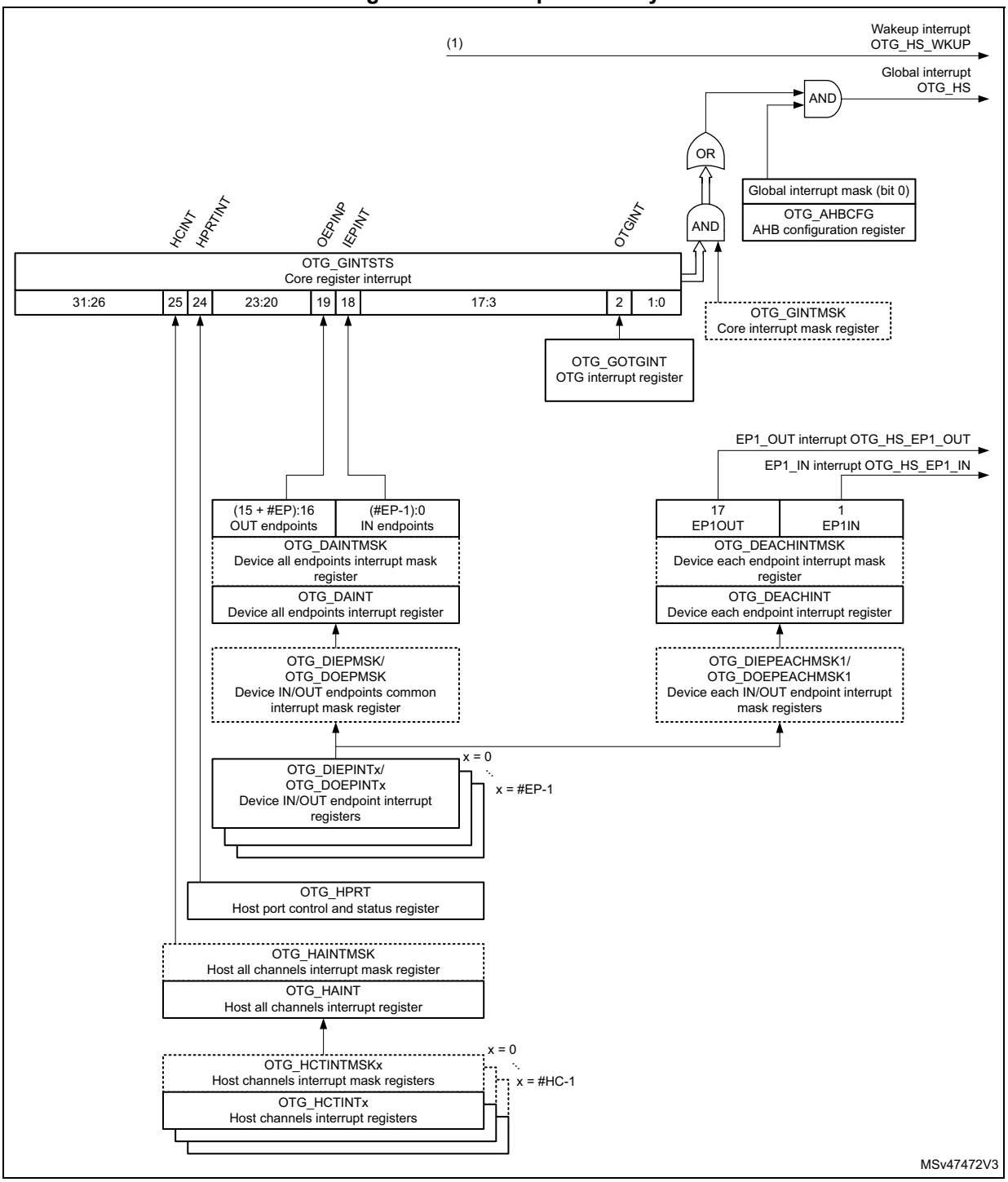
Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.

56.12 OTG_HS interrupts

When the OTG_HS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

Figure 752 shows the interrupt hierarchy.

Figure 752. Interrupt hierarchy



1. OTG_HS_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.

56.13 OTG_HS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG_HS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG_HS registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG_HS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

56.13.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

Table 493. Core global control and status registers (CSRs)

Acronym	Address offset	Register name
OTG_GOTGCTL	0x000	*Section 56.14.1: OTG control and status register (OTG_GOTGCTL)*
OTG_GOTGINT	0x004	*Section 56.14.2: OTG interrupt register (OTG_GOTGINT)*
OTG_GAHBCFG	0x008	*Section 56.14.3: OTG AHB configuration register (OTG_GAHBCFG)*
OTG_GUSBCFG	0x00C	*Section 56.14.4: OTG USB configuration register (OTG_GUSBCFG)*
OTG_GRSTCTL	0x010	*Section 56.14.5: OTG reset register (OTG_GRSTCTL)*
OTG_GINTSTS	0x014	*Section 56.14.6: OTG core interrupt register (OTG_GINTSTS)*
OTG_GINTMSK	0x018	*Section 56.14.7: OTG interrupt mask register (OTG_GINTMSK)*

Table 493. Core global control and status registers (CSRs) (continued)

Acronym	Address offset	Register name
OTG_GRXSTSR	0x01C	*Section 56.14.8: OTG receive status debug read register (OTG_GRXSTSR)*
		Section 56.14.9: OTG receive status debug read [alternate] (OTG_GRXSTSR)
OTG_GRXSTSP	0x020	*Section 56.14.10: OTG status read and pop registers (OTG_GRXSTSP)*
		Section 56.14.11: OTG status read and pop registers [alternate] (OTG_GRXSTSP)
OTG_GRXFSIZ	0x024	*Section 56.14.12: OTG receive FIFO size register (OTG_GRXFSIZ)*
OTG_HNPTXFSIZ/ OTG_DIEPTXF0(1)	0x028	*Section 56.14.13: OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)*
OTG_HNPTXSTS	0x02C	*Section 56.14.14: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)*
OTG_GCCFG	0x038	*Section 56.14.15: OTG general core configuration register (OTG_GCCFG)*
OTG_CID	0x03C	*Section 56.14.16: OTG core ID register (OTG_CID)*
OTG_GLPMCFG	0x54	*Section 56.14.17: OTG core LPM configuration register (OTG_GLPMCFG)*
OTG_HPTXFSIZ	0x100	*Section 56.14.18: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)*
OTG_DIEPTXFx	0x104 0x108 ... 0x120	*Section 56.14.19: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)*

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

Table 494. Host-mode control and status registers (CSRs)

Acronym	Offset address	Register name
OTG_HCFG	0x400	*Section 56.14.21: OTG host configuration register (OTG_HCFG)*
OTG_HFIR	0x404	*Section 56.14.22: OTG host frame interval register (OTG_HFIR)*
OTG_HFNUM	0x408	*Section 56.14.23: OTG host frame number/frame time remaining register (OTG_HFNUM)*
OTG_HPTXSTS	0x410	*Section 56.14.24: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)*
OTG_HAINT	0x414	*Section 56.14.25: OTG host all channels interrupt register (OTG_HAINT)*
OTG_HAINTMSK	0x418	*Section 56.14.26: OTG host all channels interrupt mask register (OTG_HAINTMSK)*

Table 494. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HPRT	0x440	*Section 56.14.27: OTG host port control and status register (OTG_HPRT)*
OTG_HCCHARx	0x500 0x520 ... 0x6E0	*Section 56.14.28: OTG host channel x characteristics register (OTG_HCCHARx)*
OTG_HCSPLTx	0x504 0x524 .... 0x6E4	*Section 56.14.29: OTG host channel x split control register (OTG_HCSPLTx)*
OTG_HCINTx	0x508 0x528 .... 0x6E8	*Section 56.14.30: OTG host channel x interrupt register (OTG_HCINTx)*
OTG_HCINTMSKx	0x50C 0x52C 0x6EC	*Section 56.14.31: OTG host channel x interrupt mask register (OTG_HCINTMSKx)*
OTG_HCTSIZx	0x510 0x530 .... 0x6F0	*Section 56.14.32: OTG host channel x transfer size register (OTG_HCTSIZx)*
OTG_HCDMAx	0x514 0x534 .... 0x6F4	*Section 56.14.33: OTG host channel x DMA address register (OTG_HCDMAx) (x = 0..15, where x = Channel number)*

Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 495. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_DCFG	0x800	*Section 56.14.35: OTG device configuration register (OTG_DCFG)*
OTG_DCTL	0x804	*Section 56.14.36: OTG device control register (OTG_DCTL)*
OTG_DSTS	0x808	*Section 56.14.37: OTG device status register (OTG_DSTS)*
OTG_DIEPMSK	0x810	*Section 56.14.38: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)*
OTG_DOEPMSK	0x814	*Section 56.14.39: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)*

Table 495. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DAINT	0x818	*Section 56.14.40: OTG device all endpoints interrupt register (OTG_DAINT)*
OTG_DAINTMSK	0x81C	*Section 56.14.41: OTG all endpoints interrupt mask register (OTG_DAINTMSK)*
OTG_DVBUSDIS	0x828	*Section 56.14.42: OTG device V_{BUS} discharge time register (OTG_DVBUSDIS)*
OTG_DVBUSPULSE	0x82C	*Section 56.14.43: OTG device V_{BUS} pulsing time register (OTG_DVBUSPULSE)*
OTG_DTHRCTL	0x830	*Section 56.14.44: OTG device threshold control register (OTG_DTHRCTL)*
OTG_DIEPEMPMSK	0x834	*Section 56.14.45: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)*
OTG_DEACHINT	0x838	*Section 56.14.46: OTG device each endpoint interrupt register (OTG_DEACHINT)*
OTG_DEACHINTMSK	0x83C	*Section 56.14.47: OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK)*
OTG_HS_DIEPEACHMSK1	0x844	*Section 56.14.48: OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHMSK1)*
OTG_HS_DOEPEACHMSK1	0x884	*Section 56.14.49: OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHMSK1)*
OTG_DIEPCTLx	0x900 0x920 ... 0xA00	*Section 56.14.50: OTG device IN endpoint x control register (OTG_DIEPCTLx)*
OTG_DIEPINTx	0x908 0x928 ... 0x9E8	*Section 56.14.51: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)*
OTG_DIEPTSIZ0	0x910	*Section 56.14.52: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)*
OTG_DIEPDMAx	0x914 0x934 ... 0x9F4	*Section 56.14.53: OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)*
OTG_DTXFSTSx	0x918 0x938 ..... 0x9F8	*Section 56.14.54: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)*

Table 495. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DIEPTSIZx	0x930 0x950 ... 0x9F0	*Section 56.14.55: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)*
OTG_DOEPCTL0	0xB00	*Section 56.14.56: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)*
OTG_DOEPINTx	0xB08 0XB28 ... 0xC08	*Section 56.14.57: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)*
OTG_DOEPTSIZ0	0xB10	*Section 56.14.58: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)*
OTG_DOEPDMAx	0xB14 0xB34 ... 0xC14	*Section 56.14.59: OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)*
OTG_DOEPCTLx	0xB20 0xB40 ... 0xC00	*Section 56.14.60: OTG device OUT endpoint x control register (OTG_DOEPCTLx)*
OTG_DOEPTSIZx	0xB30 0xB50 .. 0xBF0	*Section 56.14.61: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)*

Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 496. Data FIFO (DFIFO) access register map

FIFO access register section	Offset address	Access
Device IN endpoint 0/Host OUT Channel 0: DFIFO write access Device OUT endpoint 0/Host IN Channel 0: DFIFO read access	0x1000–0x1FFC	w r
Device IN endpoint 1/Host OUT Channel 1: DFIFO write access Device OUT endpoint 1/Host IN Channel 1: DFIFO read access	0x2000–0x2FFC	w r

Table 496. Data FIFO (DFIFO) access register map (continued)

FIFO access register section	Offset address	Access
...	...	...
Device IN endpoint x[1]/Host OUT Channel x[1]: DFIFO write access Device OUT endpoint x[1]/Host IN Channel x[1]: DFIFO read access	0xX000–0xXFFC	w r

1. Where x is 8in device mode and 15in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

Table 497. Power and clock gating control and status registers

Acronym	Offset address	Register name
OTG_PCGCCTL	0xE00–0xE04	*Section 56.14.62: OTG power and clock gating control register (OTG_PCGCCTL)*

56.14 OTG_HS registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

56.14.1 OTG control and status register (OTG_GOTGCTL)

Address offset: 0x000

Reset value: 0x0001 0000

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUR MOD	OTG VER	BSVLD	ASVLD	DBCT	CID STS
										r	rw	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EHEN	DHNP EN	HSHNP EN	HNP RQ	HNG SCS	BVALO VAL	BVALO EN	AVALO VAL	AVALO EN	VBVAL OVAL	VBVAL OEN	SRQ	SRQ SCS
			rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CURMOD:** Current mode of operation

Indicates the current mode (host or device).

0: Device mode

1: Host mode

Bit 20 **OTGVER:** OTG version

Selects the OTG revision.

0:OTG Version 1.3. OTG1.3 is obsolete for new product development.

1:OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 **BSVLD:** B-session valid

Indicates the device mode transceiver status.

0: B-session is not valid.

1: B-session is valid.

In OTG mode, the user can use this bit to determine if the device is connected or disconnected.

Note: Only accessible in device mode.

Bit 18 **ASVLD:** A-session valid

Indicates the host mode transceiver status.

0: A-session is not valid

1: A-session is valid

Note: Only accessible in host mode.

Bit 17 **DBCT:** Long/short debounce time

Indicates the debounce time of a detected connection.

0: Long debounce time, used for physical connections (100 ms + 2.5 µs)

1: Short debounce time, used for soft connections (2.5 µs)

Note: Only accessible in host mode.

Bit 16 **CIDSTS:** Connector ID status

Indicates the connector ID status on a connect event.

0: The OTG_HS controller is in A-device mode

1: The OTG_HS controller is in B-device mode

Note: Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN:** Embedded host enable

It is used to select between OTG A device state machine and embedded host state machine.

0: OTG A device state machine is selected

1: Embedded host state machine is selected

Bit 11 **DHNPEN:** Device HNP enabled

The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.

0: HNP is not enabled in the application

1: HNP is enabled in the application

Note: Only accessible in device mode.

Bit 10 **HSHNPEN:** host set HNP enable

The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.

0: Host Set HNP is not enabled

1: Host Set HNP is enabled

Note: Only accessible in host mode.

Bit 9 **HNPRQ:** HNP request

The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

0: No HNP request

1: HNP request

Note: Only accessible in device mode.

Bit 8 **HNGSCS:** Host negotiation success

The core sets this bit when host negotiation is successful. The core clears this bit when the HNP request (HNPRQ) bit in this register is set.

0: Host negotiation failure

1: Host negotiation success

Note: Only accessible in device mode.

Bit 7 **BVALOVAL:** B-peripheral session valid override value.

This bit is used to set override value for Bvalid signal when BVALOEN bit is set.

0: Bvalid value is '0' when BVALOEN = 1

1: Bvalid value is '1' when BVALOEN = 1

Note: Only accessible in device mode.

Bit 6 **BVALOEN:** B-peripheral session valid override enable.

This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.

0:Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core

1:Internally Bvalid received from the PHY is overridden with BVALOVAL bit value

Note: Only accessible in device mode.

Bit 5 **AVALOVAL:** A-peripheral session valid override value.

This bit is used to set override value for Avalid signal when AVALOEN bit is set.

0: Avalid value is '0' when AVALOEN = 1

1: Avalid value is '1' when AVALOEN = 1

Note: Only accessible in host mode.

Bit 4 **AVALOEN:** A-peripheral session valid override enable.

This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.

0:Override is disabled and Avalid signal from the respective PHY selected is used internally by the core

1:Internally Avalid received from the PHY is overridden with AVALOVAL bit value

Note: Only accessible in host mode.

Bit 3 **VBVALOVAL:** V_{BUS} valid override value.

This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.

0: vbusvalid value is '0' when VBVALOEN = 1

1: vbusvalid value is '1' when VBVALOEN = 1

Note: Only accessible in host mode.

Bit 2 **VBVALOEN:** V_{BUS} valid override enable.

This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.

0: Override is disabled and vbusvalid signal from the respective PHY selected is used internally by the core

1: Internally vbusvalid received from the PHY is overridden with VBVALOVAL bit value

Note: Only accessible in host mode.

Bit 1 **SRQ:** Session request

The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

If the user uses the USB 1.1 full-speed serial transceiver interface to initiate the session request, the application must wait until V_{BUS} discharges to 0.2 V, after the B-session valid bit in this register (BSVLD bit in OTG_GOTGCTL) is cleared.

0: No session request

1: Session request

Note: Only accessible in device mode.

Bit 0 **SRQSCS:** Session request success

The core sets this bit when a session request initiation is successful.

0: Session request failure

1: Session request success

Note: Only accessible in device mode.

56.14.2 OTG interrupt register (OTG_GOTGINT)

Address offset: 0x04

Reset value: 0x0000 0000

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID CHNG	DBC DNE	ADTO CHG	HNG DET	Res.
											rc_w1	rc_w1	rc_w1	rc_w1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HNSS CHG	SRSS CHG	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
						rc_w1	rc_w1						rc_w1		

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **IDCHNG**:

This bit when set indicates that there is a change in the value of the ID input pin.

Bit 19 **DBCDNE:** Debounce done

The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the OTG_GUSBCFG register (HNPCAP bit or SRPCAP bit in OTG_GUSBCFG, respectively).

Note: Only accessible in host mode.

Bit 18 **ADTOCHG:** A-device timeout change

The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

Note: Accessible in both device and host modes.

Bit 17 **HNGDET:** Host negotiation detected

The core sets this bit when it detects a host negotiation request on the USB.

Note: Accessible in both device and host modes.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **HNSSCHG:** Host negotiation success status change

The core sets this bit on the success or failure of a USB host negotiation request. The application must read the host negotiation success bit of the OTG_GOTGCTL register (HNGSCS bit in OTG_GOTGCTL) to check for success or failure.

Note: Accessible in both device and host modes.

Bits 7:3 Reserved, must be kept at reset value.

Bit 8 **SRSSCHG:** Session request success status change

The core sets this bit on the success or failure of a session request. The application must read the session request success bit in the OTG_GOTGCTL register (SRQSCS bit in OTG_GOTGCTL) to check for success or failure.

Note: Accessible in both device and host modes.

Bit 2 **SEDET:** Session end detected

The core sets this bit to indicate that the level of the voltage on V_{BUS} is no longer valid for a B-Peripheral session when $V_{BUS} < 0.8$ V.

Note: Accessible in both device and host modes.

Bits 1:0 Reserved, must be kept at reset value.

56.14.3 OTG AHB configuration register (OTG_GAHBCFG)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFE LVL	TXFE LVL	Res.	DMAEN	HBSTLEN[3:0]				GINT MSK
							rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PTXFELVL:** Periodic Tx FIFO empty level

Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered.

0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty

1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

Note: Only accessible in host mode.

Bit 7 **TXFELVL:** Tx FIFO empty level

In device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:

0:The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty

1:The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

In host mode, this bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:

0:The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty

1:The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bit 6 Reserved, must be kept at reset value.

Bit 5 **DMAEN:** DMA enabled

0: The core operates in slave mode

1: The core operates in DMA mode

Bits 4:1 **HBSTLEN[3:0]:** Burst length/type

0000 Single: Bus transactions use single 32 bit accesses (not recommended)

0001 INCR: Bus transactions use unspecified length accesses (not recommended, uses the INCR AHB bus command)

0011 INCR4: Bus transactions target 4x 32 bit accesses

0101 INCR8: Bus transactions target 8x 32 bit accesses

0111 INCR16: Bus transactions based on 16x 32 bit accesses

Others: Reserved

Bit 0 **GINTMSK:** Global interrupt mask

The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core.

0: Mask the interrupt assertion to the application.

1: Unmask the interrupt assertion to the application.

Note: Accessible in both device and host modes.

56.14.4 OTG USB configuration register (OTG_GUSBCFG)

Address offset: 0x00C

Reset value: 0x0000 1400

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD MOD	FH MOD	Res.	Res.	Res.	ULPI IPD	PTCI	PCCI	TSDPS	ULPIE VBUSI	ULPIE VBUSD	ULPI CSM	ULPI AR	ULPI FSL	Res.
	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYL PC	Res.	TRDT[3:0]				HNP CAP	SRP CAP	Res.	PHY SEL	Res.	Res.	Res.	TOCAL[2:0]		
rw		rw	rw	rw	rw	rw	rw		rw				rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FDMOD:** Force device mode

Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force device mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bit 29 **FHMOD:** Force host mode

Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force host mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **ULPIIPD:** ULPI interface protect disable

This bit controls the circuitry built in the PHY to protect the ULPI interface when the link tri-states stp and data. Any pull-up or pull-down resistors employed by this feature can be disabled. Refer to the ULPI specification for more details.
0: Enables the interface protection circuit
1: Disables the interface protection circuit

Bit 24 **PTCI:** Indicator pass through

This bit controls whether the complement output is qualified with the internal V_{BUS} valid comparator before being used in the V_{BUS} state in the RX CMD. Refer to the ULPI specification for more details.
0: Complement Output signal is qualified with the Internal V_{BUS} valid comparator
1: Complement Output signal is not qualified with the Internal V_{BUS} valid comparator

Bit 23 **PCCI:** Indicator complement

This bit controls the PHY to invert the ExternalVbusIndicator input signal, and generate the complement output. Refer to the ULPI specification for more details.
0: PHY does not invert the ExternalVbusIndicator signal
1: PHY inverts ExternalVbusIndicator signal

Bit 22 **TSDPS**: TermSel DLine pulsing selection

This bit selects utmi_termselect to drive the data line pulse during SRP (session request protocol).
0: Data line pulsing using utmi_txvalid (default)
1: Data line pulsing using utmi_termsel

Bit 21 **ULPIEVBUSI:** ULPI external V_{BUS} indicator

This bit indicates to the ULPI PHY to use an external V_{BUS} overcurrent indicator.
0: PHY uses an internal V_{BUS} valid comparator
1: PHY uses an external V_{BUS} valid comparator

Bit 20 **ULPIEVBUSD:** ULPI External V_{BUS} Drive

This bit selects between internal or external supply to drive 5 V on V_{BUS}, in the ULPI PHY.
0: PHY drives V_{BUS} using internal charge pump (default)
1: PHY drives V_{BUS} using external supply.

Bit 19 **ULPICSM:** ULPI clock SuspendM

This bit sets the ClockSuspendM bit in the interface control register on the ULPI PHY. This bit applies only in the serial and carkit modes.

0: PHY powers down the internal clock during suspend

1: PHY does not power down the internal clock

Bit 18 **ULPIAR:** ULPI Auto-resume

This bit sets the AutoResume bit in the interface control register on the ULPI PHY.

0: PHY does not use AutoResume feature

1: PHY uses AutoResume feature

Bit 17 **ULPIFSLS:** ULPI FS/LS select

The application uses this bit to select the FS/LS serial interface for the ULPI PHY. This bit is valid only when the FS serial transceiver is selected on the ULPI PHY.

0: ULPI interface

1: ULPI FS/LS serial interface

Bit 16 Reserved, must be kept at reset value.

Bit 15 **PHYLPC:** PHY Low-power clock select

This bit selects either 480 MHz or 48 MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48 MHz clock to save power.

0: 480 MHz internal PLL clock

1: 48 MHz external clock

In 480 MHz mode, the UTMI interface operates at either 60 or 30 MHz, depending on whether the 8- or 16-bit data width is selected. In 48 MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes.

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **TRDT[3:0]:** USB turnaround time

These bits allows to set the turnaround time in PHY clocks. They must be configured according to or *Table 499: TRDT values (HS)*, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

Note: Only accessible in device mode.

Bit 9 **HNPCAP:** HNP-capable

The application uses this bit to control the OTG_HS controller's HNP capabilities.

0: HNP capability is not enabled.

1: HNP capability is enabled.

Note: Accessible in both device and host modes.

Bit 8 **SRPCAP:** SRP-capable

The application uses this bit to control the OTG_HS controller's SRP capabilities. If the core operates as a non-SRP-capable
B-device, it cannot request the connected A-device (host) to activate V_{BUS} and start a session.

0: SRP capability is not enabled.

1: SRP capability is enabled.

Note: Accessible in both device and host modes.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PHYSEL:** Full speed serial transceiver select

0: USB 2.0 external ULPI high-speed PHY.

1: USB 1.1 full-speed serial transceiver.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]:** FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

Table 498. TRDT values

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	-	0x6

Table 499. TRDT values (HS)

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
30	-	0x9

56.14.5 OTG reset register (OTG_GRSTCTL)

Address offset: 0x10

Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB IDL	DMAR EQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXFNUM[4:0]					TXF FLSH	RXF FLSH	Res.	Res.	PSRST	CSRST
					rw	rw	rw	rw	rw	rs	rs			rs	rs

Bit 31 **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

Note: Accessible in both device and host modes.

Bit 30 **DMAREQ:** DMA request signal enabled

This bit indicates that the DMA request is in progress. Used for debug.

Bits 29:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]:** Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

00000:

- Non-periodic Tx FIFO flush in host mode
- Tx FIFO 0 flush in device mode

00001:

- Periodic Tx FIFO flush in host mode
- Tx FIFO 1 flush in device mode

00010: Tx FIFO 2 flush in device mode

...

01111: Tx FIFO 15 flush in device mode

10000: Flush all the transmit FIFOs in device or host mode.

Note: Accessible in both device and host modes.

Bit 5 **TXFFLSH:** Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

Read—NAK Effective interrupt ensures the core is not reading from the FIFO

Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.

Note: Accessible in both device and host modes.

Bit 4 **RXFFLSH:** Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

Note: Accessible in both device and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **PSRST:** Partial soft reset

Resets the internal state machines but keeps the enumeration info. Could be used to recover some specific PHY errors.

Note: Accessible in both device and host modes.

Bit 0 **CSRST:** Core soft reset

Resets the HCLK and PHY clock domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

– GATEHCLK bit in OTG_PCGCCTL
– STPPCLK bit in OTG_PCGCCTL
– FSLSPCS bits in OTG_HCFG
– DSPD bit in OTG_DCFG
– SDIS bit in OTG_DCTL
– OTG_GCCFG register

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.

The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit has been cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). The software must also check that bit 31 in this register is set to 1 (AHB Master is Idle) before starting any operation.

Typically, the software reset is used during software development and also when the user dynamically changes the PHY selection bits in the above listed USB configuration registers. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Note: Accessible in both device and host modes.

56.14.6 OTG core interrupt register (OTG_GINTSTS)

Address offset: 0x014

Reset value: 0x1400 0020

This register interrupts the application for system-level events in the current mode (device mode or host mode).

Some of the bits in this register are valid only in host mode, while others are valid in device mode only. This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	Res.	PTXFE	HCINT	HPRT INT	Res.	DATAF SUSP	IPXFR/ IN COMP ISO OUT	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1		r	r	r		rc_w1	rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wakeup detected interrupt

Wakeup interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG_HS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake If the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21 **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

INCOMPISOOUT: Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

56.14.7 OTG interrupt mask register (OTG_GINTMSK)

Address offset: 0x018

Reset value: 0x0000 0000

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wakeup detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 29 **DISCINT:** Disconnect detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 26 **PTXFEM:** Periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 25 **HCIM:** Host channels interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 24 **PRTIM:** Host port interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 23 **RSTDETM:** Reset detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 22 **FSUSPM:** Data fetch suspended mask
0: Masked interrupt
1: Unmasked interrupt
Only accessible in peripheral mode.

Bit 21 **IPXFRM:** Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.
IISOOXFRM: Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 20 **IISOIXFRM:** Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM:** End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 14 **ISOODRPM:** Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 13 **ENUMDNEM:** Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 11 **USBSUSPM:** USB suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 10 **ESUSPM:** Early suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM:** Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 6 **GINAKEFFM:** Global non-periodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 5 **NPTXFEM:** Non-periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 0 Reserved, must be kept at reset value.

56.14.8 OTG receive status debug read register (OTG_GRXSTSR)

Address offset for read: 0x01C

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

56.14.9 OTG receive status debug read [alternate] (OTG_GRXSTSR)

Address offset for read: 0x01C

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

56.14.10 OTG status read and pop registers (OTG_GRXSTSP)

Address offset for pop: 0x020

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number
Indicates the endpoint number to which the current received packet belongs.

56.14.11 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Address offset for pop: 0x020

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

56.14.12 OTG receive FIFO size register (OTG_GRXFSIZ)

Address offset: 0x024

Reset value: 0x0000 0400

The application can program the RAM size that must be allocated to the Rx FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFD[15:0]:** Rx FIFO depth
This value is in terms of 32-bit words.
Maximum value is 1024
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

56.14.13 OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPTXFD/TX0FD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSA/TX0FSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Host mode

Bits 31:16 **NPTXFD[15:0]:** Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]:** Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

Device mode

Bits 31:16 **TX0FD:** Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA:** Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

56.14.14 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Address offset: 0x02C

Reset value: 0x0008 0400

Note: In device mode, this register is not valid.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	NPTXQTOP[6:0]							NPTQXSAV[7:0]							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]:** Top of the non-periodic transmit request queue
Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
Bits 30:27: Channel/endpoint number
Bits 26:25:
00: IN/OUT token
01: Zero-length transmit packet (device IN/host OUT)
11: Channel halt command
Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]:** Non-periodic transmit request queue space available
Indicates the amount of free space available in the non-periodic transmit request queue. This queue holds both IN and OUT requests.
0: Non-periodic transmit request queue is full
1: 1 location available
2: locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]:** Non-periodic Tx FIFO space available
Indicates the amount of free space available in the non-periodic Tx FIFO.
Values are in terms of 32-bit words.
0: Non-periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where 0 ≤ n ≤ 512)
Others: Reserved

56.14.15 OTG general core configuration register (OTG_GCCFG)

Address offset: 0x038

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBDEN	SDEN	PDEN	DCD EN	BCDEN	PWR DWN
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2 DET	SDET	PDET	DCDET
												r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **VBDEN:** USB V_{BUS} detection enable

Enables V_{BUS} sensing comparators to detect V_{BUS} valid levels on the V_{BUS} PAD for USB host and device operation. If HNP and/or SRP support is enabled, V_{BUS} comparators are automatically enabled independently of VBDEN value.

0 = V_{BUS} detection disabled

1 = V_{BUS} detection enabled

Bit 20 **SDEN:** Secondary detection (SD) mode enable

This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly

Bit 19 **PDEN:** Primary detection (PD) mode enable

This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 18 **DCDEN:** Data contact detection (DCD) mode enable

This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 17 **BCDEN:** Battery charging detector (BCD) enable

This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD should be placed in OFF mode by clearing this bit to ‘0’ in order to allow the normal USB operation.

Bit 16 **PWRDWN:** Power down control

Used to activate the transceiver in transmission/reception. When reset, the transceiver is kept in power-down. When set, the BCD function must be off (BCDEN=0).

0 = USB FS transceiver disabled

1 = USB FS transceiver enabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PS2DET:** DM pull-up detection status

This bit is active only during PD and gives the result of comparison between DM voltage level and VLGC threshold. In normal situation, the DM level should be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, CDP or DCP)

1: PS2 port or proprietary charger detected

Bit 2 **SDET:** Secondary detection (SD) status

This bit gives the result of SD.

0: CDP detected

1: DCP detected

Bit 1 **PDET:** Primary detection (PD) status

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to CDP or DCP).

Bit 0 **DCDET:** Data contact detection (DCD) status

This bit gives the result of DCD.

0: data lines contact not detected

1: data lines contact detected

56.14.16 OTG core ID register (OTG_CID)

Address offset: 0x03C

Reset value: 0x0000 2300

This is a register containing the Product ID as reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCT_ID[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_ID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRODUCT_ID[31:0]:** Product ID field
Application-programmable ID field.

56.14.17 OTG core LPM configuration register (OTG_GLPMCFG)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EN BESL	LPMRCNTSTS[2:0]			SND LPM	LPMRCNT[2:0]			LPMCHIDX[3:0]				L1RSM OK
			rw	r	r	r	rs	rw	rw	rw	rw	rw	rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLP STS	LPMRSP[1:0]		L1DS EN	BESLTHRS[3:0]				L1SS EN	REM WAKE	BESL[3:0]				LPM ACK	LPM EN
r	r	r	rw	rw	rw	rw	rw	rw	rw/r	rw/r	rw/r	rw/r	rw/r	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ENBESL:** Enable best effort service latency
This bit enables the BESL feature as defined in the LPM errata:
0:The core works as described in the following document:
USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
1:The core works as described in the LPM Errata:
Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit should be set to '1' by application SW.

Bits 27:25 **LPMRCNTSTS[2:0]:** LPM retry count status
Number of LPM host retries still remaining to be transmitted for the current LPM sequence.
Note: Accessible only in host mode.

Bit 24 **SNDLPM:** Send LPM transaction

When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

Note: This bit must be set only when the host is connected to a local port.

Note: Accessible only in host mode.

Bits 23:21 **LPMRCNT:[2:0]** LPM retry count

When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

Note: Accessible only in host mode.

Bits 20:17 **LPMCHIDX[3:0]:** LPM Channel Index

The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

Note: Accessible only in host mode.

Bit 16 **L1RSMOK:** Sleep state resume OK

Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 µs ($T_{L1Residency}$).

This bit is reset when SLPSTS is 0.

1: The application or host can start resume from Sleep state

0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS:** Port sleep status

Device mode:

This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the $T_{L1TokenRetry}$ timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.

The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.

The core comes out of sleep:

– When there is any activity on the USB linestate

– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

Host mode:

The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.

The core clears this bit after:

– The core detects a remote L1 wakeup signal,

– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or

– The application sets the L1Resume/ remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).

0: Core not in L1

1: Core in L1

Bits 14:13 **LPMRST[1:0]:** LPM response

Device mode:
The response of the core to LPM transaction received is reflected in these two bits.
Host mode:
Handshake response received from local device for LPM transaction
11: ACK
10: NYET
01: STALL
00: ERROR (No handshake response)

Bit 12 **L1DSEN:** L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to '1' by application SW in all the cases.

Bits11:8 **BESLTHRS[3:0]:** BESL threshold

Device mode:
The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].
Host mode:
The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ($T_{L1HubDrvResume2}$) on the USB bus when it detects device initiated resume.
BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum $T_{L1HubDrvResume2}$.
Thres[3:0] host mode resume signaling time (µs):
0000: 75
0001: 100
0010: 150
0011: 250
0100: 350
0101: 450
0110: 950
All other values: reserved

Bit 7 **L1SSEN:** L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to '1' by application SW in all the cases.

Bit 6 **REMWAKE:** bRemoteWake value

Host mode:
The value of remote wake up to be sent in the wIndex field of LPM transaction.
Device mode (read-only):
This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

Bits 5:2 **BESL[3:0]:** Best effort service latency

Host mode:

The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{L1HubDrvResume1}$ for host initiated resume.

Device mode (read-only):

This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

BESL[3:0]T_{BESL} (µs)

0000: 125
0001: 150
0010: 200
0011: 300
0100: 400
0101: 500
0110: 1000
0111: 2000
1000: 3000
1001: 4000
1010: 5000
1011: 6000
1100: 7000
1101: 8000
1110: 9000
1111: 10000

Bit 1 **LPMACK:** LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

1: ACK

Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:

– No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
– Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
– No data pending in transmit queue (else NYET).

0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

– The received bLinkState is not L1 (STALL response), or
– An error is detected in either of the LPM token packets because of corruption (ERROR response).

Note: Accessible only in device mode.

Bit 0 **LPMEN:** LPM support enable

The application uses this bit to control the OTG_HS core LPM capabilities.

If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.

If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled
1: LPM capability is enabled

56.14.18 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0200 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXFSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PTXFSIZ[15:0]:** Host periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **PTXSA[15:0]:** Host periodic Tx FIFO start address
This field configures the memory start address for periodic transmit FIFO RAM.

56.14.19 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1..8)

Reset value: 0x0200 0200 + 0x200 * x

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INEPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INEPTXFD[15:0]:** IN endpoint Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]:** IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

56.14.20 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

56.14.21 OTG host configuration register (OTG_HCFG)

Address offset: 0x400

Reset value: 0x0000 0000

This register configures the core after power-on. Do not make changes to this register after initializing the host.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLSPCS[1:0]	
													r	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS:** FS- and LS-only support

The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.

Bits 1:0 **FSLSPCS[1:0]:** FS/LS PHY clock select

When the core is in FS host mode

01: PHY clock is running at 48 MHz

Others: Reserved

When the core is in LS host mode

00: Reserved

01: Select 48 MHz PHY clock frequency

10: Select 6 MHz PHY clock frequency

11: Reserved

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

56.14.22 OTG host frame interval register (OTG_HFIR)

Address offset: 0x404

Reset value: 0x0000 EA60

This register stores the frame interval information for the current speed to which the OTG_HS controller has enumerated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLD CTRL
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRIVL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RLDCTRL:** Reload control

This bit allows dynamic reloading of the HFIR register during run time.

0: The HFIR can be dynamically reloaded during run time.

1: The HFIR cannot be reloaded dynamically

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

Caution: RLDCTRL = 1 is not recommended.

Bits 15:0 **FRIVL[15:0]:** Frame interval

The value that the application programs to this field, specifies the interval between two consecutive micro-SOFs (HS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

– Frame interval = 125 µs × (FRIVL - 1) in high speed operation (PHYSEL = 0)

– Frame interval = 1 ms × (FRIVL - 1) in low/full speed operation (PHYSEL = 1)

56.14.23 OTG host frame number/frame time remaining register (OTG_HFNUM)

Address offset: 0x408

Reset value: 0x0000 3FFF

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTREM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRNUM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **FTREM[15:0]:** Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]:** Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

56.14.24 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)

Address offset: 0x410

Reset value: 0x0008 0100

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXQTOP[7:0]								PTXQSAV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFSAVL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PTXQTOP[7:0]:** Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit 31: Odd/Even frame

0: send in even frame

1: send in odd frame

Bits 30:27: Channel/endpoint number

Bits 26:25: Type

00: IN/OUT

01: Zero-length packet

11: Disable channel command

Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]:** Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.

00: Periodic transmit request queue is full

01: 1 location available

10: 2 locations available

bxn: n locations available (0 ≤ n ≤ 8)

Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]:** Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.

Values are in terms of 32-bit words

0000: Periodic Tx FIFO is full

0001: 1 word available

0010: 2 words available

bxn: n words available (where 0 ≤ n ≤ PTXFD)

Others: Reserved

56.14.25 OTG host all channels interrupt register (OTG_HAINT)

Address offset: 0x414

Reset value: 0x0000 0000

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in *Figure 752*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINT[15:0]:** Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

56.14.26 OTG host all channels interrupt mask register (OTG_HAINTMSK)

Address offset: 0x418

Reset value: 0x0000 0000

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINTM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINTM[15:0]:** Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

56.14.27 OTG host port control and status register (OTG_HPRT)

Address offset: 0x440

Reset value: 0x0000 0000

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 752*. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD[1:0]		PTCTL[3]
													r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTCTL[2:0]			PPWR	PLSTS[1:0]		Res.	PRST	PSUSP	PRES	POC CHNG	POCA	PEN CHNG	PENA	PCDET	PCSTS
rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD[1:0]:** Port speed

Indicates the speed of the device attached to this port.
01: Full speed
10: Low speed
11: Reserved
00: High speed

Bits 16:13 **PTCTL[3:0]:** Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved

Bit 12 **PPWR:** Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition.
0: Power off
1: Power on

Bits 11:10 **PLSTS[1:0]:** Port line status

Indicates the current logic level USB data lines
Bit 10: Logic level of OTG_DP
Bit 11: Logic level of OTG_DM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

0: Port not in reset

1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

High speed: 50 ms

Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).

0: Port not in suspend mode

1: Port in suspend mode

Bit 6 **PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wakeup sequence, as indicated by the port resume/remote wakeup detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:

1. The application sets this bit to drive resume signaling on the port.

2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.

3. If the core detects a USB remote wakeup sequence, as indicated by the port L1Resume/Remote L1Wakeup detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume.This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5 **POCCHNG:** Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

Bit 3 **PENCHNG:** Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2 **PENA:** Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled

1: Port enabled

Bit 1 **PCDET:** Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status

0: No device is attached to the port

1: A device is attached to the port

56.14.28 OTG host channel x characteristics register (OTG_HCCHARx)

Address offset: 0x500 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODD FRM	DAD[6:0]							MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR	EPNUM[3:0]				MPSIZ[10:0]										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA:** Channel enable

This field is set by the application and cleared by the OTG host.

0: Channel disabled

1: Channel enabled

Bit 30 **CHDIS:** Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM:** Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even frame

1: Odd frame

Bits 28:22 **DAD[6:0]:** Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MCNT[1:0]:** Multicount

This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used

00: Reserved. This field yields undefined results

01: 1 transaction

10: 2 transactions per frame to be issued for this endpoint

11: 3 transactions per frame to be issued for this endpoint

Note: This field must be set to at least 01.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Indicates the transfer type selected.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **LSDEV:** Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR:** Endpoint direction

Indicates whether the transaction is IN or OUT.

0: OUT

1: IN

Bits 14:11 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

Indicates the maximum packet size of the associated endpoint.

56.14.29 OTG host channel x split control register (OTG_HCSPLTx)

Address offset: 0x504 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLIT EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP LSPLT
rw															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XACTPOS[1:0]		HUBADDR[6:0]							PRTADDR[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPLITEN:** Split enable

The application sets this bit to indicate that this channel is enabled to perform split transactions.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **COMPLSPLT:** Do complete split

The application sets this bit to request the OTG host to perform a complete split transaction.

Bits 15:14 **XACTPOS[1:0]:** Transaction position

This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.

11: All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)

10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes)

00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes)

01: End. This is the last payload of this transaction (which is larger than 188 bytes)

Bits 13:7 **HUBADDR[6:0]:** Hub address

This field holds the device address of the transaction translator's hub.

Bits 6:0 **PRTADDR[6:0]:** Port address

This field is the port number of the recipient transaction translator.

56.14.30 OTG host channel x interrupt register (OTG_HCINTx)

Address offset: 0x508 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 752*. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERR	FRM OR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHB ERR	CHH	XFRC
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERR:** Data toggle error.

Bit 9 **FRMOR:** Frame overrun.

Bit 8 **BBERR:** Babble error.

Bit 7 **TXERR:** Transaction error.

Indicates one of the following errors occurred on the USB.

CRC check failure

Timeout

Bit stuff error

False EOP

Bit 6 **NYET:** Not yet ready response received interrupt.

Bit 5 **ACK:** ACK response received/transmitted interrupt.

Bit 4 **NAK:** NAK response received interrupt.

Bit 3 **STALL:** STALL response received interrupt.

Bit 2 **AHBERR:** AHB error

This error is generated only in Internal DMA mode when an AHB error occurs during an AHB read/write operation. The application can read the corresponding DMA channel address register to get the error address.

Bit 1 **CHH:** Channel halted.

Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.

Bit 0 **XFRC:** Transfer completed.

Transfer completed normally without any errors.

56.14.31 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

Address offset: 0x50C + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register reflects the mask for each channel status described in the previous section.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERRM:** Data toggle error mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 9 **FRMORM:** Frame overrun mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 8 **BBERRM:** Babble error mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 7 **TXERRM:** Transaction error mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 6 **NYET:** response received interrupt mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 5 **ACKM:** ACK response received/transmitted interrupt mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 4 **NAKM:** NAK response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STALLM:** STALL response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error.
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **CHHM:** Channel halted mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed mask
0: Masked interrupt
1: Unmasked interrupt

56.14.32 OTG host channel x transfer size register (OTG_HCTSIZx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **DPID[1:0]:** Data PID

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.
00: DATA0
01: DATA2
10: DATA1
11: SETUP (control) / MDATA (non-control)

Bits 28:19 **PKTCNT[9:0]:** Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).
The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer.
For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

56.14.33 OTG host channel x DMA address register (OTG_HCDMAx) (x = 0..15, where x = Channel number)

Address offset: 0x514 + 0x20 *x

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA address

This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

56.14.34 Device-mode registers

These registers must be programmed every time the core changes to device mode

56.14.35 OTG device configuration register (OTG_DCFG)

Address offset: 0x800

Reset value: 0x0220 0000

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PERSCHIVL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRATIM	XCVRDLY	Res.	PFIVL[1:0]		DAD[6:0]							Res.	NZLSOHSK	DSPD[1:0]	
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **PERSCHIVL[1:0]:** Periodic schedule interval

This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of the (micro) frame.

- When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data
- When no periodic endpoint is active, then the internal DMA engine services nonperiodic endpoints, ignoring this field
- After the specified time within a (micro) frame, the DMA switches to fetching nonperiodic endpoints

00: 25% of (micro)frame
01: 50% of (micro)frame
10: 75% of (micro)frame
11: Reserved

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **ERRATIM:** Erratic error interrupt mask

1: Mask early suspend interrupt on erratic error
0: Early suspend interrupt is generated on erratic error

Bit 14 **XCVRDLY:** Transceiver delay

Enables or disables delay in ULPI timing during device chirp.
0: Disable delay (use default timing)
1: Enable delay to default timing, necessary for some ULPI PHYs

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]:** Periodic frame interval

Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.
00: 80% of the frame interval
01: 85% of the frame interval
10: 90% of the frame interval
11: 95% of the frame interval

Bits 10:4 **DAD[6:0]:** Device address
The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Non-zero-length status OUT handshake
The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's status stage.
1:Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.
0:Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]:** Device speed
Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.
00: High speed
01: Full speed using HS
10: Reserved
11: Full speed using internal FS PHY

56.14.36 OTG device control register (OTG_DCTL)

Address offset: 0x804

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS BESL RJCT	Res.	Res.
													rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PO PRG DNE	CGO NAK	SGO NAK	CGI NAK	SGI NAK	TCTL[2:0]			GON STS	GIN STS	SDIS	RWU SIG
				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DSBESLRJCT:** Deep sleep BESL reject
Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE:** Power-on programming done
The application uses this bit to indicate that register programming is completed after a wakeup from power down mode.

Bit 10 **CGONAK:** Clear global OUT NAK
Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK:** Set global OUT NAK

Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8 **CGINAK:** Clear global IN NAK

Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK:** Set global IN NAK

Writing 1 to this field sets the Global non-periodic IN NAK.The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.

Bits 6:4 **TCTL[2:0]:** Test control

000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3 **GONSTS:** Global OUT NAK status

0:A handshake is sent based on the FIFO status and the NAK and STALL bit settings.
1:No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS:** Global IN NAK status

0:A handshake is sent out based on the data availability in the transmit FIFO.
1:A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS:** Soft disconnect

The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.
0:Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.
1:The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG:** Remote wakeup signaling

When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.
If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 µs ($T_{L1DevDrvResume}$) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

Table 500 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 500. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
Full speed	Suspended	1 ms + 2.5 µs
Full speed	Idle	2.5 µs
Full speed	Not Idle or suspended (Performing transactions)	2.5 µs
High speed	Not Idle or suspended (Performing transactions)	125 µs

56.14.37 OTG device status register (OTG_DSTS)

Address offset: 0x808

Reset value: 0x0000 0010

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS[1:0]		FNSOF[13:8]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNSOF[7:0]								Res.	Res.	Res.	Res.	EERR	ENUMSPD[1:0]		SUSP STS
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **DEVLNSTS[1:0]:** Device line status
Indicates the current logic level USB data lines.
Bit [23]: Logic level of D+
Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]:** Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EERR:** Erratic error

The core sets this bit to report any erratic errors.

Due to erratic errors, the OTG_HS controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]:** Enumerated speed

Indicates the speed at which the OTG_HS controller has come up after speed detection through a chirp sequence.

01: Reserved

10: Reserved

11: Full speed (PHY clock is running at 48 MHz)

Others: reserved

Bit 0 **SUSPSTS:** Suspend status

In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:

– When there is an activity on the USB data lines

– When the application writes to the remote wakeup signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).

56.14.38 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

Address offset: 0x810

Reset value: 0x0000 0000

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
		rw					rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask

0: Masked interrupt

1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

56.14.39 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

Address offset: 0x814

Reset value: 0x0000 0000

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NYET MSK	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	B2B STUPM	STS PHSR XM	OTEPD M	STUPM	AHB ERRM	EPDM	XFRC M
	rw	rw	rw				rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK:** NYET interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUPM:** Back-to-back SETUP packets received mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **STSPHSRXM**: Status phase received for control write mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** STUPM: SETUP phase done mask. Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

56.14.40 OTG device all endpoints interrupt register (OTG_DAINT)

Address offset: 0x818

Reset value: 0x0000 0000

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OEPINT[15:0]:** OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]:** IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.

56.14.41 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

Address offset: 0x81C

Reset value: 0x0000 0000

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **OEPM[15:0]:** OUT EP interrupt mask bits
One per OUT endpoint:
Bit 16 for OUT EP 0, bit 19 for OUT EP 3
0: Masked interrupt
1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]:** IN EP interrupt mask bits
One bit per IN endpoint:
Bit 0 for IN EP 0, bit 3 for IN EP 3
0: Masked interrupt
1: Unmasked interrupt

56.14.42 OTG device V_{BUS} discharge time register (OTG_DVBUSDIS)

Address offset: 0x0828

Reset value: 0x0000 17D7

This register specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBUSDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VBUSDT[15:0]:** Device V_{BUS} discharge time
Specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP. This value equals:
V_{BUS} discharge time in PHY clocks / 1 024
Depending on your V_{BUS} load, this value may need adjusting.

56.14.43 OTG device V_{BUS} pulsing time register (OTG_DVBUSPULSE)

Address offset: 0x082C

Reset value: 0x0000 05B8

This register specifies the V_{BUS} pulsing time during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DVBUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DVBUSP[15:0]:** Device V_{BUS} pulsing time. This feature is only relevant to OTG1.3.
Specifies the V_{BUS} pulsing time during SRP. This value equals:
V_{BUS} pulsing time in PHY clocks / 1 024

56.14.44 OTG device threshold control register (OTG_DTHRCTL)

Address offset: 0x0830

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN[8:0]									RXTHREN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXTHRLEN[8:0]									ISOTHREN	NONISOTHREN
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **ARPEN:** Arbiter parking enable
This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default parking is enabled.

Bit 26 Reserved, must be kept at reset value.

Bits 25:17 **RXTHRLEN[8:0]:** Receive threshold length
This field specifies the receive thresholding size in 32-bit words. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight 32-bit words. The recommended value for RXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 16 **RXTHREN:** Receive threshold enable
When this bit is set, the core enables thresholding in the receive direction.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:2 **TXTHRLEN[8:0]:** Transmit threshold length
This field specifies the transmit thresholding size in 32-bit words. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmitting on the USB. The threshold length has to be at least eight 32-bit words. This field controls both isochronous and nonisochronous IN endpoint thresholds. The recommended value for TXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 1 **ISOTHREN:** ISO IN endpoint threshold enable
When this bit is set, the core enables thresholding for isochronous IN endpoints.

Bit 0 **NONISOTHREN:** Nonisochronous IN endpoints threshold enable
When this bit is set, the core enables thresholding for nonisochronous IN endpoints.

56.14.45 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

Address offset: 0x834

Reset value: 0x0000 0000

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFEM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM[15:0]:** IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
0: Masked interrupt
1: Unmasked interrupt

56.14.46 OTG device each endpoint interrupt register (OTG_DEACHINT)

Address offset: 0x0838

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1 INT	Res.
														r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1 INT	Res.
														r	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **OEP1INT:** OUT endpoint 1 interrupt bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **IEP1INT:** IN endpoint 1interrupt bit

Bit 0 Reserved, must be kept at reset value.

56.14.47 OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK)

Address offset: 0x083C

Reset value: 0x0000 0000

There is one interrupt bit for endpoint 1 IN and one interrupt bit for endpoint 1 OUT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1INTM	Res.
														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1INTM	Res.
														rw	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **OEP1INTM:** OUT endpoint 1 interrupt mask bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **IEP1INTM:** IN endpoint 1 interrupt mask bit

Bit 0 Reserved, must be kept at reset value.

56.14.48 OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHMSK1)

Address offset: 0x844

Reset value: 0x0000 0000

This register works with the OTG_DIEPINT1 register to generate a dedicated interrupt OTG_HS_EP1_IN for endpoint #1. The IN endpoint interrupt for a specific status in the OTG_DOEPINT1 register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	Res.	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
		rw					rw		rw		rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

56.14.49 OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHMSK1)

Address offset: 0x884

Reset value: 0x0000 0000

This register works with the OTG_DOEPINT1 register to generate a dedicated interrupt OTG_HS_EP1_OUT for endpoint #1. The OUT endpoint interrupt for a specific status in the OTG_DOEPINT1 register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NYET MSK	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	B2B STUPM	Res.	OTEPD M	STUPM	AHB ERRM	EPDM	XFRC M
	rw	rw	rw				rw		rw		rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK:** NYET interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUPM:** Back-to-back SETUP packets received mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** STUPM: SETUP phase done mask
Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

56.14.50 OTG device IN endpoint x control register (OTG_DIEPCTLx)

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SD0 PID/ SEVN FRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	EO NUM/ DPID
rs	rs	w	w	w	w	rw	rw	rw	rw	rw/rs		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.
For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.
Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.
0: Even frame
1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk IN endpoints only.
Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.
0: DATA0
1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

56.14.51 OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

Address offset: 0x908 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 752*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	Res.	PKTD RPSTS	Res.	Res.	TXFIF OUD RN	TXFE	IN EPNE	IN EPNM	ITTXFE	TOC	AHB ERR	EP DISD	XFRC
		rc_w1		rc_w1			rc_w1	r	r	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PKTDRPSTS:** Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFIFOUDRN:** Transmit Fifo Underrun (TxfifoUndrn)

The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when Thresholding is enabled

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 **INEPNM**: IN token received with EP mismatch

Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE:** IN token received when Tx FIFO is empty

Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 **AHBERR**: AHB error

This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

56.14.52 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)

Address offset: 0x910

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT[1:0]		Res.	Res.	Res.
											rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

56.14.53 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)

Address offset: 0x914 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

56.14.54 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)

Address offset: 0x918 + 0x20 * x, (x = 0 to 8)

Reset Value: 0x0000 0200

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTFSAV[15:0]:** IN endpoint Tx FIFO space available

Indicates the amount of free space available in the endpoint Tx FIFO.
Values are in terms of 32-bit words:
0x0: Endpoint Tx FIFO is full
0x1: 1 word available
0x2: 2 words available
0xn: n words available
Others: Reserved

56.14.55 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

Address offset: 0x910 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]:** Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

56.14.56 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

Address offset: 0xB00

Reset value: 0x0000 8000

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	Res.
w	r			w	w					rs	rw	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														r	r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Hardcoded to 2'b00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

56.14.57 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

Address offset: 0xB08 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 752*. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STPK TRX	NYET	NAK	BERR	Res.	Res.	Res.	OUT PKT ERR	Res.	B2B STUP	STSPH SRX	OTEP DIS	STUP	AHB ERR	EP DISD	XFRC
rc_w1	rc_w1	rc_w1	rc_w1				rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **STPKTRX:** Setup packet received.

Applicable for control OUT endpoints in only in the Buffer DMA Mode. Set by the OTG_HS, this bit indicates that this buffer holds 8 bytes of setup data. There is only one setup packet per buffer. On receiving a setup packet, the OTG_HS closes the buffer and disables the corresponding endpoint after SETUP_COMPLETE status is seen in the Rx FIFO. OTG_HS puts a SETUP_COMPLETE status into the Rx FIFO when it sees the first IN or OUT token after the SETUP packet for that particular endpoint. The application must then re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Because of the above behavior, OTG_HS can receive any number of back to back setup packets and one buffer for every setup packet is used.

Bit 14 **NYET:** NYET interrupt

This interrupt is generated when a NYET response is transmitted for a non isochronous OUT endpoint.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR:** Babble error interrupt

The core generates this interrupt when babble is received for the endpoint.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERR**: OUT packet error

This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. This interrupt is valid only when thresholding is enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP:** Back-to-back SETUP packets received

Applies to control OUT endpoint only.

This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Bit 5 **STSPHSRX**: Status phase received for control write

This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG_HS has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled

Applies only to control OUT endpoints.

Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done

Applies to control OUT endpoint only.

Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 **AHBERR**: AHB error

This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

56.14.58 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

Address offset: 0xB10

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–8.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STUPCNT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.
	rw	rw										rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT[1:0]:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

56.14.59 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)

Address offset: 0xB14 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

56.14.60 OTG device OUT endpoint x control register (OTG_DOEPCTLx)

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID/ SODD FRM	SD0 PID/ SEVN FRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	EO NUM/ DPID
rs	rs	w	w	w	w					rw/rs	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

SODDFRM: Set odd frame

Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk OUT endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN and OUT endpoints only.
Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.
0: Even frame
1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.
Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.
0: DATA0
1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

56.14.61 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

Address offset: 0xB10 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID/ STUPCNT[1:0]		PKTCNT[9:0]										XFRSIZ		
	r/rw	r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RXDPID[1:0]:** Received data PID

Applies to isochronous OUT endpoints only.
This is the data PID received in the last packet for this endpoint.
00: DATA0
01: DATA2
10: DATA1
11: MDATA

STUPCNT[1:0]: SETUP packet count

Applies to control OUT endpoints only.
This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

56.14.62 OTG power and clock gating control register (OTG_PCGCCTL)

Address offset: 0xE00

Reset value: 0x200B 8000

This register is available in host and device modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHY SLEEP	ENL1 GTG	PHY SUSP	Res.	Res.	GATE HCLK	STPP CLK
								r	r	rw	r			rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SUSP:** Deep Sleep

This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6 **PHYSLEEP:** PHY in Sleep

This bit indicates that the PHY is in the Sleep state.

Bit 5 **ENL1GTG:** Enable sleep clock gating

When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP:** PHY suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

56.14.63 OTG_HS register map

The table below gives the USB OTG register map and reset values.

Table 501. OTG_HS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_GOTGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURMOD	OTGVER	BSVLD	ASVLD	DBCT	CIDSTS	Res.	Res.	Res.	EHEN	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	BVALOVAL	BVALOEN	AVALOVAL	AVALOEN	VBVALOVA	VBVALOEN	SRQ	SRQSCS
	Reset value											0	0	0	0	0	1				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	OTG_GOTGINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDCHNG	DBCDNE	ADTOCHG	HNGDET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HNSSCHG	SRSSCHG	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
	Reset value												0	0	0	0							-	0	0						0		
0x008	OTG_GAHBCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFELVL	TXFELVL	Res.	DMAEN	HBSTLEN				GINTMSK
	Reset value																								0	0		0	0	0	0	0	0
0x00C	OTG_GUSBCFG	Res.	FDMOD	FHMOD	Res.	Res.	Res.	ULPIIPD	PTCI	PCCI	TSDPS	ULPIEVBUSI	ULPIEVBUSD	ULPICSM	ULPIAR	ULPIFSL	Res.	PHYLPC	Res.	TRDT				HNPCAP	SRPCAP	Res.	PHYSEL	Res.	Res.	Res.	TOCAL		
	Reset value		0	0				0	0	0	0	0	0	0	0	0		0		0	1	0	1	0	0		1				0	0	0
0x010	OTG_GRSTCTL	AHBIDL	DMAREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFNUM					TXFFLSH	RXFFLSH	Res.	Res.	PSRST	CSRST
	Reset value	1	0																				0	0	0	0	0	0	0			0	0
0x014	OTG_GINTSTS	WKUPINT	SRQINT	DISCINT	CIDSCHG	Res.	PTXFE	HCINT	HPRTINT	Res.	DATAFSUSP	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	1		1	0	0		0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	OTG_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0	

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**OTG_ GRXSTSR (Device mode)**	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSR (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**OTG_ GRXSTSP (Device mode)**	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSP (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**OTG_ GRXFSIZ**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	**OTG_ HNPTXFSIZ/ OTG_ DIEPTXF0**	NPTXFD/TX0FD																NPTXFSA/TX0FSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	**OTG_ HNPTXSTS**	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	**OTG_ GCCFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBDEN	SDEN	PDEN	DCDEN	BCDEN	PWRDWN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET
	Reset value											0	0	0	0	0	0													X	X	X	X
0x03C	**OTG_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
0x054	**OTG_ GLPMCFG**	Res.	Res.	Res.	ENBESL	LPMR CNTSTS			SNDLPM	LPM RCNT			LPMCHIDX				L1RSMOK	SLPSTS	LPM RSP		L1DSEN	BESLTHRS			L1SSEN	REMWAKE	BESL			LPMACK	LPMEN		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	**OTG_ HPTXFSIZ**	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x104	**OTG_ DIEPTXF1**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	**OTG_ DIEPTXF2**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
. . . .	. . . .	. . . .																															
. . . .	. . . .	. . . .																															
0x120	**OTG_ DIEPTXF7**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0x400	**OTG_ HCFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLS PCS	
	Reset value																														0	0	0
0x404	**OTG_ HFIR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLDCTR	FRIVL															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	**OTG_ HFNUM**	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	**OTG_ HPTXSTS**	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x414	**OTG_ HAINT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	**OTG_ HAINTMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	**OTG_ HPRT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD		PTCTL				PPWR	PLSTS		Res.	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	Reset value														0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x500	**OTG_ HCCHAR0**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x504	**OTG_ HCSPLT0**	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_ HCINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50C	**OTG_ HCINTMSK0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x510	**OTG_ HCTSIZ0**	Res.	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	**OTG_ HCDMA0**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	**OTG_ HCCHAR1**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP	LSDEV	Res.	EPDIR	EPNUM					MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x524	**OTG_ HCSPLT1**	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	**OTG_ HCINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x52C	**OTG_ HCINTMSK1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x530	**OTG_ HCTSIZ1**	Res.	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x534	**OTG_ HCDMA1**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	. . . .																															
0x660	**OTG_ HCCHAR11**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP	LSDEV	Res.	EPDIR	EPNUM					MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x664	**OTG_ HCSPLT11**	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x668	**OTG_ HCINT11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x66C	**OTG_ HCINTMSK11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x670	**OTG_ HCTSIZ11**	Res.	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x674	**OTG_ HCDMA11**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0x6E0	**OTG_ HCCHAR15**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP	LSDEV	Res.	EPDIR	EPNUM					MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E4	**OTG_ HCSPLT15**	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E8	**OTG_ HCINT15**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x6EC	**OTG_ HCINTMSK15**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x6F0	**OTG_ HCTSIZ15**	Res.	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6F4	**OTG_ HCDMA15**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	**OTG_ DCFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRATIM	XCVRDLY	Res.	PFIVL		DAD							Res.	NZLSOHSK	DSPD	
	Reset value																	0	0		0	0	0	0	0	0	0	0	0		0	0	0

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804	**OTG_ DCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	Res.	Res.	Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value														0							0	0	0	0	0	0	0	0	0	0	1	0
0x808	**OTG_ DSTS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV LN STS	FNSOF															Res.	Res.	Res.	Res.	EERR	ENUMSPD		SUSPSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	**OTG_ DIEPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	Reset value																			0					0		0	0	0	0	0	0	0
0x814	**OTG_ DOEPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYETMSK	NAKMSK	BERRM	Res.	Res.	Res.	OUTPKTERRM	Res.	B2BSTUPM	Res.	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
	Reset value																		0	0	0				0		0		0	0	0	0	0
0x818	**OTG_ DAINT**	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	**OTG_ DAINTMSK**	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	**OTG_ DVBUSDIS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	**OTG_DVB USPULSE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVBUSP															
	Reset value																	0	0	0	0	0	1	0	1	1	0	1	1	1	0	0	0
0x830	**OTG_ DTHRCTL**	Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN									RXTHREN	Res.	Res.	Res.	Res.	TXTHRLEN										ISOTHREN	NONISOTHREN
	Reset value					0											0					0	0	0	0	0	0	0	0	0	0	0	0
0x834	**OTG_DIE PEMPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x838	**OTG_ DEACHINT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1INT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1INT	Res.
	Reset value															0																0	

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x83C	OTG_DEACHINTMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1INTM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1INTM	Res.
	Reset value															0																0	
0x844	OTG_HS_DIEPEACHMSK1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	Res.	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	Reset value																			0					0	-	0		0	0	0	0	0
0x884	OTG_HS_DOEPEACHMSK1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYETMSK	NAKMSK	BERRM	Res.	Res.	Res.	OUTPKTERRM	Res.	B2BSTUPM	Res.	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
	Reset value																		0	0	0				0		0		0	0	0	0	0
0x900	OTG_DIEPCTL0	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	OTG_DIEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x910	OTG_DIEPTSIZ0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x914	OTG_DIEPDMA	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x918	OTG_DTXFSTS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	OTG_DIEPCTL1	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x928	**OTG_ DIEPINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x930	**OTG_ DIEPTSIZ1**	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x938	**OTG_ DTXFSTS1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x940	**OTG_ DIEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
. . . .	. . . .	.																															
. . . .	. . . .	.																															
. . . .	. . . .	.																															
0x9E0	**OTG_ DIEPCTL7**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0x9E8	**OTG_ DIEPINT7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
. . . .	. . . .	. . . .																															
0x9F0	**OTG_ DIEPTSIZ7**	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	. . . .																															
0x9F8	**OTG_ DTXFSTS7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0xB00	**OTG_ DOEPCTL0**	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB08	**OTG_ DOEPINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0				0		0	0	0	0	0	0	0
0xB10	**OTG_ DOEPTSIZ0**	Res.	STUPCNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB14	**OTG_ DOEPDMA0**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB20	**OTG_ DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB28	**OTG_ DOEPINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0				0		0	0	0	0	0	0	0
0xB30	**OTG_ DOEPTSIZ1**	Res.	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 501. OTG_HS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB34	**OTG_ DOEPDMA1**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0xC00	**OTG_ DOEPCTL8**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xC08	**OTG_ DOEPINT8**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0		-		0		0	0	0	0	0	0	0
0xC10	**OTG_ DOEPTSIZ8**	Res.	RXDPID/ STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC14	**OTG_ DOEPDMA8**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	**OTG_ PCGCCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHYSLEEP	ENL1GTG	PHYSUSP	Res.	Res.	GATEHCLK	STPPCLK
	Reset value																									0	0	0	0			0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

56.15 OTG_HS programming model

56.15.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG_HS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG_HS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   - Periodic Tx FIFO empty level
2. Program the following fields in the OTG_GUSBCFG register:
   - HNP capable bit
   - SRP capable bit
   - OTG_HS timeout calibration field
   - USB turnaround time field
3. The software must unmask the following bits in the OTG_GINTMSK register:

   OTG interrupt mask

   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG_HS controller is operating in host or device mode.

56.15.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. This drives V_{BUS} on the USB.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

56.15.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
2. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
3. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.

Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion on page 2711*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

56.15.4 DMA mode

The OTG host uses the AHB master interface to fetch the transmit packet data (AHB to USB) and receive the data update (USB to AHB). The AHB master uses the programmed DMA address (OTG_HCDMAx register in host mode and OTG_DIEPDMAx/OTG_DOEPDMAx register in peripheral mode) to access the data buffers.

56.15.5 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
2. Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).
7. Program the selected channels in the OTG_HCSPLTx register(s) with the hub and port addresses (split transactions only).
8. Program the selected channels in the OTG_HCDMAx register(s) with the buffer start address (DMA transactions only).

Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_HS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG_HS host does not interrupt the transaction that has already been started on the USB.

To disable a channel in DMA mode operation, the application does not need to check for space in the request queue. The OTG_HS host checks for space to write the disable

request on the disabled channel's turn during arbitration. Meanwhile, all posted requests are dropped from the request queue when the CHDIS bit in OTG_HCCHARx is set to 1.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1, and the CHENA bit cleared to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When an XFRC interrupt in OTG_HCINTx is received during a non periodic IN transfer or high-bandwidth interrupt IN transfer
3. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).
4. When the application aborts a transfer before normal completion.

Ping protocol

When the OTG_HS host operates in high speed, the application must initiate the ping protocol when communicating with high-speed bulk or control (data and status stage) OUT endpoints.The application must initiate the ping protocol when it receives a NAK/NYET/TXERR interrupt. When the OTG_HS host receives one of the above responses, it does not continue any transaction for a specific endpoint, drops all posted or fetched OUT requests (from the request queue), and flushes the corresponding data (from the transmit FIFO).This is valid in slave mode only. In Slave mode, the application can send a ping token either by setting the DOPING bit in OTG_HCTSIZx before enabling the channel or by just writing the OTG_HCTSIZx register with the DOPING bit set when the channel is already enabled. This enables the OTG_HS host to write a ping request entry to the request queue. The application must wait for the response to the ping token (a NAK, ACK, or TXERR interrupt) before continuing the transaction or sending another ping token. The application can continue the data transaction only after receiving an ACK from the OUT endpoint for the requested ping. In DMA mode operation, the application does not need to set the DOPING bit in OTG_HCTSIZx for a NAK/NYET response in case of bulk/control OUT. The OTG_HS host automatically sets the DOPING bit in OTG_HCTSIZx, and issues the ping tokens for bulk/control OUT. The OTG_HS host continues sending ping tokens until it receives an ACK, and then switches automatically to the data transaction.

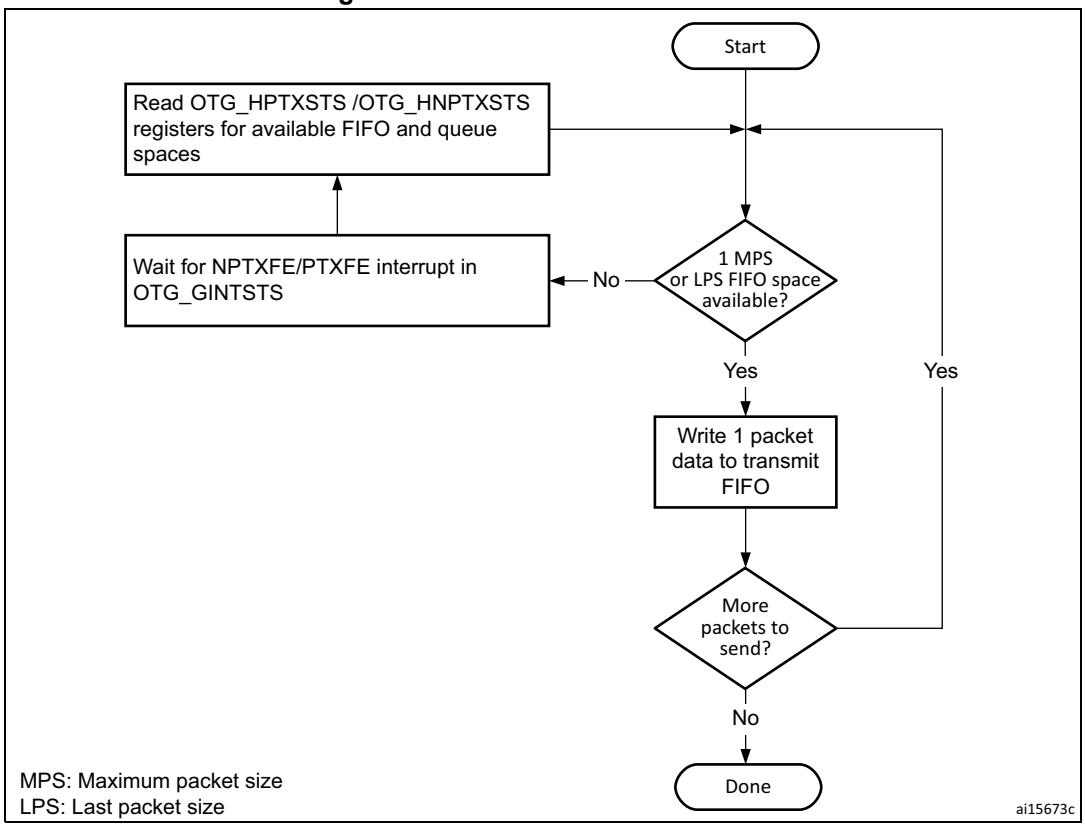
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG_HS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG_HS host determines the actual packet size based on the programmed maximum packet size and transfer size.

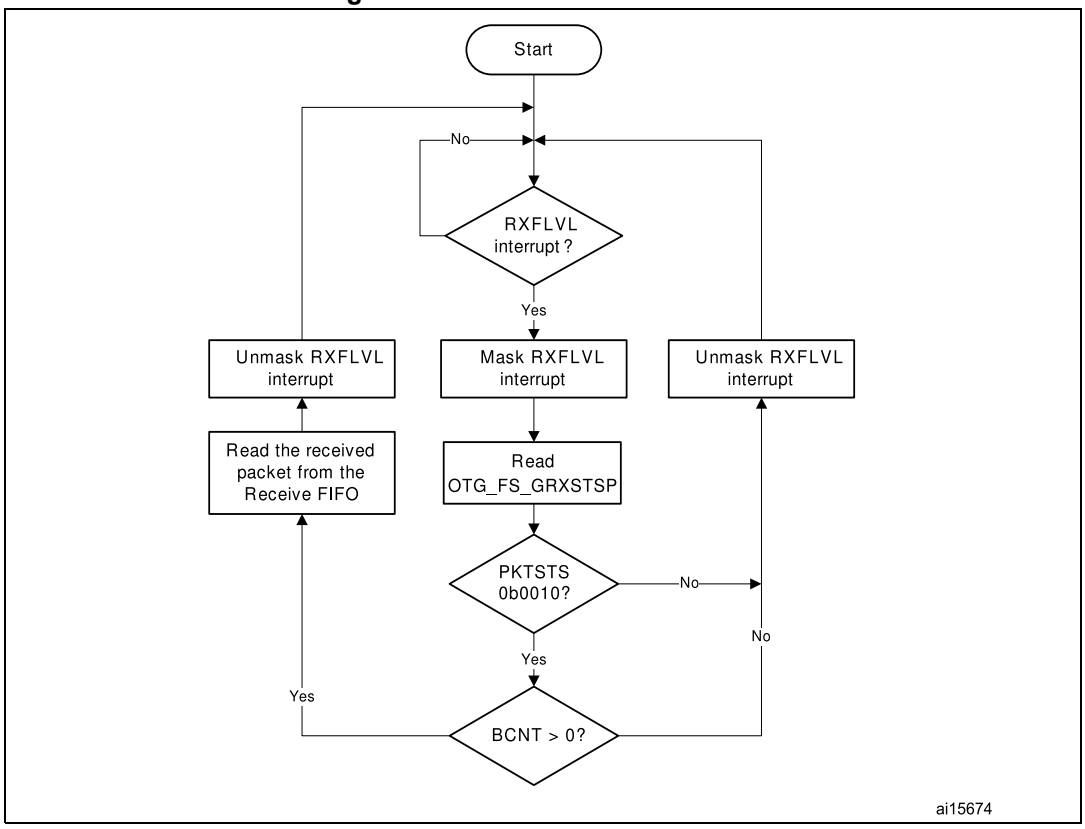
Figure 753. Transmit FIFO write task



- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 754. Receive FIFO read task



- **Bulk and control OUT/SETUP transactions**

  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 755*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:

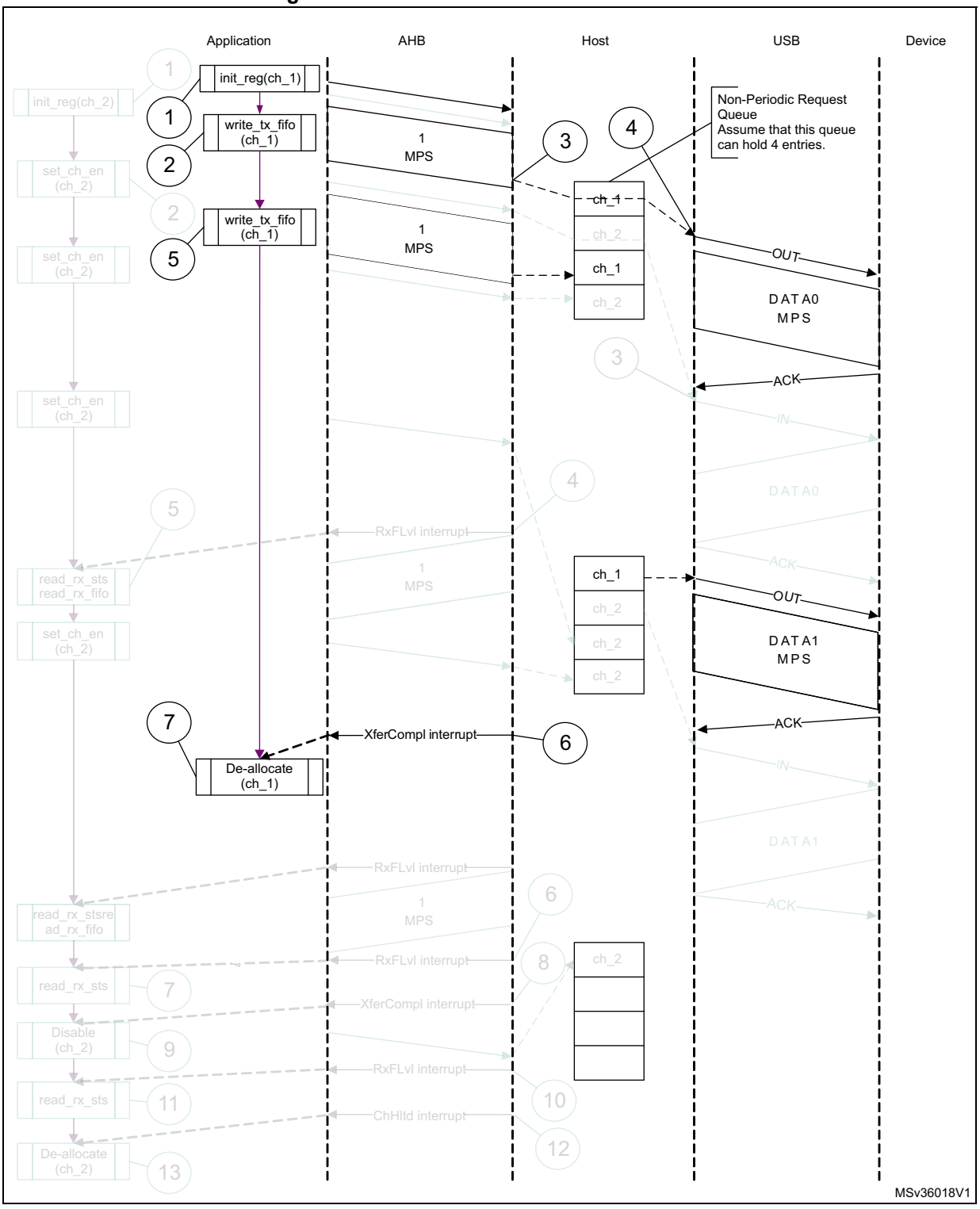
  – The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
  – The non-periodic transmit FIFO can hold two packets (1 KB for HS).
  – The non-periodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses

Figure 755. Normal bulk/control OUT/SETUP



1. The grayed elements are not relevant in the context of this figure.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

  a) Bulk/control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
```

```
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
```

```
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

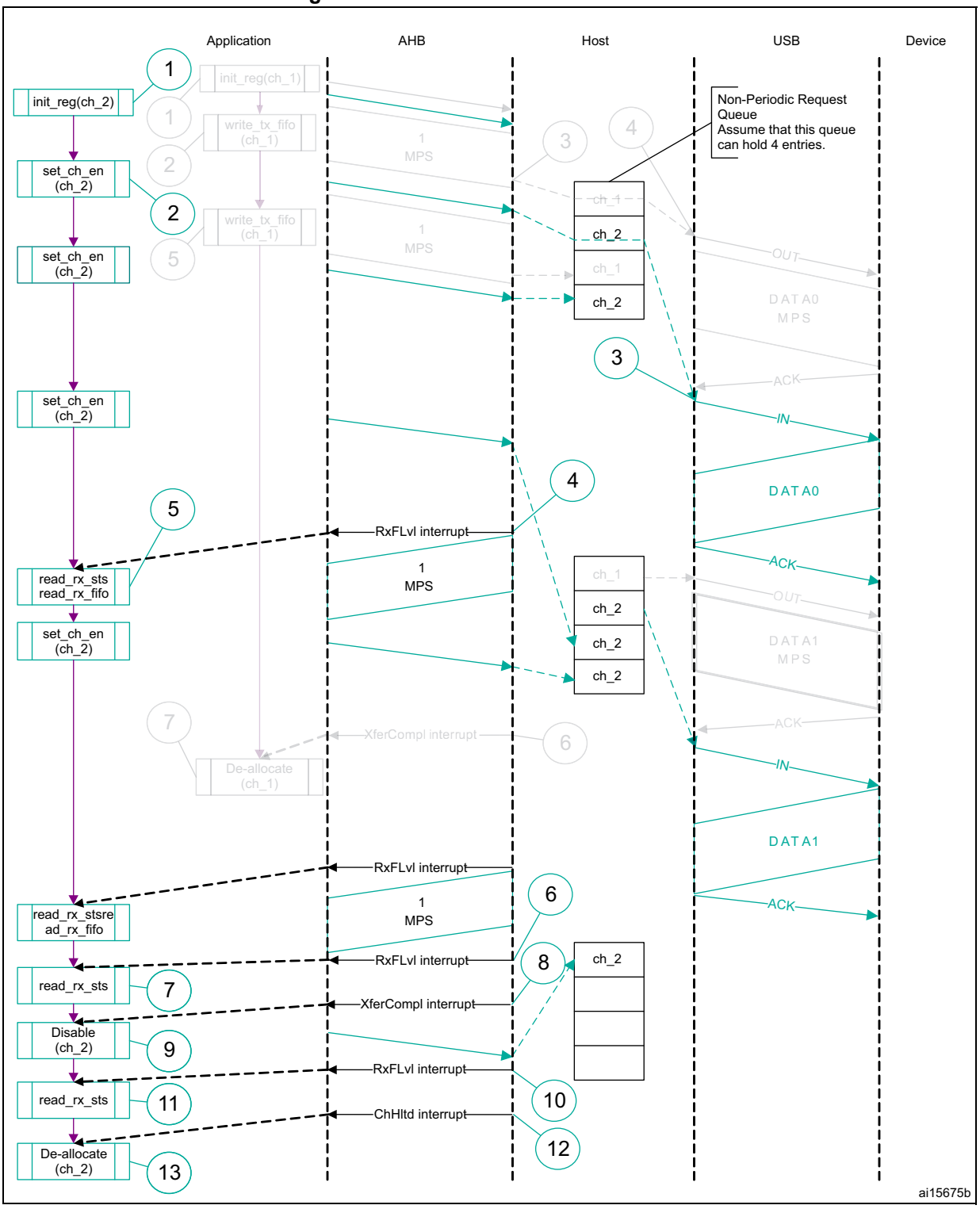
  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 756*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (520 bytes for HS).

  – The non-periodic request queue depth = 4.

Figure 756. Bulk/control IN transactions



1. The grayed elements are not relevant in the context of this figure.

The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

  Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 757*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

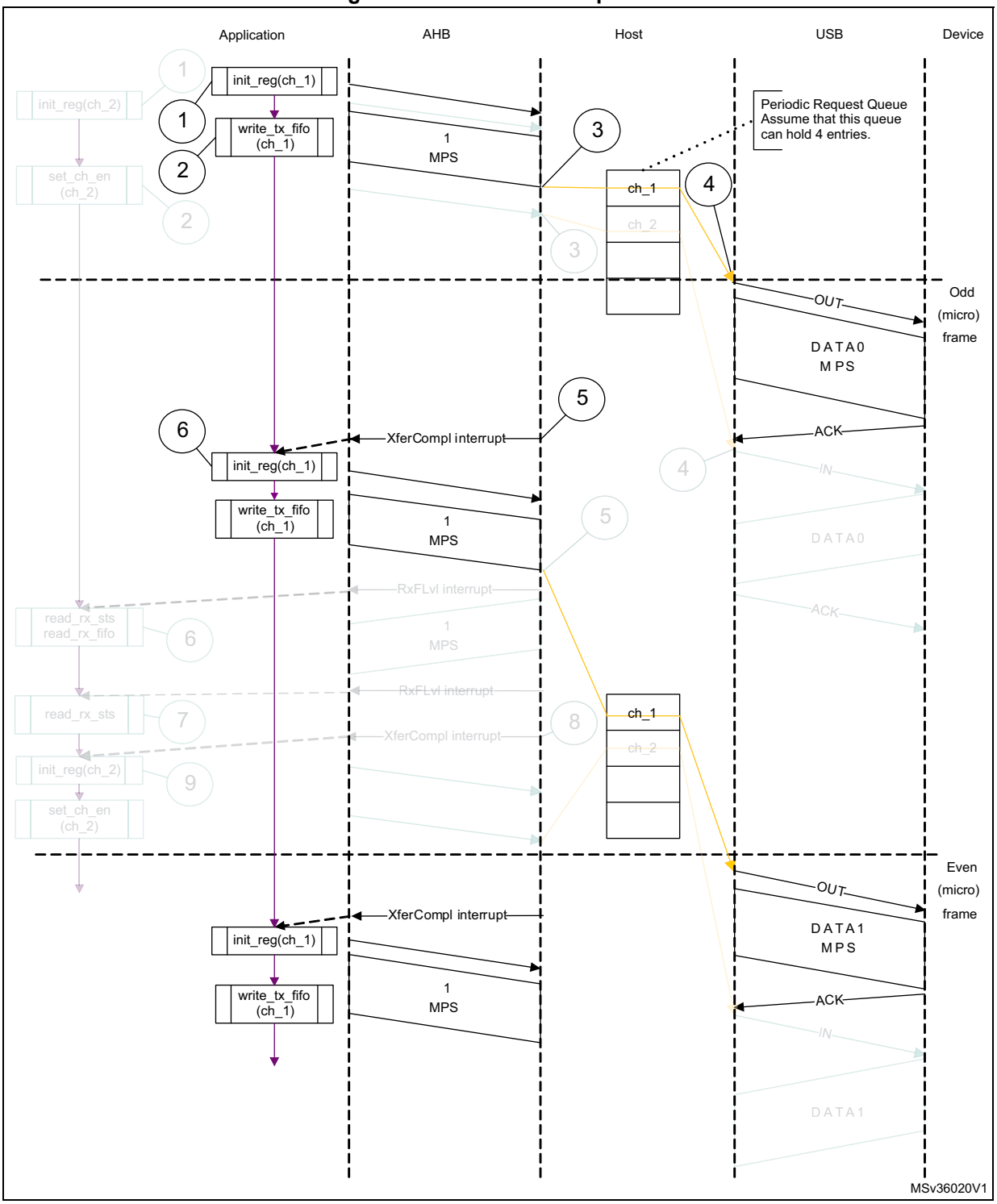
  – The periodic transmit FIFO can hold one packet (1 KB)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_HS host writes an entry to the periodic request queue.
4. The OTG_HS host attempts to send an OUT token in the next (odd) frame.
5. The OTG_HS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 757. Normal interrupt OUT



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**

  a) Interrupt OUT

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
```

```
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL)
         {
         Transfer Done = 1
         }
      }
   else
      if (NAK or TXERR)
         {
         Rewind Buffer Pointers
         Reset Error Count
         Mask ACK
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
            {
            De-allocate Channel
            }
         else
            {
            Re-initialize Channel (in next b_interval - 1 Frame)
            }
         }
      else
         if (ACK)
         {
         Reset Error Count
         Mask ACK
         }
```

```
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the
transmit FIFO space.
Interrupt IN
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
```

```
   if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
         {
         De-allocate Channel
         }
      else
         Re-initialize Channel (in next b_interval - 1 /Frame)
      }
   }
else
   if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **Interrupt IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).

  – Periodic request queue depth = 4.

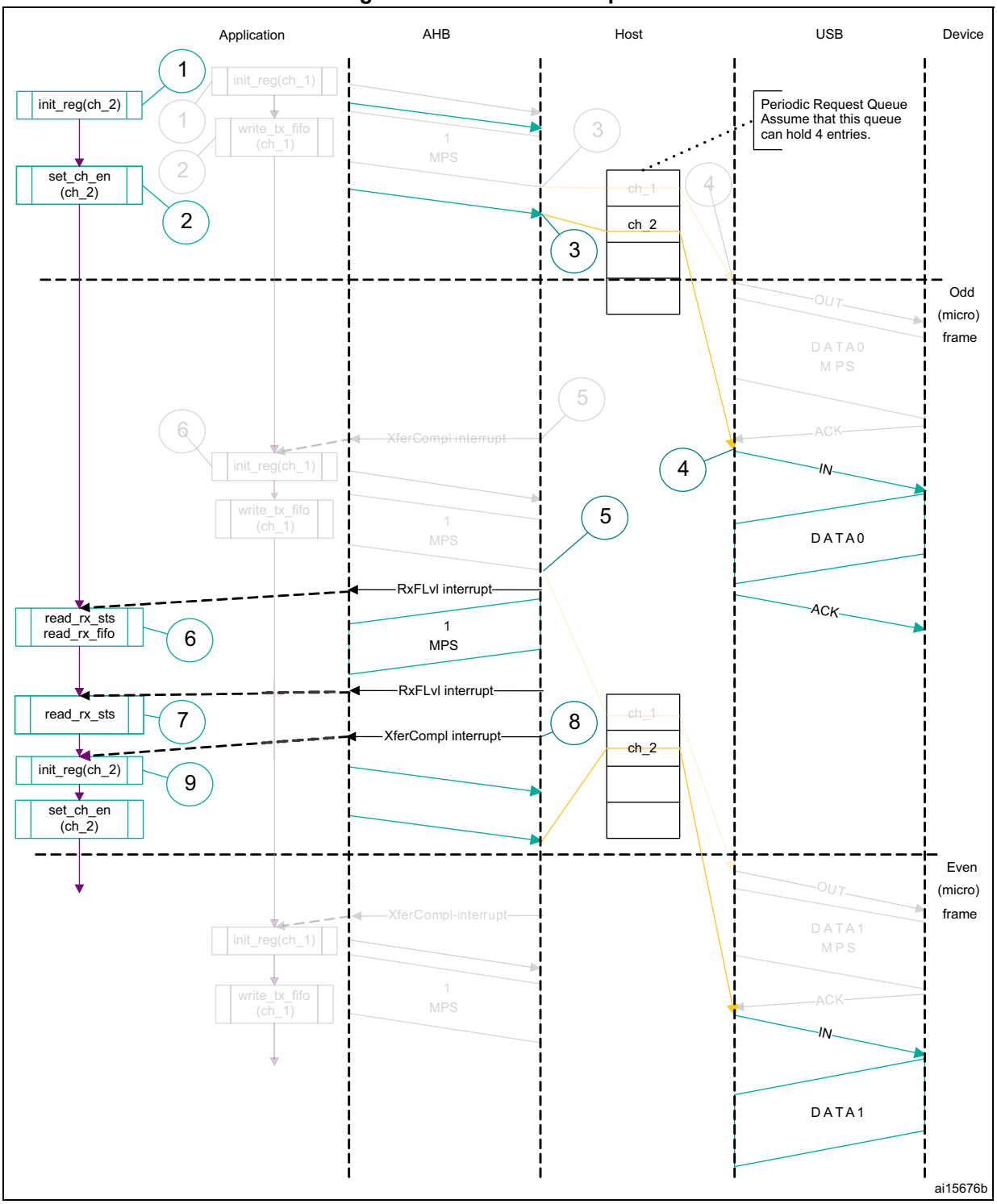
- **Normal interrupt IN operation**

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_HS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_HS host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-

initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 758. Normal interrupt IN



1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in *Figure 758*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum

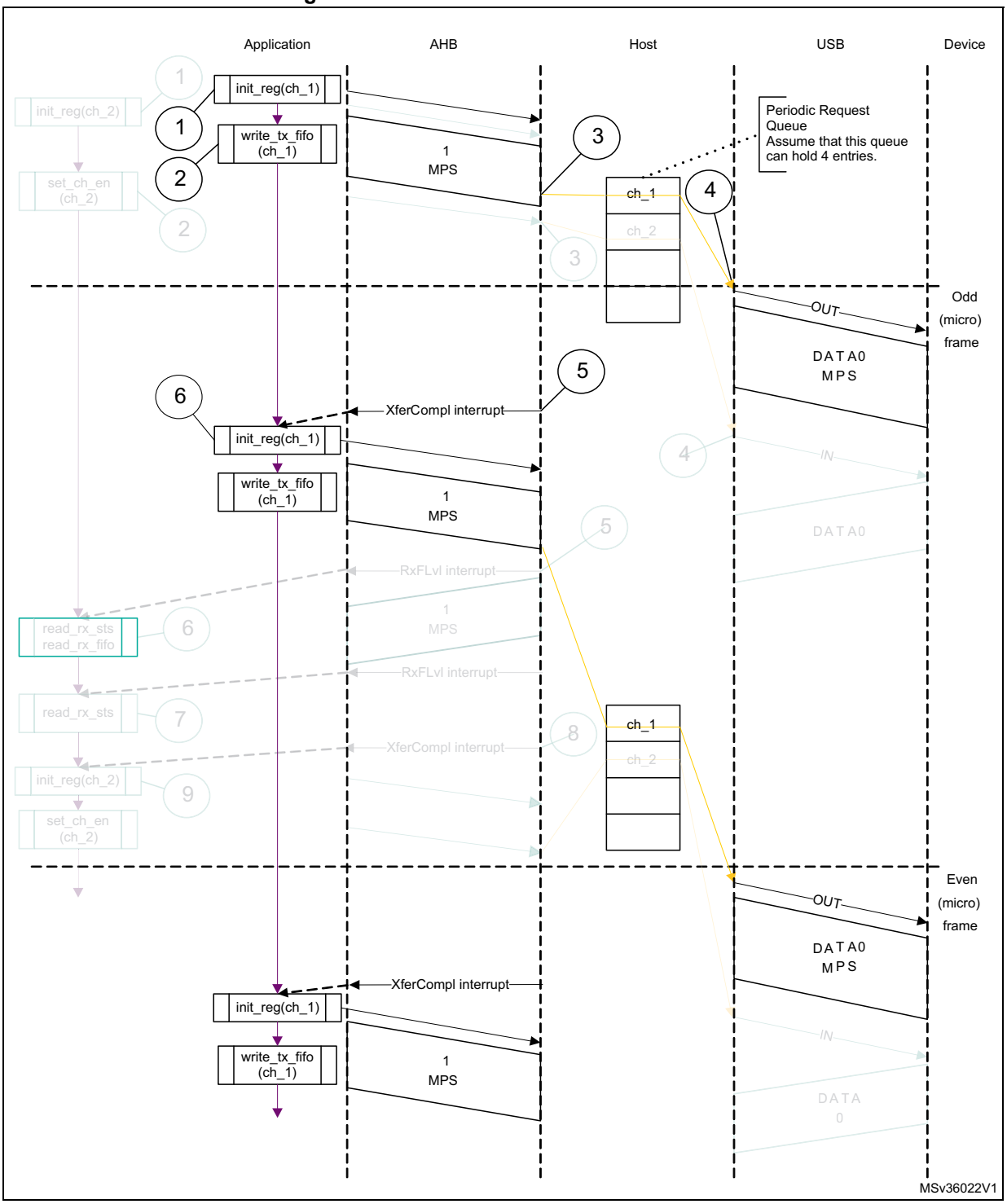
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

– The periodic transmit FIFO can hold one packet (1 KB).
– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_HS host writes an entry to the periodic request queue.
4. The OTG_HS host attempts to send the OUT token in the next frame (odd).
5. The OTG_HS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses

Figure 759. Isochronous OUT transactions



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
```

```
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
```

```
else
   {
   Re-initialize Channel
   }
}
```

- **Isochronous IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).

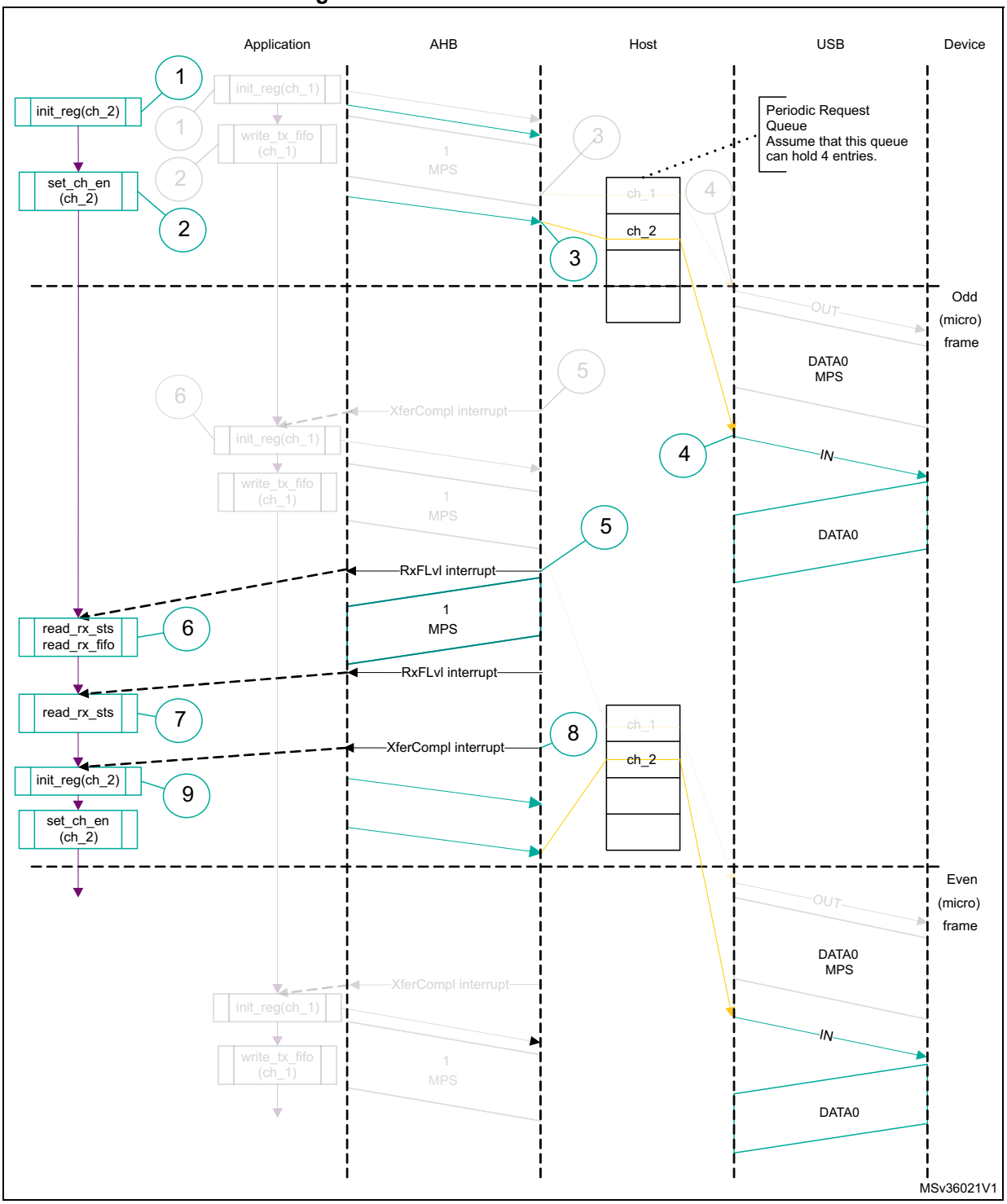
  – The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_HS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_HS host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 760. Isochronous IN transactions



1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic

transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG_HS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG_HS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

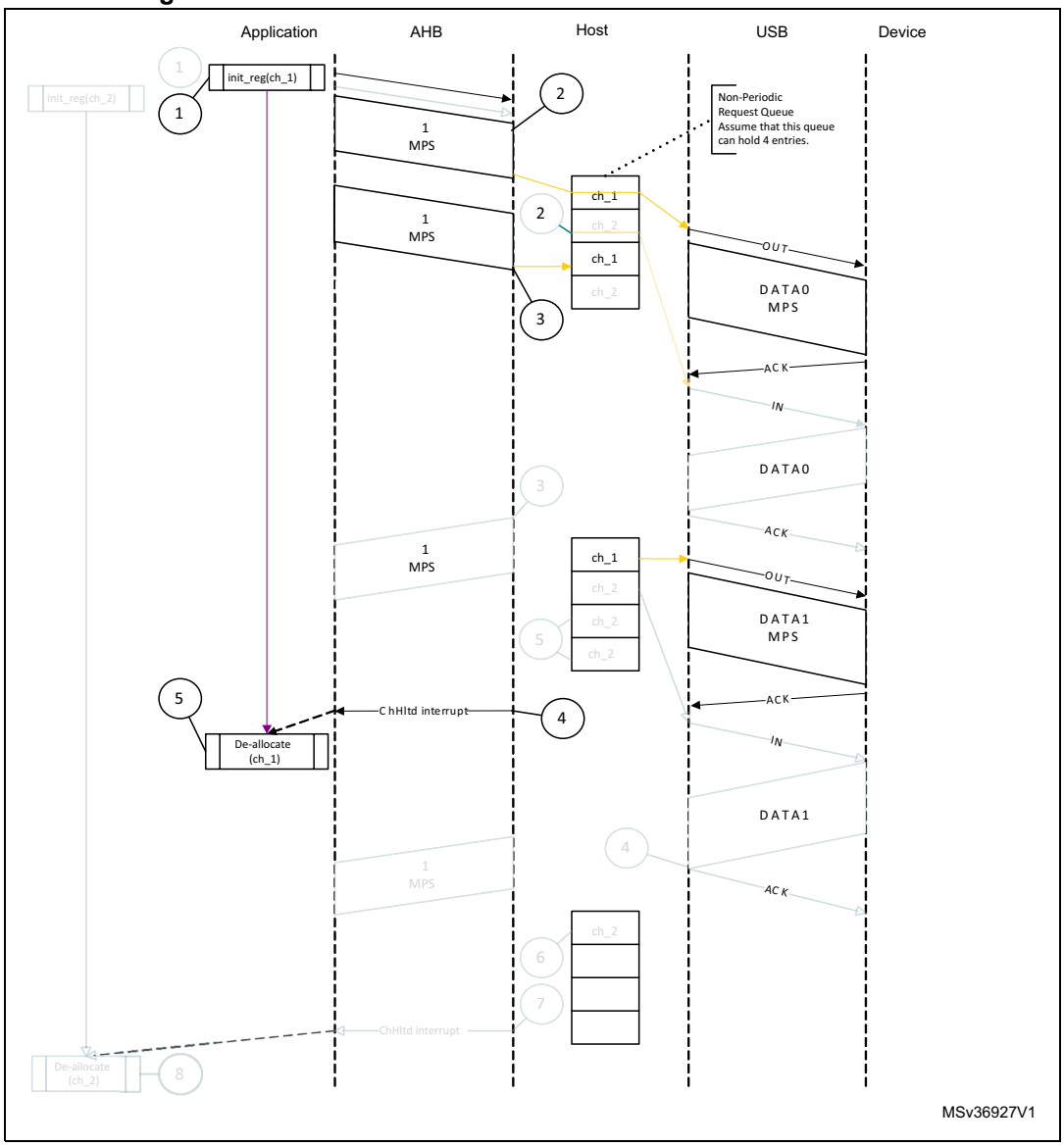
  When OTG_HS controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

- **Bulk and control OUT/SETUP transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable channel 1 as explained in *Section : Channel initialization*.
2. The OTG_HS host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the OTG_HS host uses the programmed DMA address to fetch the packet.
3. After fetching the last 32-bit word of the second (last) packet, the OTG_HS host masks channel 1 internally for further arbitration.
4. The OTG_HS host generates a CHH interrupt as soon as the last packet is sent.
5. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 761. Normal bulk/control OUT/SETUP transactions - DMA



- **NAK and NYET handling with internal DMA:**

1. The OTG_HS host sends a bulk OUT transaction.
2. The device responds with NAK or NYET.
3. If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application. The application is not required to service these interrupts, since the core takes care of rewinding the buffer pointers and re-initializing the Channel without application intervention.
4. The core automatically issues a ping token.
5. When the device returns an ACK, the core continues with the transfer. Optionally, the application can utilize these interrupts, in which case the NAK or NYET interrupt is masked by the application.

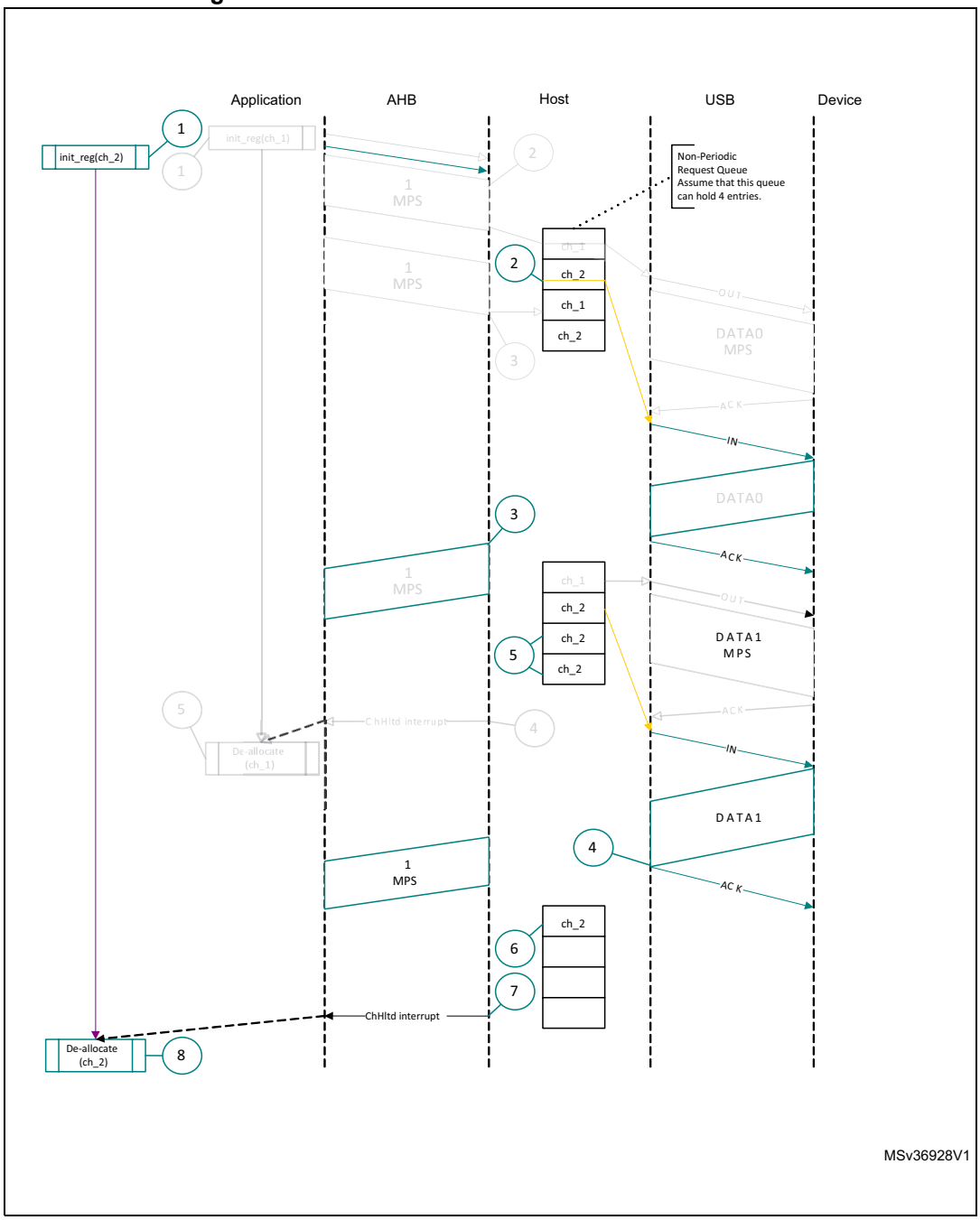
The core does not generate a separate interrupt when NAK or NYET is received by the host functionality.

- **Bulk and control IN transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable the used channel (channel x) as explained in *Section : Channel initialization*.
2. The OTG_HS host writes an IN request to the request queue as soon as the channel receives the grant from the arbiter (arbitration is performed in a round-robin fashion).
3. The OTG_HS host starts writing the received data to the system memory as soon as the last byte is received with no errors.
4. When the last packet is received, the OTG_HS host sets an internal flag to remove any extra IN requests from the request queue.
5. The OTG_HS host flushes the extra requests.
6. The final request to disable channel x is written to the request queue. At this point, channel 2 is internally masked for further arbitration.
7. The OTG_HS host generates the CHH interrupt as soon as the disable request comes to the top of the queue.
8. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 762. Normal bulk/control IN transaction - DMA

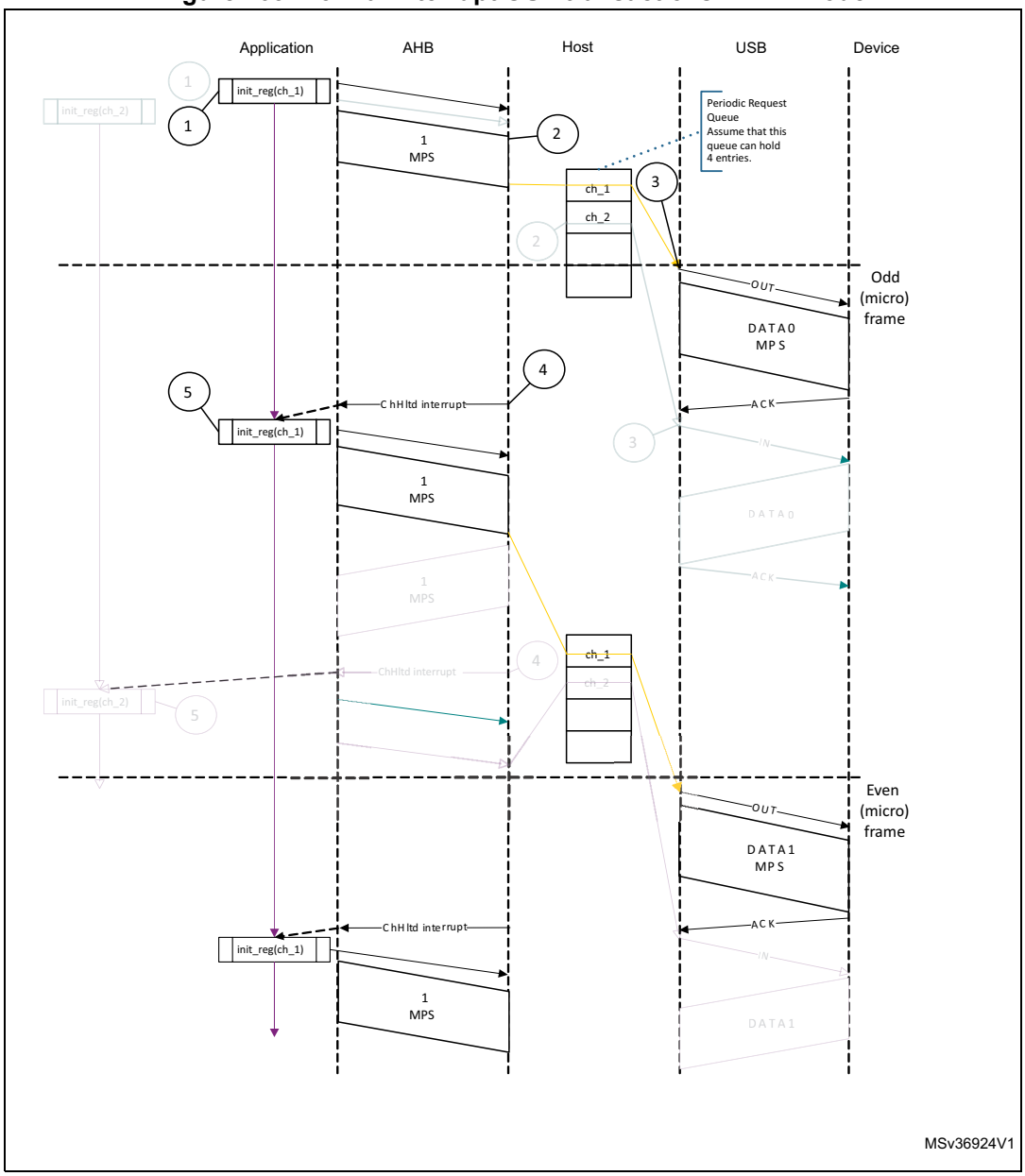


- **Interrupt OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth transfers, the OTG_HS host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The OTG_HS host attempts to send the OUT token at the beginning of the next odd frame/micro-frame.

4. After successfully transmitting the packet, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 763. Normal interrupt OUT transactions - DMA mode



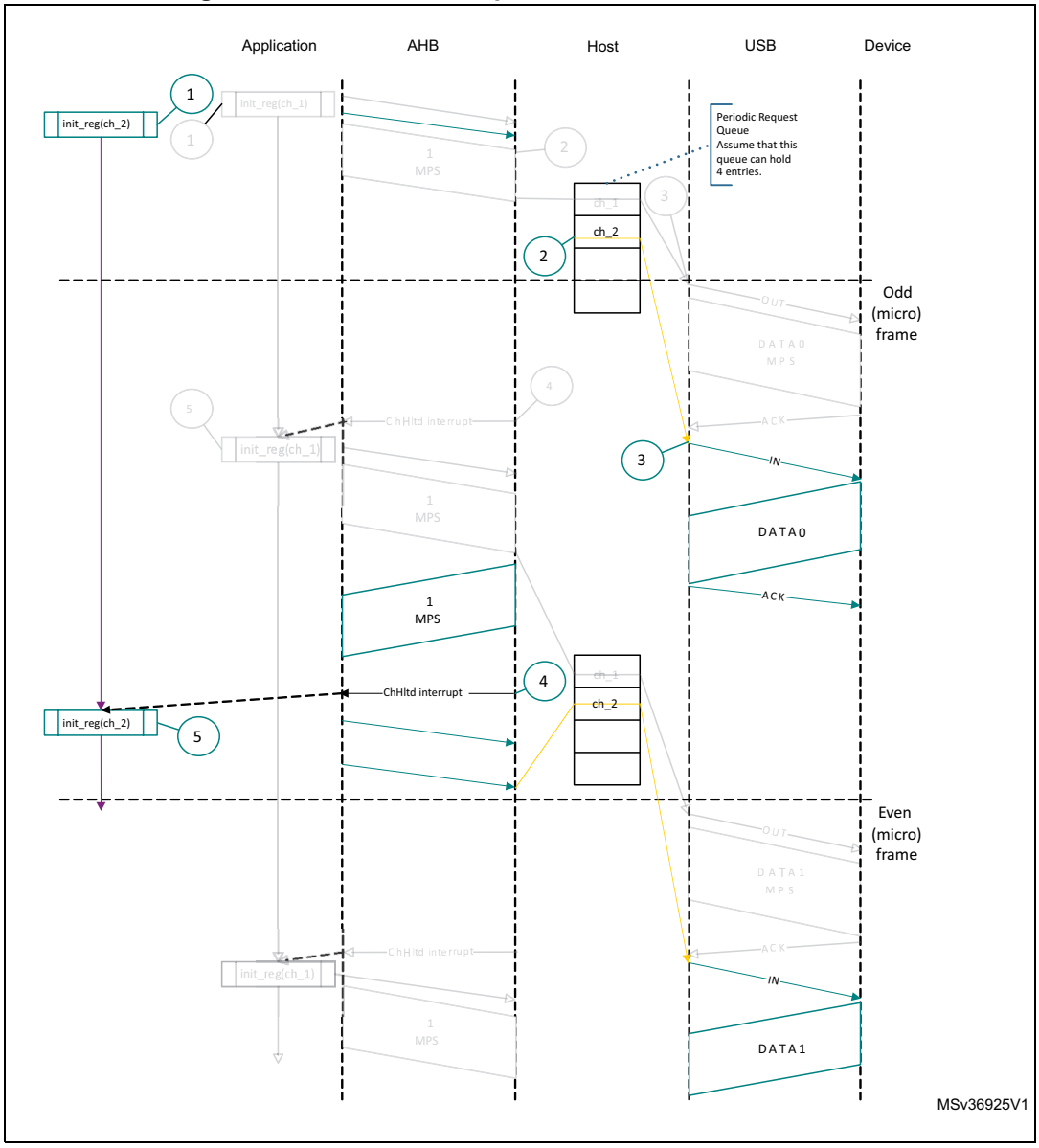
- **Interrupt IN transactions in DMA mode**

  The sequence of operations (channelx) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG_HS host writes consecutive writes up to MC times.

3. The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.
4. As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 764. Normal interrupt IN transactions - DMA mode



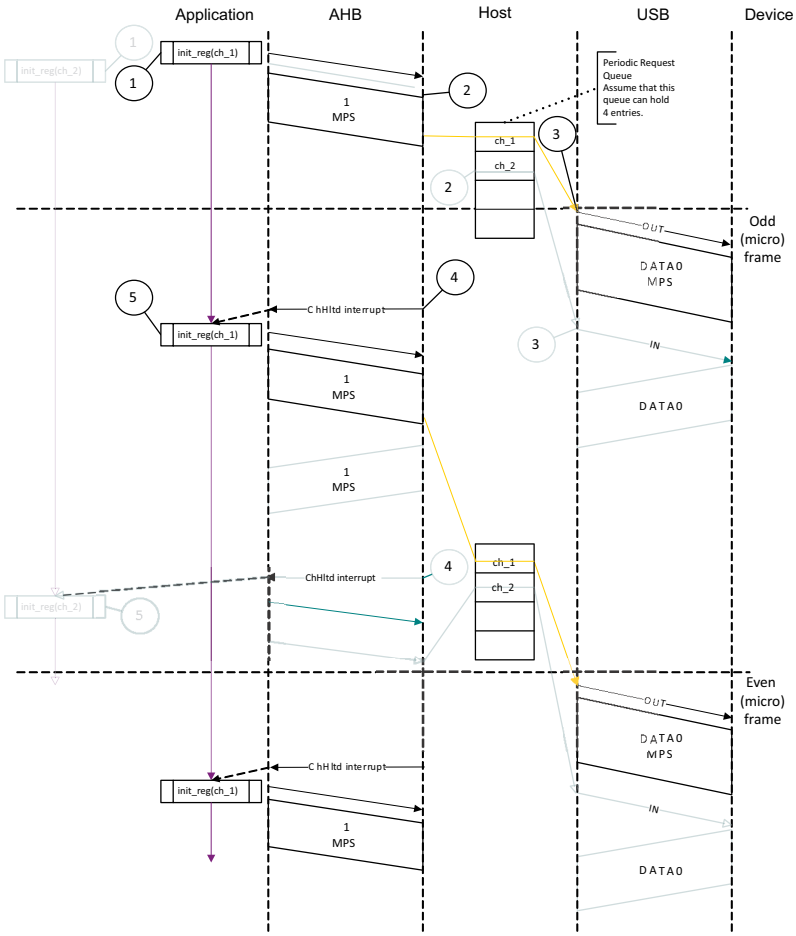
- **Isochronous OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG_HS host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth

transfers, the OTG_HS host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

3. The OTG_HS host attempts to send an OUT token at the beginning of the next (odd) frame/micro-frame.
4. After successfully transmitting the packet, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 765. Normal isochronous OUT transaction - DMA mode



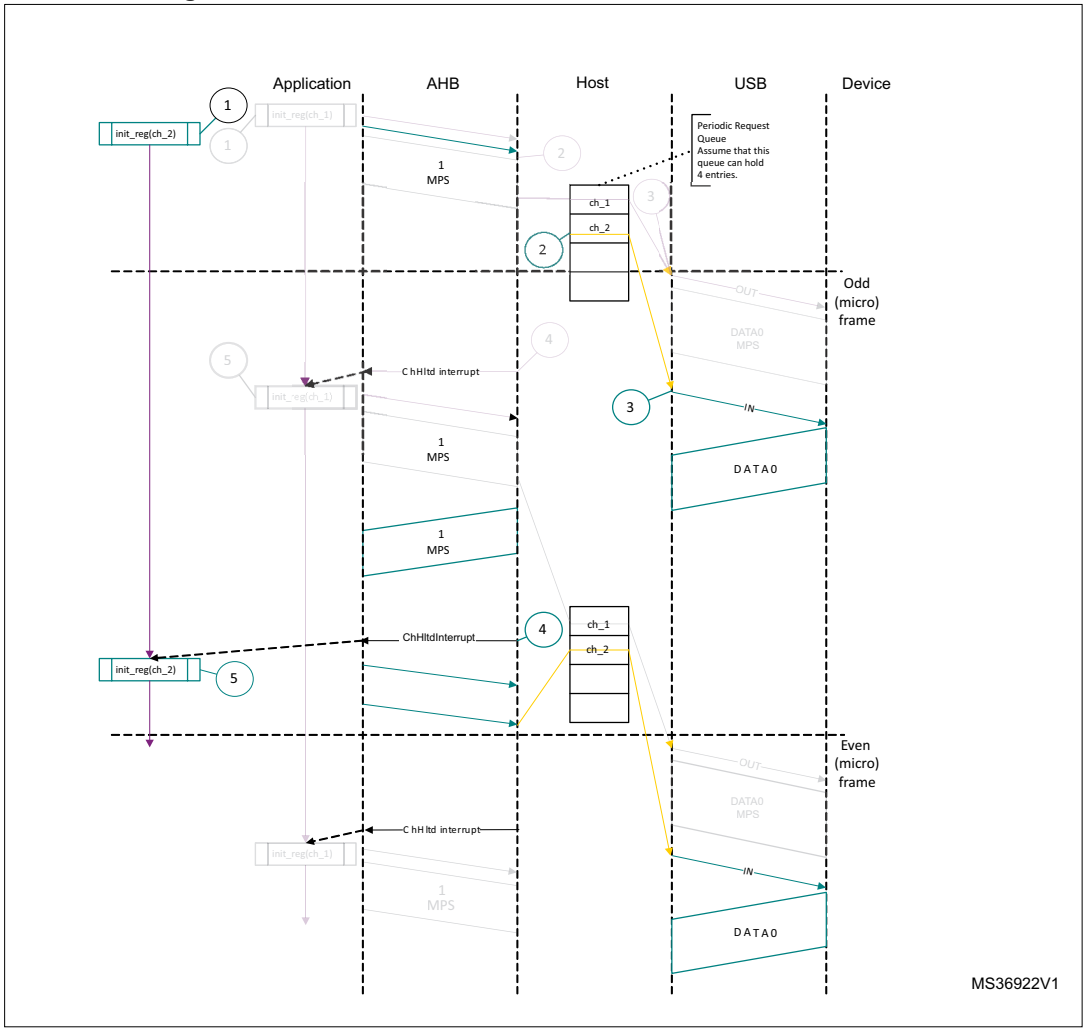
- **Isochronous IN transactions in DMA mode**

The sequence of operations ((channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG_HS host performs consecutive write operations up to MC times.

3. The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.
4. As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 766. Normal isochronous IN transactions - DMA mode



- **Bulk and control OUT/SETUP split transactions in DMA mode**

  The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch.
3. After successfully transmitting start split, the OTG_HS host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.

5. After successfully transmitting complete split, the OTG_HS host generates the CHH interrupt.
6. In response to the CHH interrupt, de-allocate the channel.

- **Bulk/control IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG_HS host writes the start split request to the nonperiodic request after getting the grant from the arbiter. The OTG_HS host masks the channel x internally for the arbitration after writing the request.
3. As soon as the IN token is transmitted, the OTG_HS host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 and re-enable the channel to send the complete split token. This unmasks channel x for arbitration.
5. The OTG_HS host writes the complete split request to the nonperiodic request after receiving the grant from the arbiter.
6. The OTG_HS host starts writing the packet to the system memory after receiving the packet successfully.
7. As soon as the received packet is written to the system memory, the OTG_HS host generates a CHH interrupt.
8. In response to the CHH interrupt, de-allocate the channel.

- **Interrupt OUT split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel 1 for start split as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. The OTG_HS host starts reading the packet.
3. The OTG_HS host attempts to send the start split transaction.
4. After successfully transmitting the start split, the OTG_HS host generates the CHH interrupt.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
6. After successfully completing the complete split transaction, the OTG_HS host generates the CHH interrupt.
7. In response to CHH interrupt, de-allocate the channel.

- **Interrupt IN split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.

6. As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory.
7. The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory.
8. In response to the CHH interrupt, de-allocate or reinitialize the channel for the next start split.

- **Isochronous OUT split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split (begin) as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1. Program the MPS field.
2. The OTG_HS host starts reading the packet.
3. After successfully transmitting the start split (begin), the OTG_HS host generates the CHH interrupt.
4. In response to the CHH interrupt, reinitialize the registers to send the start split (end).
5. After successfully transmitting the start split (end), the OTG_HS host generates a CHH interrupt.
6. In response to the CHH interrupt, de-allocate the channel.

- **Isochronous IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.
6. As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory.

   The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory. In response to the CHH interrupt, de-allocate the channel or reinitialize the channel for the next start split.

56.15.6 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   – SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   – INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   – OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   – STUPM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DIEPMSK
   – TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   – Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   – Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   – STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
5. For USB OTG_HS in DMA mode, the OTG_DOEPDMA0 register should have a valid memory address to store any SETUP packets received.

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. For USB OTG_HS in DMA mode, program the OTG_DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - Tx FIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.

Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

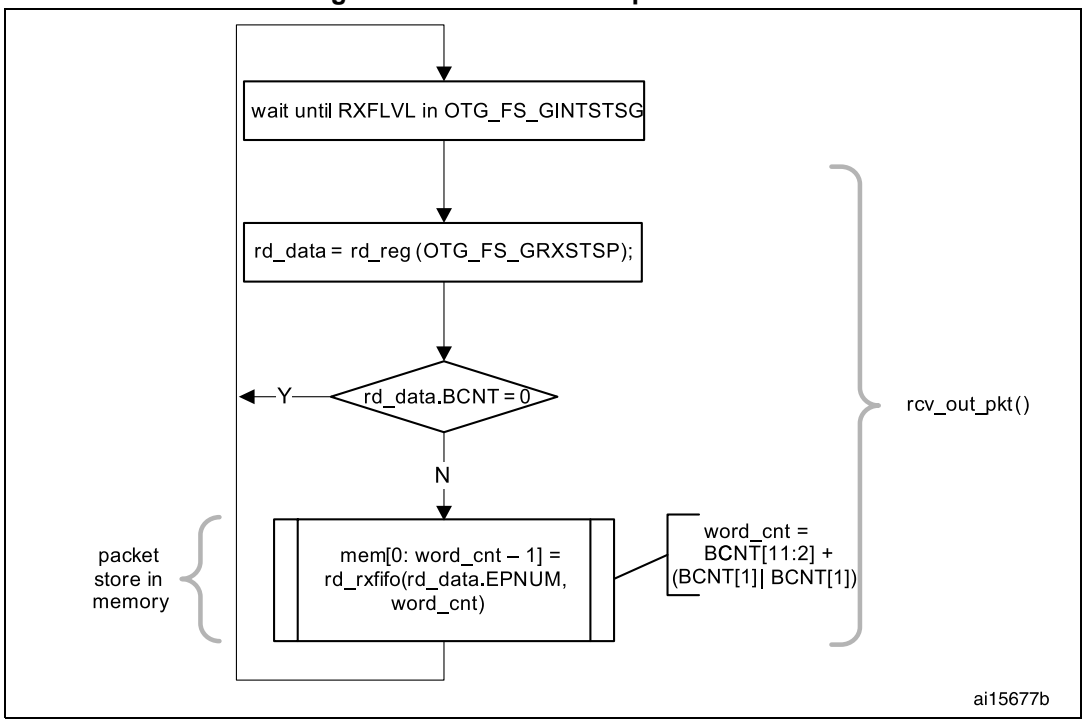
- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
   PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0),
   DPID = (0b00).
   These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num,
   DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   c) Setup stage done pattern:
   PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num,
   DPID = (0b00).
   These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.
   d) Data OUT packet pattern:
   PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
   e) Data transfer completed pattern:
   PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
   These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.
5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 767 provides a flowchart of the above procedure.

Figure 767. Receive FIFO packet read



SETUP transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
   – STUPCNT = 3 in OTG_DOEPTSIZx
2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
   – 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the

receive data FIFO to write SETUP data only, and never uses this space for data packets.

3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

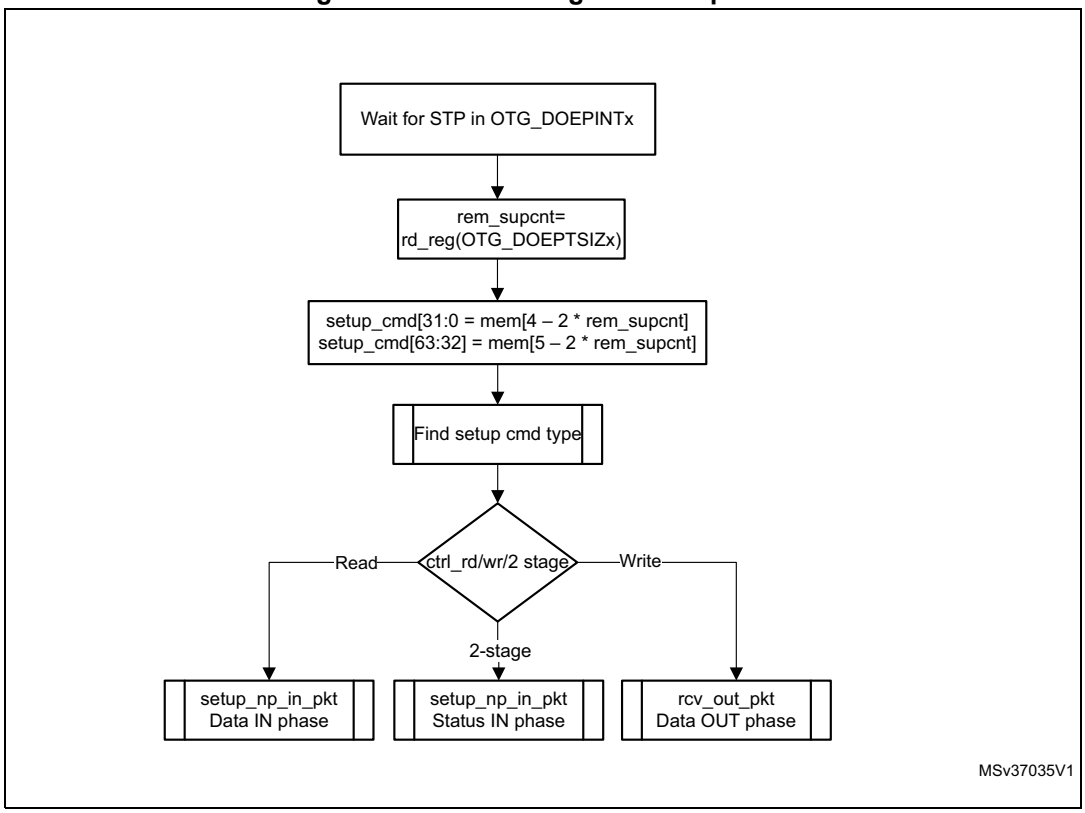
- **Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first word contains control information used internally by the core
   – The second word contains the first 4 bytes of the SETUP command
   – The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   – On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 768. Processing a SETUP packet



- **Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG_HS controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_DOEPCTLx
   – SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_DOEPCTLx
   – EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   – SGONAK = 0 in OTG_DCTL

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint's transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   – transfer size[EPNUM] = *n* × (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – packet count[EPNUM] = *n*
   – *n* > 0
3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core updated final transfer size
   – Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   – OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   – After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   – If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   – In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   – The transfer size is 0 and the packet count is 0
   – The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)

7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   – EPENA = 1 in OTG_DOEPCTLx
   – CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   – This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   – EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   – EPENA = 1
   – CNAK = 1
   – EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   – RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   – RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2

     RXDPID = D2 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 3The number of USB packets in which this payload was received =
     Application programmed initial packet count – core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application could fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:
   – When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   – When the isochronous OUT data packet is received with CRC errors
   – When the isochronous OUT token received by the core is corrupted
   – When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At

this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   – When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   – EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   – EPENA = 1 (in OTG_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.
6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   – Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).

   The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

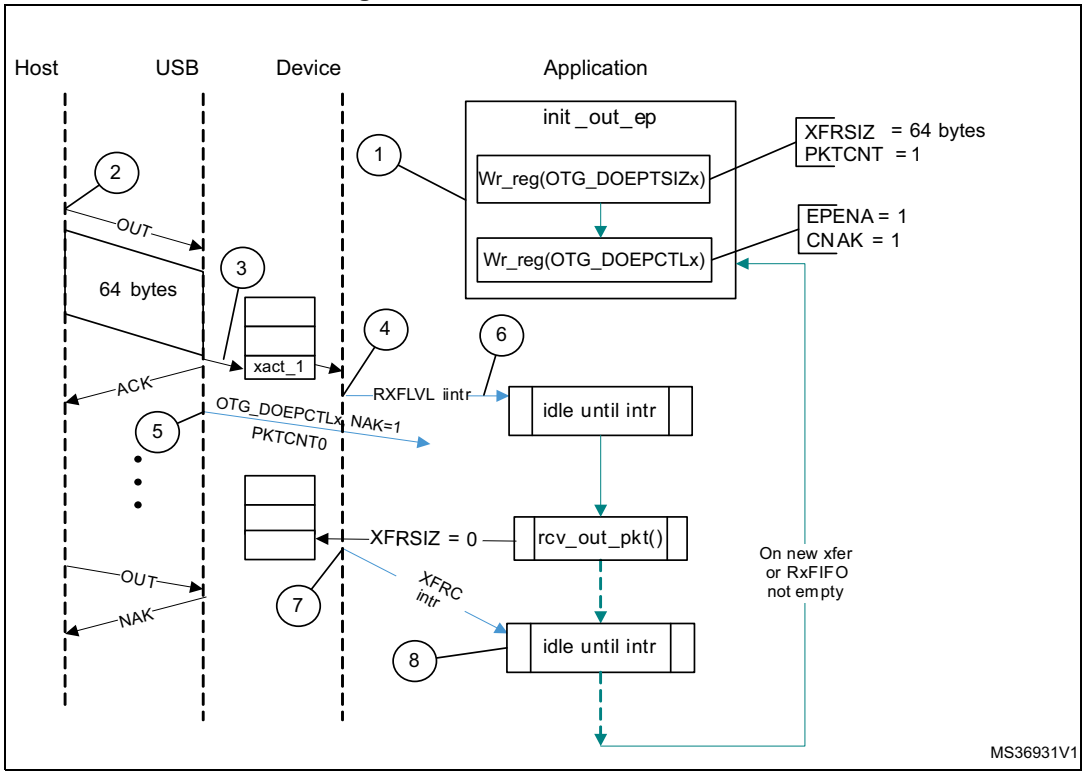
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

Figure 769 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

Figure 769. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   – In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   – To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   – Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   – INEPNEM = 0 in OTG_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).

– CNAK = 1 in OTG_DIEPCTLx

6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in OTG_DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_DIEPCTLx
   – SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_DIEPCTLx
   – EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   – TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   – To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:

     Transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp

     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = x
   – To transmit a single zero-length data packet:

Transfer size[EPNUM] = 0

Packet count[EPNUM] = 1

– To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.

  First transfer: transfer size[EPNUM] = *x* × MPSIZ[epnum]; packet count = *n*;

  Second transfer: transfer size[EPNUM] = 0; packet count = 1;

3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   – Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) × MPSIZ[EPNUM]
   – Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers on page 2724* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   - The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

     transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
     (where *x* is an integer ≥ 0, and 0 ≤ sp < MPSIZ[EPNUM])

     If (sp > 0), packet count[EPNUM] = *x* + 1
     Otherwise, packet count[EPNUM] = *x*;

     MCNT[EPNUM] = packet count[EPNUM]
   - The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
   - transfer size[EPNUM] = 0

     packet count[EPNUM] = 1

     MCNT[EPNUM] = packet count[EPNUM]
2. The application can only schedule data transfers one frame at a time.
   - (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   - PKTCNT = MCNT (in OTG_DIEPTSIZx)
   - If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   - Note that: MCNT is in OTG_DIEPTSIZx, MPSIZ is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and XFERSIZ is in OTG_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   - A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   - A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.
   – A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   – A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
   – For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   – For interrupt endpoints, when an ACK handshake is transmitted
   – When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

      The core transmits a zero-length data packet on the USB in response to the received IN token.
2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   – SNAK = 1 in OTG_DIEPCTLx
   – EPDIS = 1 in OTG_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   – At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt in OTG_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

56.15.7 Worst case response time

When the OTG_HS controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

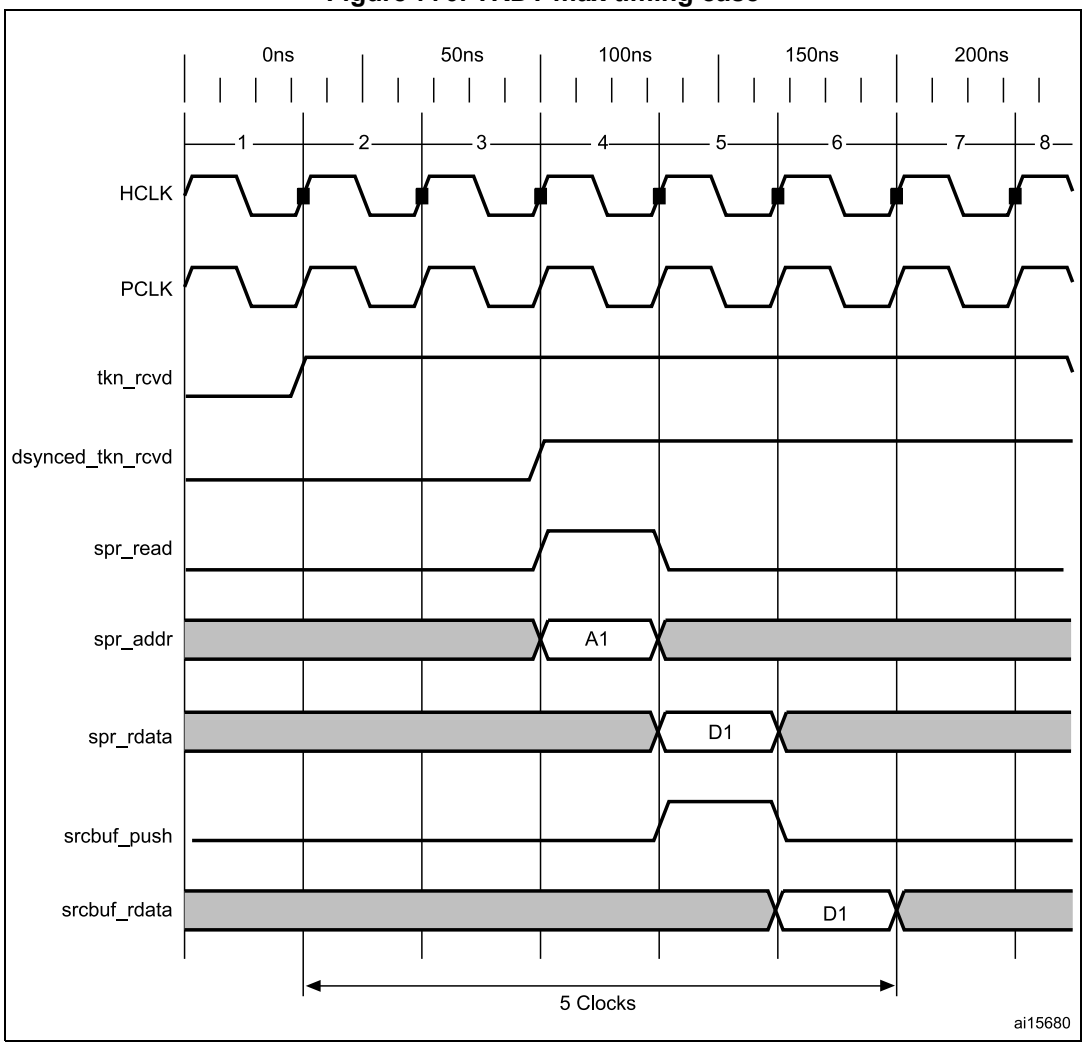
If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

Figure 770 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to *Table 499: TRDT values (HS)*.

Figure 770. TRDT max timing case



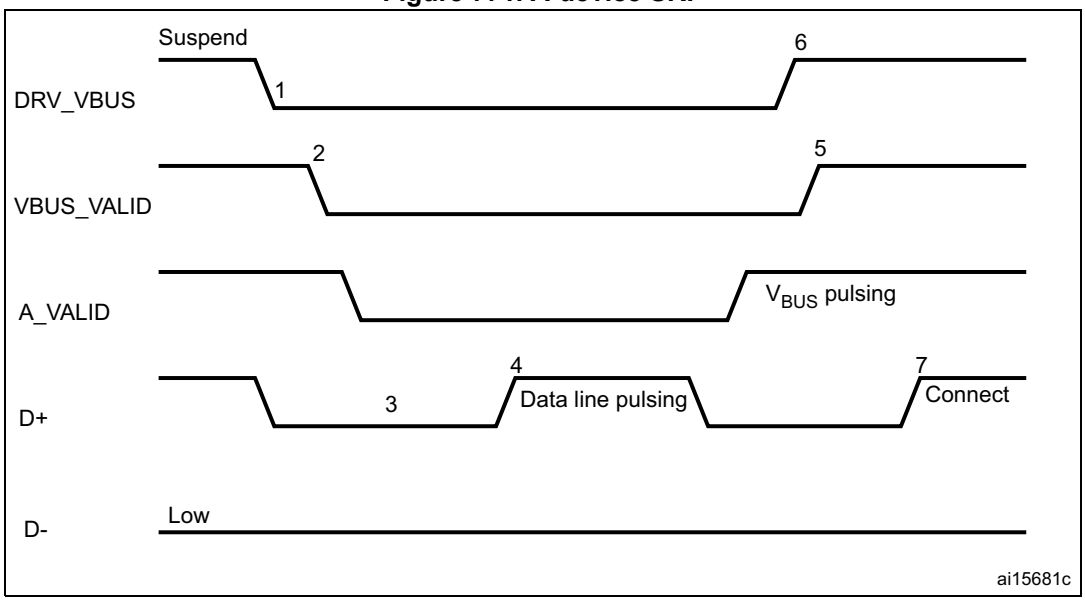
56.15.8 OTG programming model

The OTG_HS controller is an OTG device supporting HNP and SRP. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device. In host mode, the OTG_HS controller turns off V_{BUS} to conserve power. SRP is a method by which the B-device signals the A-device to turn on V_{BUS} power. A device must perform both data-line pulsing and V_{BUS} pulsing, but a host can detect either data-line pulsing or V_{BUS} pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

A-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG_HS controller to detect SRP as an A-device.

Figure 771. A-device SRP



1. DRV_VBUS = V_{BUS} drive signal to the PHY
   VBUS_VALID = V_{BUS} valid signal from PHY
   A_VALID = A-peripheral V_{BUS} level signal to PHY
   D+ = Data plus line
   D- = Data minus line

The following points refer and describe the signal numeration shown in the *Figure 771*:

1. To save power, the application suspends and turns off port power when the bus is idle by writing the port suspend and port power bits in the host port control and status register.
2. PHY indicates port power off by deasserting the VBUS_VALID signal.
3. The device must detect SE0 for at least 2 ms to start SRP when V_{BUS} power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The OTG_HS controller detects data-line pulsing.
5. The device drives V_{BUS} above the A-device session valid (2.0 V minimum) for V_{BUS} pulsing.
   The OTG_HS controller interrupts the application on detecting SRP. The session

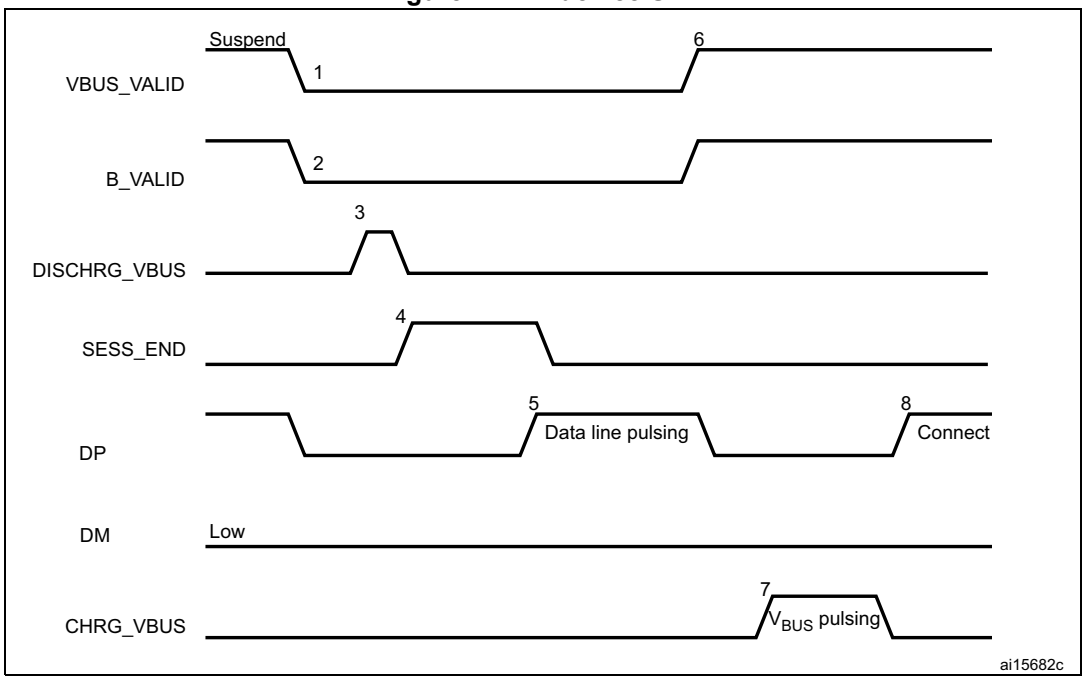
request detected bit is set in Global interrupt status register (SRQINT set in OTG_GINTSTS).

6. The application must service the session request detected interrupt and turn on the port power bit by writing the port power bit in the host port control and status register. The PHY indicates port power-on by asserting the VBUS_VALID signal.
7. When the USB is powered, the device connects, completing the SRP process.

B-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG_HS controller to initiate SRP as a B-device. SRP is a means by which the OTG_HS controller can request a new session from the host.

Figure 772. B-device SRP



1. VBUS_VALID = V_{BUS} valid signal from PHY
B_VALID = B-peripheral valid session to PHY
DISCHRG_VBUS = discharge signal to PHY
SESS_END = session end signal to PHY
CHRG_VBUS = charge V_{BUS} signal to PHY
DP = Data plus line
DM = Data minus line

The following points refer and describe the signal numeration shown in the *Figure 772*:

1. To save power, the host suspends and turns off port power when the bus is idle.
The OTG_HS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the core interrupt register.
The OTG_HS controller informs the PHY to discharge V_{BUS}.
2. The PHY indicates the session's end to the device. This is the initial condition for SRP.
The OTG_HS controller requires 2 ms of SE0 before initiating SRP.
For a USB 1.1 full-speed serial transceiver, the application must wait until V_{BUS} discharges to 0.2 V after BSVLD (in OTG_GOTGCTL) is deasserted. This discharge

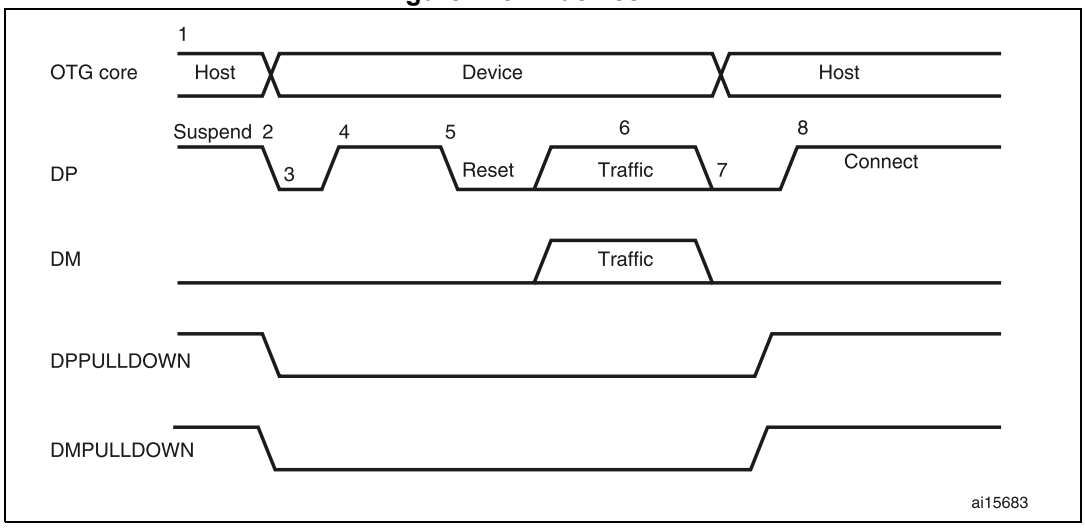
time can be obtained from the transceiver vendor and varies from one transceiver to another.

3. The OTG_HS core informs the PHY to speed up V_{BUS} discharge.
4. The application initiates SRP by writing the session request bit in the OTG control and status register. The OTG_HS controller perform data-line pulsing followed by V_{BUS} pulsing.
5. The host detects SRP from either the data-line or V_{BUS} pulsing, and turns on V_{BUS}. The PHY indicates V_{BUS} power-on to the device.
6. The OTG_HS controller performs V_{BUS} pulsing.
   The host starts a new session by turning on V_{BUS}, indicating SRP success. The OTG_HS controller interrupts the application by setting the session request success status change bit in the OTG interrupt status register. The application reads the session request success bit in the OTG control and status register.
7. When the USB is powered, the OTG_HS controller connects, completing the SRP process.

A-device host negotiation protocol

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG_HS controller to perform HNP as an A-device.

Figure 773. A-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.
   DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the *Figure 773*:

1. The OTG_HS controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set host Set HNP enable bit in the OTG control

and status register to indicate to the OTG_HS controller that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the port suspend bit in the host port control and status register.
3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

   The OTG_HS controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.

   The OTG_HS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_DP pull-up resistor to indicate a connect for B-device.

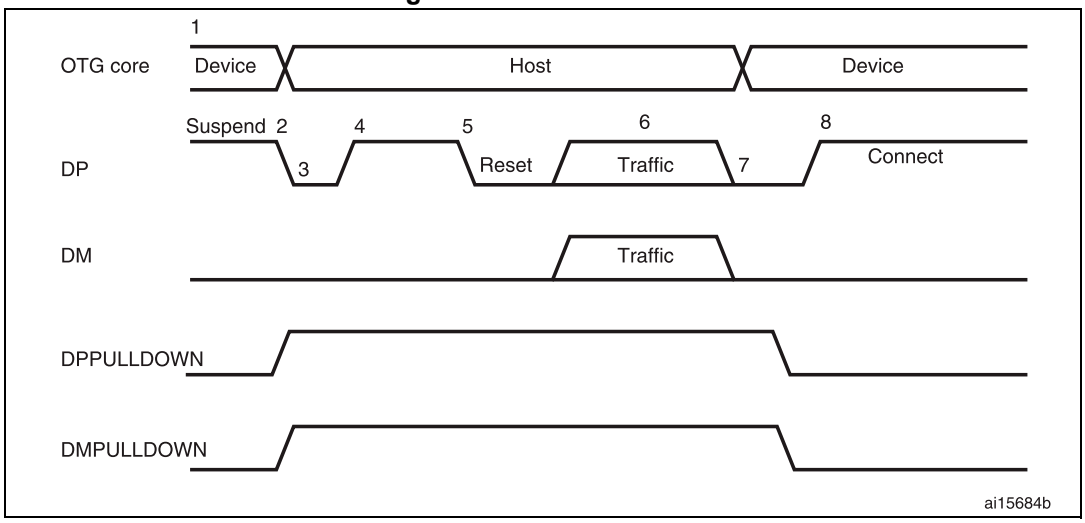
   The application must read the current mode bit in the OTG control and status register to determine device mode operation.
4. The B-device detects the connection, issues a USB reset, and enumerates the OTG_HS controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

   The OTG_HS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the core interrupt register.
6. In Negotiated mode, the OTG_HS controller detects the suspend, disconnects, and switches back to the host role. The OTG_HS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG_HS controller sets the connector ID status change interrupt in the OTG interrupt status register. The application must read the connector ID status in the OTG control and status register to determine the OTG_HS controller operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

B-device host negotiation protocol

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG_HS controller to perform HNP as a B-device.

Figure 774. B-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY. DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the *Figure 774*:

1. The A-device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The OTG_HS controller's ACK response indicates that it supports HNP. The application must set the device HNP enable bit in the OTG control and status register to indicate HNP support.

   The application sets the HNP request bit in the OTG control and status register to indicate to the OTG_HS controller to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the port suspend bit in the host port control and status register.

   The OTG_HS controller sets the Early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the core interrupt register.

   The OTG_HS controller disconnects and the A-device detects SE0 on the bus, indicating HNP. The OTG_HS controller asserts the DP pull down and DM pull down in the PHY to indicate its assumption of the host role.

   The A-device responds by activating its OTG_DP pull-up resistor within 3 ms of detecting SE0. The OTG_HS controller detects this as a connect.

   The OTG_HS controller sets the host negotiation success status change interrupt in the OTG interrupt status register, indicating the HNP status. The application must read the host negotiation success bit in the OTG control and status register to determine

host negotiation success. The application must read the current Mode bit in the core interrupt register (OTG_GINTSTS) to determine host mode operation.

3. The application sets the reset bit (PRST in OTG_HPRT) and the OTG_HS controller issues a USB reset and enumerates the A-device for data traffic.
4. The OTG_HS controller continues the host role of initiating traffic, and when done, suspends the bus by writing the port suspend bit in the host port control and status register.
5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The OTG_HS controller deasserts the DP pull down and DM pull down in the PHY to indicate the assumption of the device role.
6. The application must read the current mode bit in the core interrupt (OTG_GINTSTS) register to determine the host mode operation.
7. The OTG_HS controller connects, completing the HNP process.

57 Ethernet (ETH): media access control (MAC) with DMA controller

57.1 Ethernet introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The Ethernet peripheral enables the devices to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2002 standard.

The Ethernet provides a configurable, flexible peripheral to meet the needs of various applications and customers. It supports two industry standard interfaces to the external physical layer (PHY): the default media independent interface (MII) defined in the IEEE 802.3 specifications and the reduced media independent interface (RMII). It can be used in number of applications such as switches and network interface cards.

In addition to the default interfaces defined in the IEEE 802.3 specifications, the Ethernet peripheral supports several industry standard interfaces to the PHY. It is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC, Media Independent Interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Audio Video (AV) traffic
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master and AHB slave ports
- RMII specification version 1.2 from RMII consortium

57.2 Ethernet main features

Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

57.2.1 MAC core features

Interfaces

- Separate transmission, reception, and control interfaces to the application
- 32-bit data transfer interface on the application side
- 10, 100 data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII (default) interface to communicate with an external Fast Ethernet PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management
- Supports mandatory network statistics with RMON counters (RFC2819/RFC2665)

Main operations

- Support of both full-duplex and half-duplex operations:
  - CSMA/CD protocol for half-duplex operation
  - IEEE 802.3x flow control for full-duplex operation
  - Optional forwarding of received pause control frames to the user application in full-duplex operation
  - Back-pressure in half-duplex operation
  - Automatic transmission of zero-quanta pause frame on deassertion of flow control input in full-duplex operation
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Preamble and start-of-frame data (SFD) insertion in Transmit mode, and deletion in Receive paths
- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or up to 16 Kbyte Jumbo Ethernet packets
- Programmable Inter Packet Gap
- Layer 3/Layer 4 checksum offload for received packets
- Calculation and insertion of IPv4 header checksum and TCP, UDP, or ICMP checksum in frames transmitted in Store-and-Forward mode
- Two sets of FIFOs: a 2048-byte Transmit FIFO with programmable threshold capability, and a 2048-byte Receive FIFO with a configurable threshold
- Store-and-Forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Programmable Rx queue threshold (or cut-through) mode
- Internal loopback from Tx to Rx on MII for debugging.

VLAN management

- Source Address field insertion or replacement, as well as VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement or deletion of up to two VLAN tags
- IEEE 802.1Q VLAN tag detection and possibility to delete the VLAN tags in received packets
- Stripping of up to two VLAN tags and providing the tags in the status.

Packet filtering

- Flexible address filtering modes:
  - 3 additional 48-bit perfect destination address (DA) filters with masks for each byte
  - 3 additional 48-bit source address (SA) comparison check with masks for each byte
  - 64 bit Hash filter for multicast and unicast (DA) addresses
  - Option to pass all multicast addressed packets
  - Promiscuous mode to pass all packets without any filtering for network monitoring
  - Pass all incoming packets (as per filter) with a status report
- Additional packet filtering:
  - VLAN tag-based: perfect match and Hash-based filtering with filtering based either on the outer or inner VLAN tag
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6

IEEE 1588-2008/PTPv2

- Ethernet packet time-stamping as described in IEEE 1588-2002 and IEEE 1588-2008 specifications (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping is supported in Tx direction.
- Flexibility to control the Pulse-Per-Second (PPS) output signal (ptp_pps_o)

Low-power modes

- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII PHYs.
- Module to detect remote wakeup packets and AMD Magic packets

57.2.2 DMA features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see *Section 57.11.2: Ethernet DMA registers*) to control the DMA operations. The DMA block supports the following features:

- 32-bit data transfers
- Separate DMA in Transmit path and receive paths
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host control (AHB) access and host data interface
- Tx DMA channel with TCP Segmentation Offload (TSO) feature enabled

57.2.3 Bus interface features

AHB master interface

The AHB master interface features as the following:

- Interfaces with the application through AHB
- Little-endian mode
- 32-bit data on the AHB master port
- Split, Retry, and Error AHB responses
- AHB 1K boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)

The AHB master interface does not generate the following:

- Wrap burst
- Locked or protected transfers

AHB slave interface

The AHB slave interface supports the following features:

- Interfaces with the application through AHB
- Little-endian mode
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:

- Split
- Retry
- Error

57.3 Ethernet pins and internal signals

Table 502 lists the Ethernet inputs and output signals connected to package pins or balls, while *Table 503* shows the internal Ethernet signals.

Table 502. Ethernet peripheral pins

Pin name	Alternate function name (mapped on AF11)
PA0	ETH_MII_CRS
PA1	ETH_MII_RX_CLK/ETH_RMII_REF_CLK
PA2	ETH_MDIO
PA3	ETH_MII_COL
PA7	ETH_MII_RX_DV/ETH_RMII_CRS_DV
PA9	ETH_TX_ER
PB0	ETH_MII_RXD2
PB1	ETH_MII_RXD3
PB2	ETH_TX_ER
PB5	ETH_PPS_OUT
PB8	ETH_MII_TXD3
PB10	ETH_MII_RX_ER
PB11	ETH_MII_TX_EN/ETH_RMII_TX_EN
PB12	ETH_MII_TXD0/ETH_RMII_TXD0
PB13	ETH_MII_TXD1/ETH_RMII_TXD1
PC1	ETH_MDC
PC2	ETH_MII_TXD2
PC3	ETH_MII_TX_CLK
PC4	ETH_MII_RXD0/ETH_RMII_RXD0
PC5	ETH_MII_RXD1/ETH_RMII_RXD1
PE2	ETH_MII_TXD3
PG8	ETH_PPS_OUT
PG11	ETH_MII_TX_EN/ETH_RMII_TX_EN

Table 502. Ethernet peripheral pins (continued)

Pin name	Alternate function name (mapped on AF11)
PG12	ETH_MII_TXD1/ETH_RMII_TXD1
PG13	ETH_MII_TXD0/ETH_RMII_TXD0
PG14	ETH_MII_TXD1/ETH_RMII_TXD1
PH2	ETH_MII_CRS
PH3	ETH_MII_COL
PH6	ETH_MII_RXD2
PH7	ETH_MII_RXD3
PI10	ETH_MII_RX_ER
PI12	ETH_TX_ER

Table 503. Ethernet internal input/output signals

Signal name	Signal type	Description
eth_hclk	Digital input	AHB clock
eth_sbd_intr_it	Digital output	Main Ethernet interrupt
lpi_intr_o	Digital output	Sideband signal generated when the transmitter or receiver enters or exits the LPI state.
pmt_intr_o	Digital output	Sideband signal generated when a valid remote wakeup packet is received
eth_mii_tx_clk	Digital input	MII Tx kernel clock
eth_mii_rx_clk	Digital input	MII Rx kernel clock
eth_rmii_ref_clk	Digital input	RMII reference kernel clock

57.4 Ethernet architecture

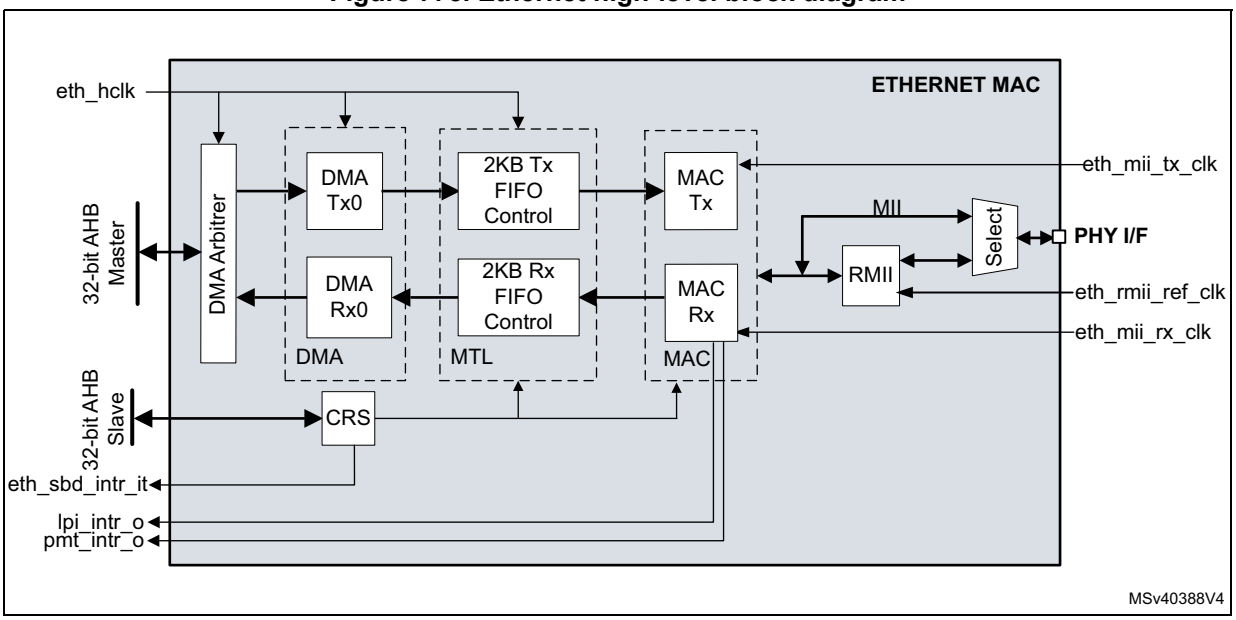
The Ethernet peripheral is composed of 4 main functional modules:

- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)

  This is the logical DMA module with one physical channel for reception and 1 for transmission. It controls the data transfers between MAC and system memory through the AMBA AHB 32-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The mac transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII PHY Media Independent Interfaces.

Figure 775. Ethernet high-level block diagram



1. For a definition of the internal signals, refer to *Table 503*.
2. Refer to RCC chapter "Clock distribution for Ethernet" for a detailed description of the Ethernet clock architecture.

57.4.1 DMA controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The base address of each list is written to the respective Tx and Rx registers: *Channel Tx descriptor list address register (ETH_DMACTXDLAR)*) and *Channel Rx descriptor list address register (ETH_DMACRXDLAR)*).

The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, *Channel Tx descriptor ring length register (ETH_DMACTXRLR)* and *Channel Rx descriptor ring length register (ETH_DMACRXRLR)*).

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

Descriptors are specified in *Section 57.10: Descriptors*.

DMA arbitration

The DMA module incorporates an arbiter that performs the arbitration between the Tx and Rx channels accesses from the AHB master interface. The following two types of arbitrations are supported and can be selected through *DMA mode register (ETH_DMAMR)*:

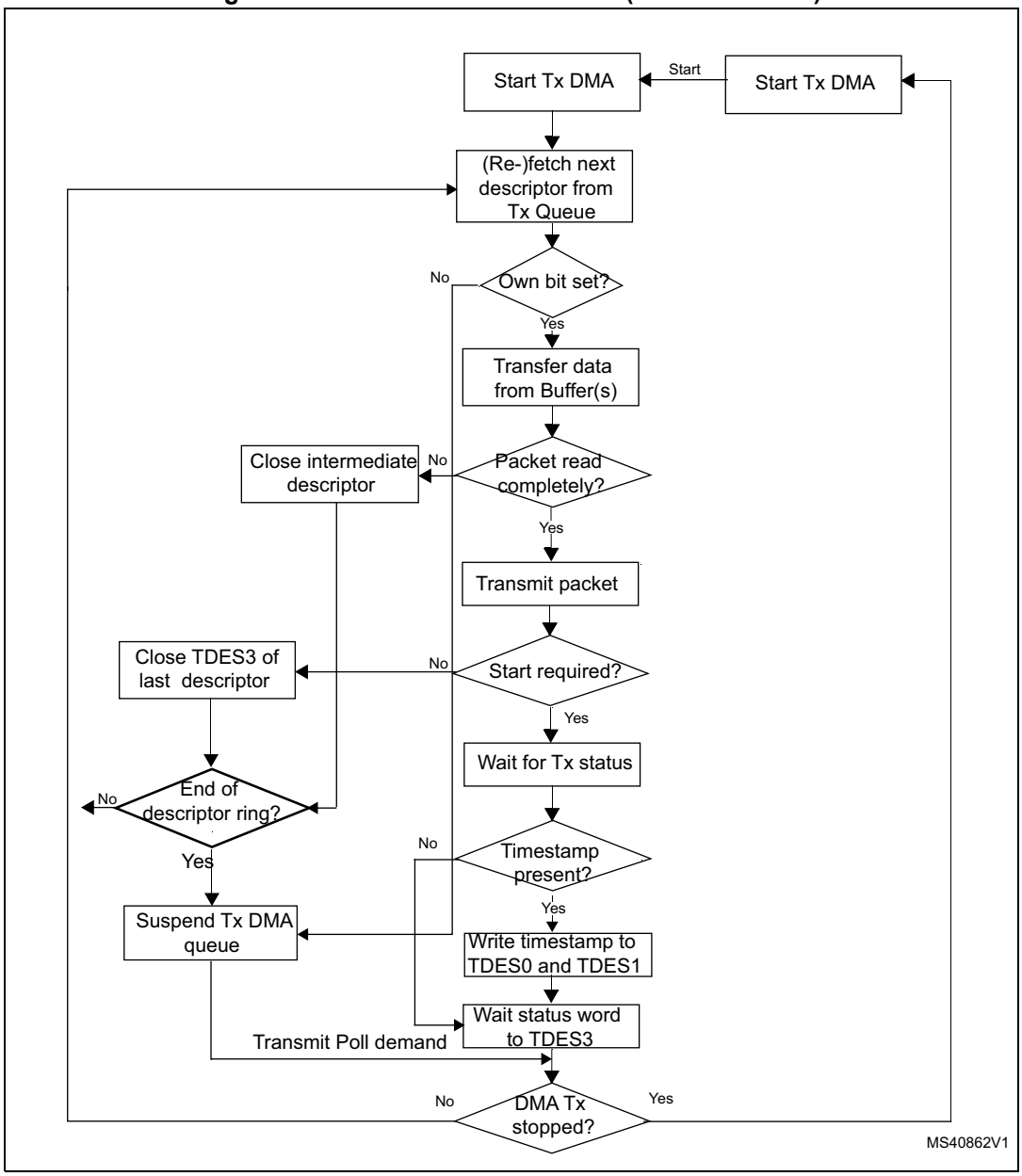
- Round-robin arbitration**:** the arbiter allocates the data bus between Rx and Tx in ratio set by Bits [14:12] of ETH_DMAMR.
- Fixed-priority arbitration**:** by default Rx DMA always gets priority over Tx DMA for data access. Setting bit 11 of ETH_DMAMR register gives priority to the Tx DMA.

DMA transmission in default mode

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application shifts the Descriptor tail pointer offset value of the Transmit channel.
3. The DMA fetches the descriptor from the application memory.
4. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   - The descriptor is flagged as owned by the application (TDES3 [31] = 0).
   - The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   - An error condition occurs.
5. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
6. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
7. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.
8. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.
9. Bit 0 of *Channel status register (ETH_DMACSR)*) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
10. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the *Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)* when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.

Figure 776. DMA transmission flow (standard mode)



DMA transmission in OSP (Operate on Second Packet) mode

In Run state, if bit 4 is set in the *Channel transmit control register (ETH_DMACTXCR)*), the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see *Section : DMA transmission in default mode*).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:

   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.

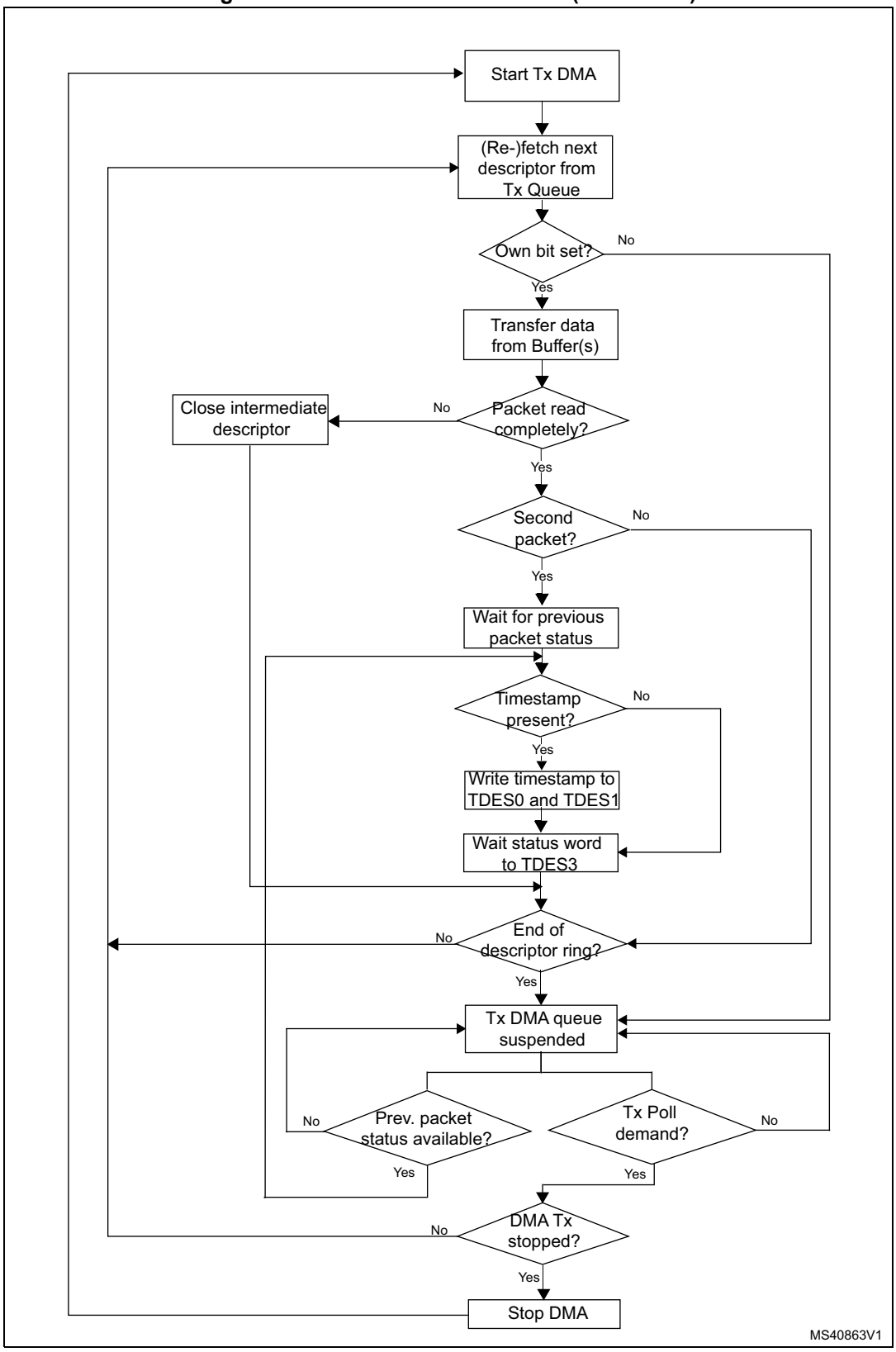
   a) The DMA writes the status to the corresponding TDES3.

   a) The DMA sets the relevant interrupts and returns to Suspend mode.

   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in *Figure 778: Receive DMA flow*.

Figure 777. DMA transmission flow (OSP mode)



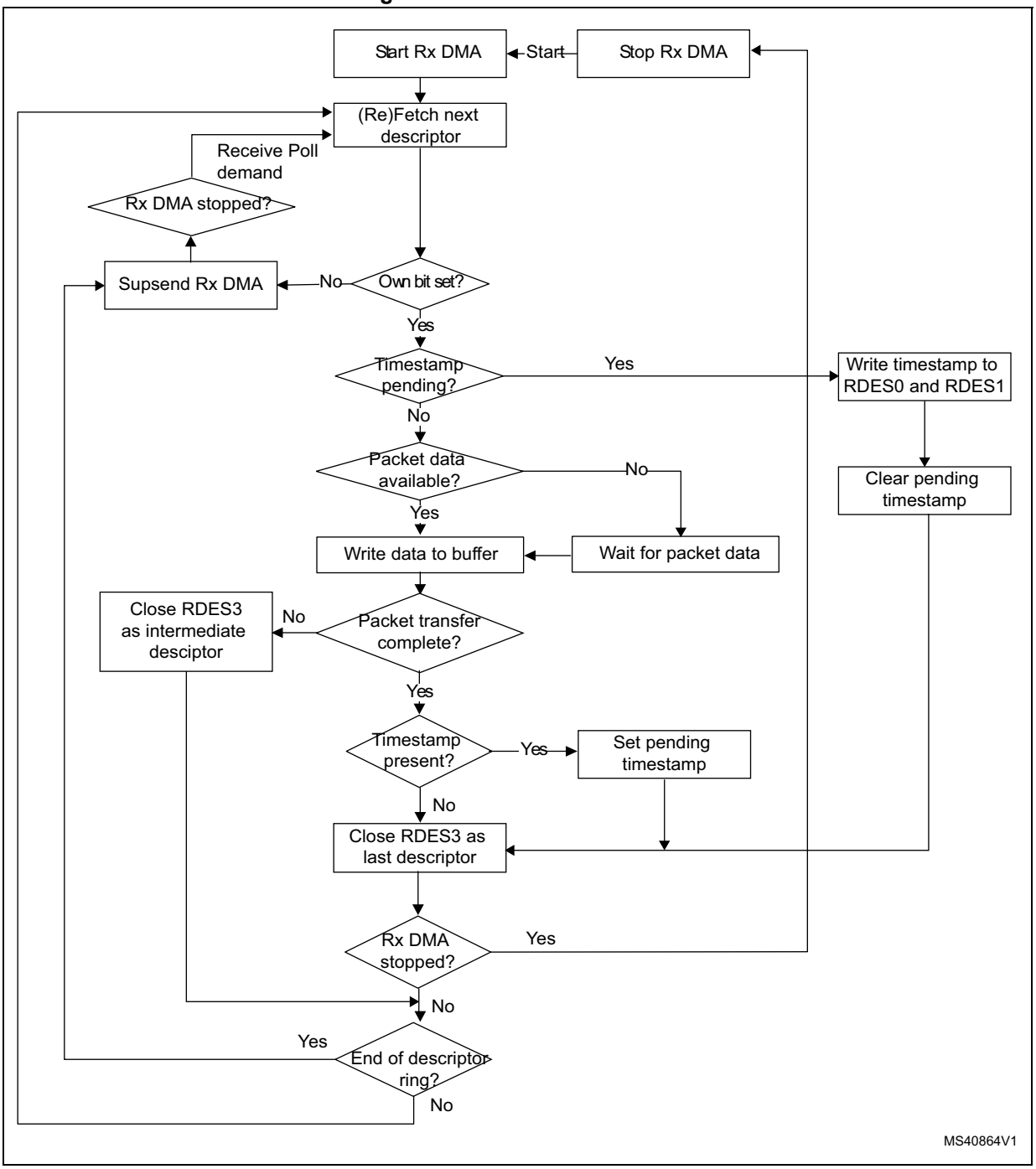
DMA reception

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

The reception sequence for Rx DMA engine is as follows (see also *Figure 778: Receive DMA flow*):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application must set the correct value in the Receive descriptor tail pointer register of corresponding DMA channel.
2. When bit 0 of *Channel receive control register (ETH_DMACRXCR)*) is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).
5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA retrieves the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.
8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. If IEEE 1588 timestamp feature is enabled, the DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).
11. The Receive DMA exits the Suspend state when a Receive Poll demand is given and the application increments the channel Receive tail pointer register. The engine proceeds to step 2 and fetches again the next descriptor.

Figure 778. Receive DMA flow



57.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 32-bit wide data paths: the Transmit path and the Receive Path.

- Transmit path

The application or internal DMA pushes the Ethernet packets read from the application or system memory into the Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 2048 bytes.

- Receive path

  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured Receive threshold (RTC bits[1:0] defined in *Rx queue operating mode register (ETH_MTLRXQOMR)*), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 2048 bytes.

57.4.3 MAC

The MAC is responsible of the Ethernet protocol processing.In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

MAC transmission

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   - The MAC completes the normal transmission and provides the transmission status to the MTL.
   - If a normal collision (in half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:

     The packet was successfully transmitted

     The maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) in observed from the MAC.
   - If any one of the following happens, the MAC aborts the packet transmission:

     No carrier (half-duplex mode)

     Loss of carrier (half-duplex mode)

     Excessive deferral (half-duplex mode)

     Late collisions (half-duplex mode)

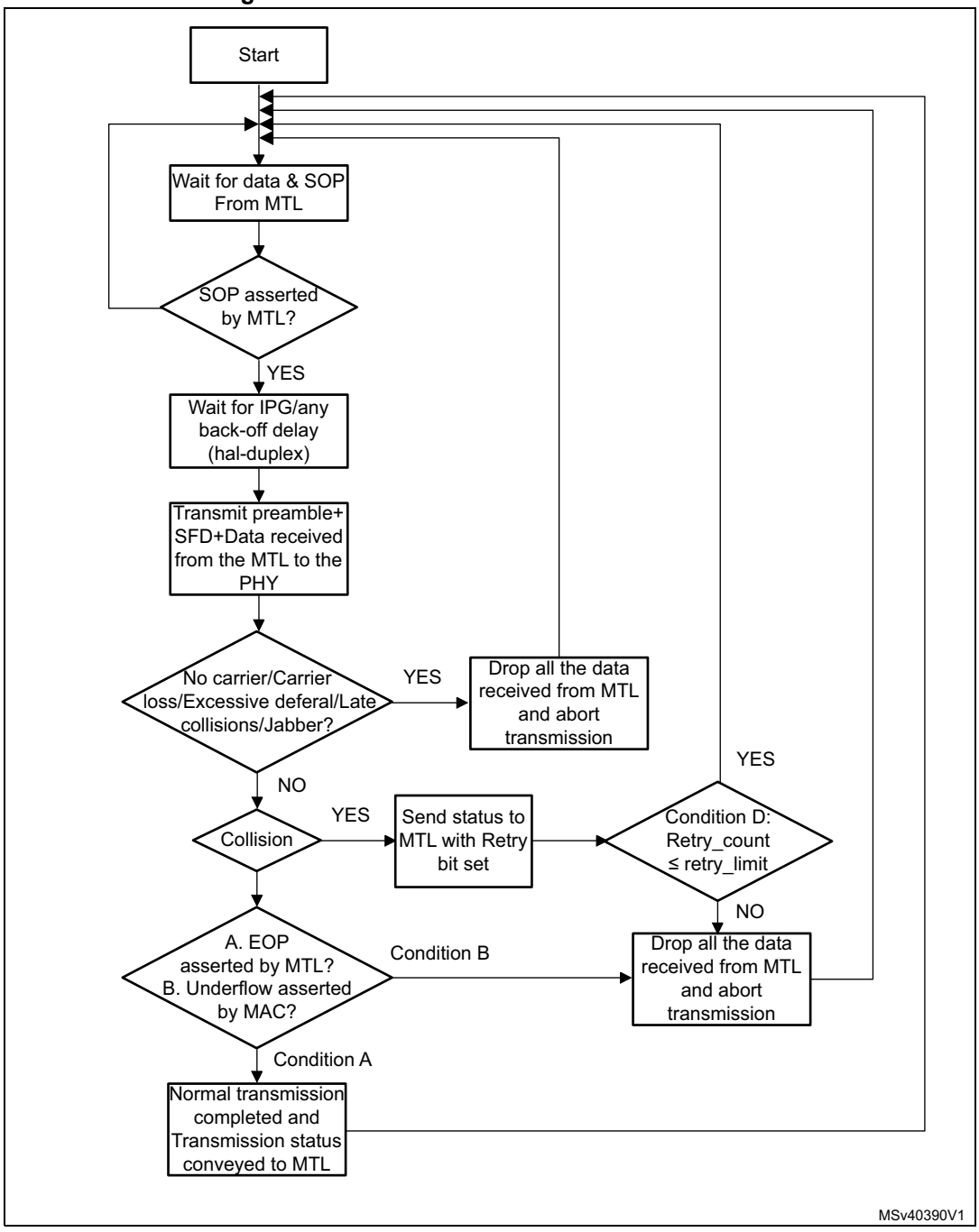
     Jabber

The MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.
5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

Figure 779: Overview of MAC transmission flow illustrates the MAC transmission process flow.

Figure 779. Overview of MAC transmission flow



MAC reception

A receive operation is initiated when the MAC detects an SFD on MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

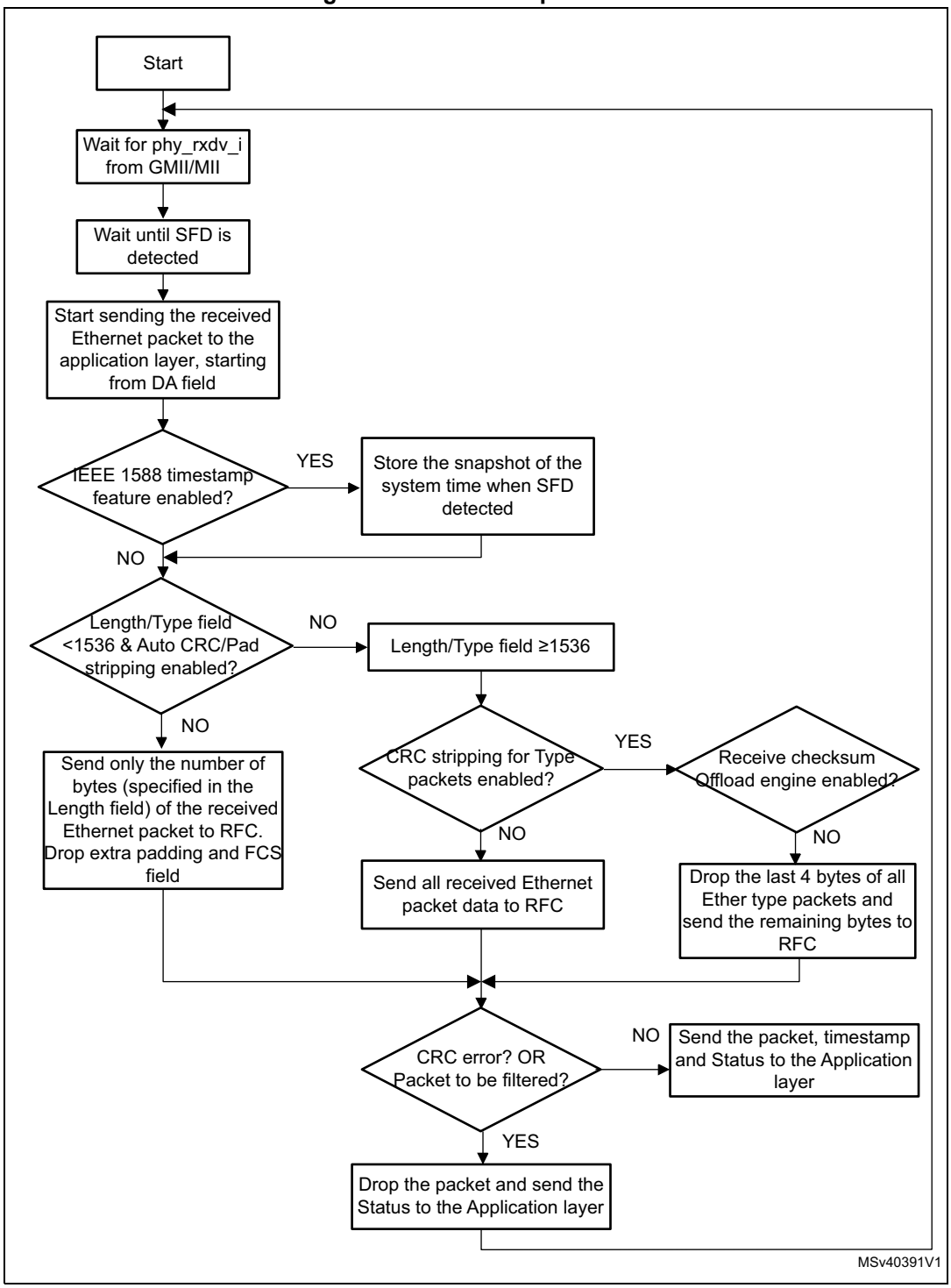
The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of MII becomes active, the Receive State Machine (RSM) starts looking for the SFD field (0xD nibble in MII mode).

   The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).
3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application. The MAC converts the received nibble data into bytes and forwards the valid packet data to the RFC module.
4. The Receive State Machine decodes the Length/Type field of the Ethernet packet being received.

   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the *Operating mode configuration register (ETH_MACCR)*), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.
5. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the *Operating mode configuration register (ETH_MACCR)*. However, if the CRC stripping has been enabled for Type packets and not enabled the Receive Checksum Offload Engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.
6. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of *Watchdog timeout register (ETH_MACWTR)*) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of *Operating mode configuration register (ETH_MACCR)*. However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.

Figure 780. MAC reception flow



57.5 Ethernet functional description: MAC

57.5.1 Double VLAN processing

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

Transmit path

Table 504: Double VLAN processing features in Tx path describes the features supported by the MAC on the Transmit side.

Table 504. Double VLAN processing features in Tx path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of *VLAN inclusion register (ETH_MACVIR)* and *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.

Table 504. Double VLAN processing features in Tx path (continued)

Feature	Description
VLAN Tag deletion	VLAN tag deletion can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the Control Word.

Receive path

Table 505: Double VLAN processing in Rx path describes the features supported by the MAC on the Receive side and the corresponding bits in the *VLAN tag register (ETH_MACVTR)*.

Table 505. Double VLAN processing in Rx path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

57.5.2 Source Address and VLAN insertion, replacement, or deletion

Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field.

Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets

  To enable this feature for all packets, program the SARC field of the *Operating mode configuration register (ETH_MACCR)*.
- Enabling SA insertion or replacement for selective packets

  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of Transmit Descriptor Word 3/TDES3, refer to *Section 57.10.3: Transmit descriptor*) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

VLAN insertion, replacement, or deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

  Insertion or replacement is performed based on the setting of VLTI bit in the *VLAN inclusion register (ETH_MACVIR)* as described in *Table 506: VLAN insertion or replacement based on VLTI bit*.

Table 506. VLAN insertion or replacement based on VLTI bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with VT field of Transmit context descriptor of the packet
VLTI bit is reset	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with the VLT field of *VLAN inclusion register (ETH_MACVIR)*

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of *VLAN inclusion register (ETH_MACVIR)*.
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see *Table 526: TDES2 normal descriptor (read format)*).

  In addition, the VLP (VLAN Priority control) bit must be reset in *VLAN inclusion register (ETH_MACVIR)* (for outer VLAN) and *Inner VLAN inclusion register (ETH_MACIVIR)* (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.

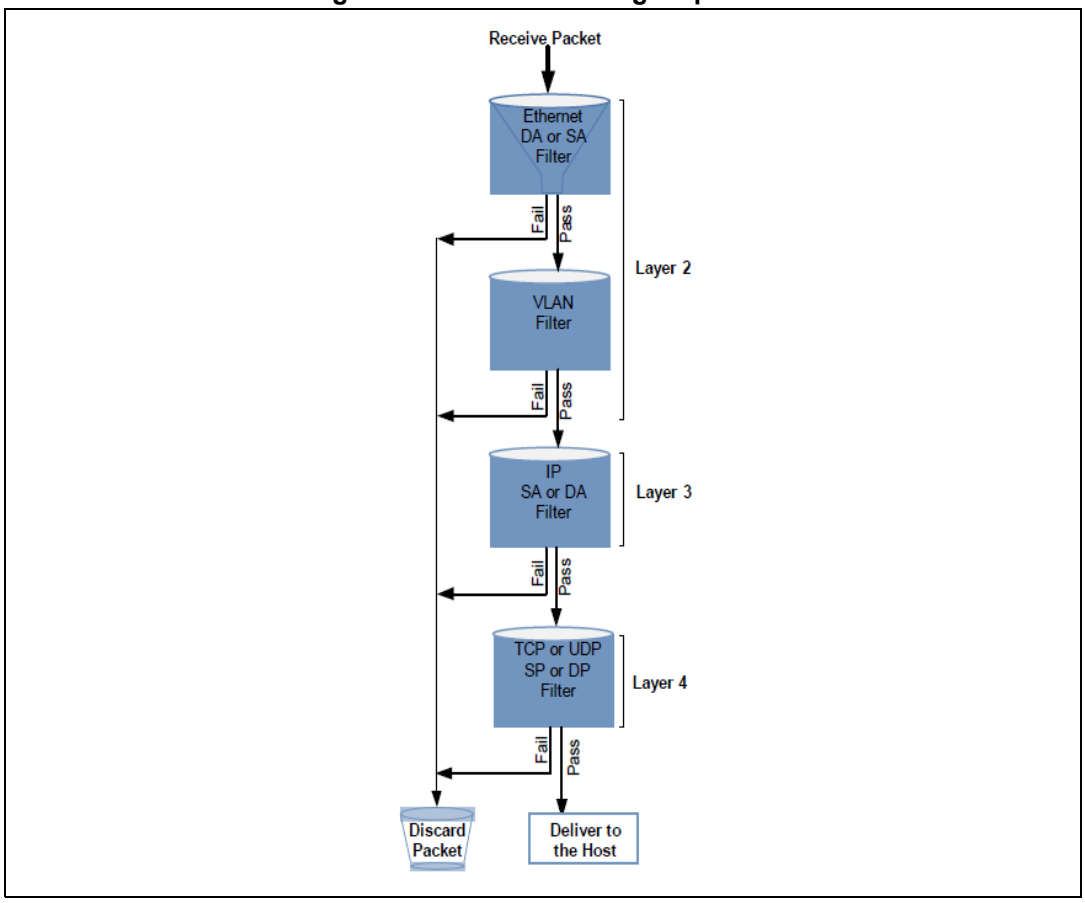
57.5.3 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- *MAC source or destination address filtering*: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- *VLAN filtering*: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- *Layer 3 and Layer 4 filtering*: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 781* shows the filtering sequence for Rx packets.

Figure 781. Packet filtering sequence



MAC source or destination address filtering

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast destination address filtering

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of *Packet filtering control register (ETH_MACPFR)* is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in the register. This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

Multicast destination address filtering

To program the MAC to pass all multicast packets, set the PM bit in *Packet filtering control register (ETH_MACPFR)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the *Packet filtering control register (ETH_MACPFR)*.

In Perfect filtering mode, the multicast address is compared with the programmed MAC Destination Address registers (1–31). Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

Hash or Perfect address filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Packet filtering control register (ETH_MACPFR)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

Broadcast address filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in *Packet filtering control register (ETH_MACPFR)*.

Unicast source address filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of corresponding register.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in *Packet filtering control register (ETH_MACPFR)*. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word (see *Table 508*). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

Inverse filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of *Packet filtering control register (ETH_MACPFR)*. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 507 and *Table 508* summarize the DA and SA filtering based on the type of packets received.

Table 507. Destination address filtering

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 508. Source address filtering

Packet type	PR	SAIF	SAF	SA Filter Operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN filtering

The MAC supports Perfect and Hash VLAN filtering. Refer to *Section 57.9.11: Programming guidelines to perform VLAN filtering on the receive* for detailed programming steps.

VLAN tag Perfect filtering

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of *VLAN tag register (ETH_MACVTR)*. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of ETH_MACVTR register. The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results. By default, the VLAN tag-based Perfect filter is available in all configurations.

VLAN tag Hash filtering

The MAC provides VLAN tag Hash filtering with a 16-bit Hash table. The MAC performs the VLAN Hash matching based on the VTHM of the *VLAN tag register (ETH_MACVTR)*. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag register (ETH_MACVTR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 509 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of *Packet filtering control register (ETH_MACPFR)* is set, all packets are received and the VLAN match status is indicated in the VF bit of *RDES2 normal descriptor (write-back format)*. When the RA bit is not set and the VTFE bit is set in *Packet filtering control register (ETH_MACPFR)*, the packet is dropped if the final VLAN match status is Fail. In *Table 509*, value *X* means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of *VLAN tag register (ETH_MACVTR)*, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in ETH_MACVTR register.

Table 509. VLAN match status(1)

VID	VLAN perfect filter match result	VTHM Bit	VLAN Hash filter match result	VTIM bit	Final VLAN match status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, X represents any value.

Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the *Operating mode configuration register (ETH_MACCR)* to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets

  The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of *Operating mode configuration register (ETH_MACCR)* is set and one of the following conditions is true:

  – All enabled Layer 3 and Layer 4 fields match.

  – At least one of the enabled field matches and other fields are bypassed or disabled

    When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

    The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets

  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets.

- Non-TCP or UDP IP Packets

  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 and Layer 4 filters register set

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as ETH_MACL3L4C0R, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- *Layer4 address filter 0 register (ETH_MACL4A0R)*
- *Layer 3 Address 0 filter 0 register (ETH_MACL3A00R)*
- *Layer3 address 1 filter 0 register (ETH_MACL3A10R)*
- *Layer3 Address 2 filter 0 register (ETH_MACL3A20)*
- *Layer3 Address 3 filter 0 register (ETH_MACL3A30)*

The second, and independent set of registers are: ETH_MACL3L4C1R, ETH_MACL4A01R, ETH_MACL4A11R, ETH_MACL4A21R and ETH_MACL4A31R

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

Layer 4 filtering

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

57.5.4 IEEE 1588 timestamps

The IEEE 1588 standard defines a Precision Time Protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588

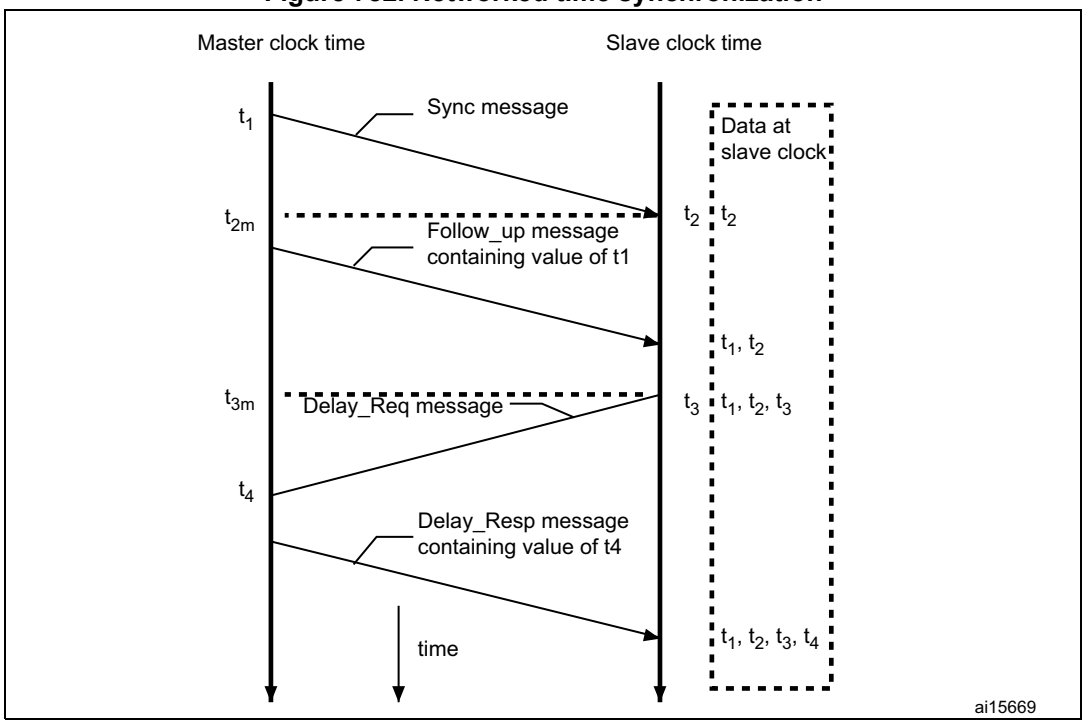
2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement sub-second time in digital or binary format

Delay request-response mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 782* shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

Figure 782. Networked time synchronization



As shown in *Figure 782*, the PTP uses the following process:

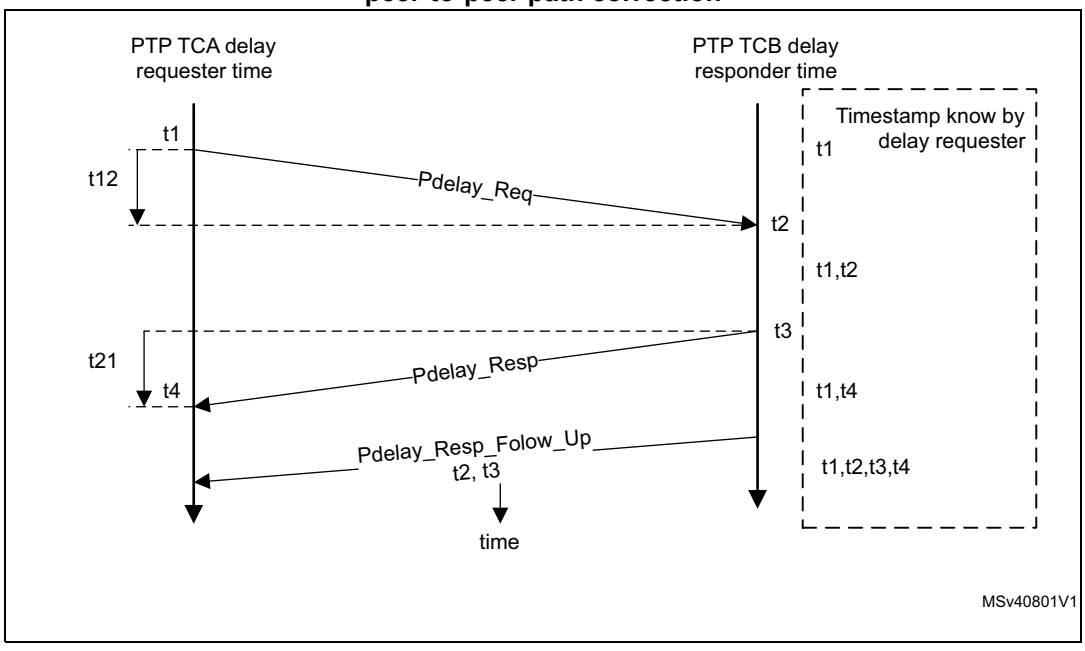
1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t_1. This time must be captured for Ethernet ports at MII.
2. The slave receives the Sync message and also captures the exact time, t_2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, t_3, at which this packet leaves the MII interface.
5. The master receives the message, capturing the exact time t_4, at which the message enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference to the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

Peer-to-peer PTP transparent clock (P2P TC) message support

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow-up, and Delay Response messages. *Figure 783* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 783. Propagation delay calculation in clocks supporting peer-to-peer path correction



As shown in *Figure 783*, the propagation delay is calculated as follows:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t_1) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t_2) for this message.
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t_3) for this message.

   To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp message
   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp_Follow_Up message
   – Timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively

   Port 1 generates a timestamp (t_4) on receiving the Pdelay_Resp message.

   Port 1 uses all four timestamps to compute the mean link delay.

Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- Ordinary clock

  The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

  The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

  – Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in *Timestamp control Register (ETH_MACTSCR)*.
  – Maintenance of the data sets such as timestamp values.

  The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 510. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

  For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in *Timestamp control Register (ETH_MACTSCR)*.

- Boundary clock

  The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and signaling end in the protocol engine of the boundary clock. Such messages are not

forwarded. The PTP message type status given by the MAC helps you to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

– The clock data sets are common to all ports of the boundary clock.

– The local clock is common to all ports of the boundary clock.

- End-to-end transparent clock

  The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

  The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in *Timestamp control Register (ETH_MACTSCR)*. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the *Timestamp control Register (ETH_MACTSCR)*. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the *Timestamp control Register (ETH_MACTSCR)*.

Table 511. End-to-end transparent clock: PTP messages for snapshot

PTP Messages
SYNC
Delay_Req

- Peer-to-peer transparent clock

  The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

  In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer. The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in *Timestamp control Register (ETH_MACTSCR)*.

Table 512. Peer-to-peer transparent clock: PTP messages for snapshot

PTP Messages
SYNC
Pdelay_Req
Pdelay_Resp

You can take the snapshot by setting the SNAPTYPESEL bit to 11 in *Timestamp control Register (ETH_MACTSCR)*.

Reference timing source

To get a snapshot of the time, the MAC provides an internal reference time in 80-bit format as defined in the IEEE 1588-2008 specifications.

The MAC clock input is uses to generate the reference time (also called the system time) and capture timestamps. The timestamp has the following fields:

- UInteger48 secondsField
- The secondsField is the integer portion of the timestamp in units of seconds. It is 48-bits wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000_0000_0002.
- UInteger32 nanosecondsField
- The nanosecondsField is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000_0001.

  The nanosecondsField supports the following two modes:

  – **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.

  – **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit.

  You can set these modes through TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)*.

The reference timing is accessible by an output pulse-per-second signal (see *Pulse-per-second output*) or by storing up to 4 snapshots in dedicated FIFO register (*Auxiliary snapshots with external events*).

Pulse-per-second output

The MAC supports either a fixed or a flexible pulse-per-second signal (ptp_pps_o)

- Fixed pulse-per-second output

  In this mode, only frequency of the PPS output can be changed by setting the PPSCTRL0 field in the *PPS control register (ETH_MACPPSCR)*.

- Fixed pulse-per-second output

  In this mode, you have the flexibility to program the start or stop time, width, and interval of the pulse generated on the ptp_pps_o output.

  – The start and stop time are programmed through *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)* registers.

  – The PPS width and interval are programmed in terms of granularity of system time (number of the units of sub-second increment value) through respective registers *PPS width register (ETH_MACPPSWR)* and *PPS interval register (ETH_MACPPSIR)*.

Refer to *Section 57.9.9: Programming guidelines for flexible pulse-per-second (PPS) output* for further details on how configuring flexible pulse output.

Note: To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

Auxiliary snapshots with external events

The auxiliary snapshot feature allows you to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the ptp_aux_ts_trig_i sideband signal.

You can configure up to four auxiliary snapshot inputs and store up to 4 snapshots. A FIFO is accessible through registers: *Auxiliary timestamp seconds register (ETH_MACATSSR)* and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*.

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the *Timestamp status register (ETH_MACTSSR)* to know the timestamp of which input is available for reading at the top of this FIFO.

When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the *Timestamp status register (ETH_MACTSSR)*. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When a application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in *Auxiliary control register (ETH_MACACR)*. When multiple snapshots are present in the FIFO, the count is indicated in Bits[27:25] of ETH_MACTSSR register.

System time register module

The 64-bit PTP time is updated using the PTP input reference clock, HCLK. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being

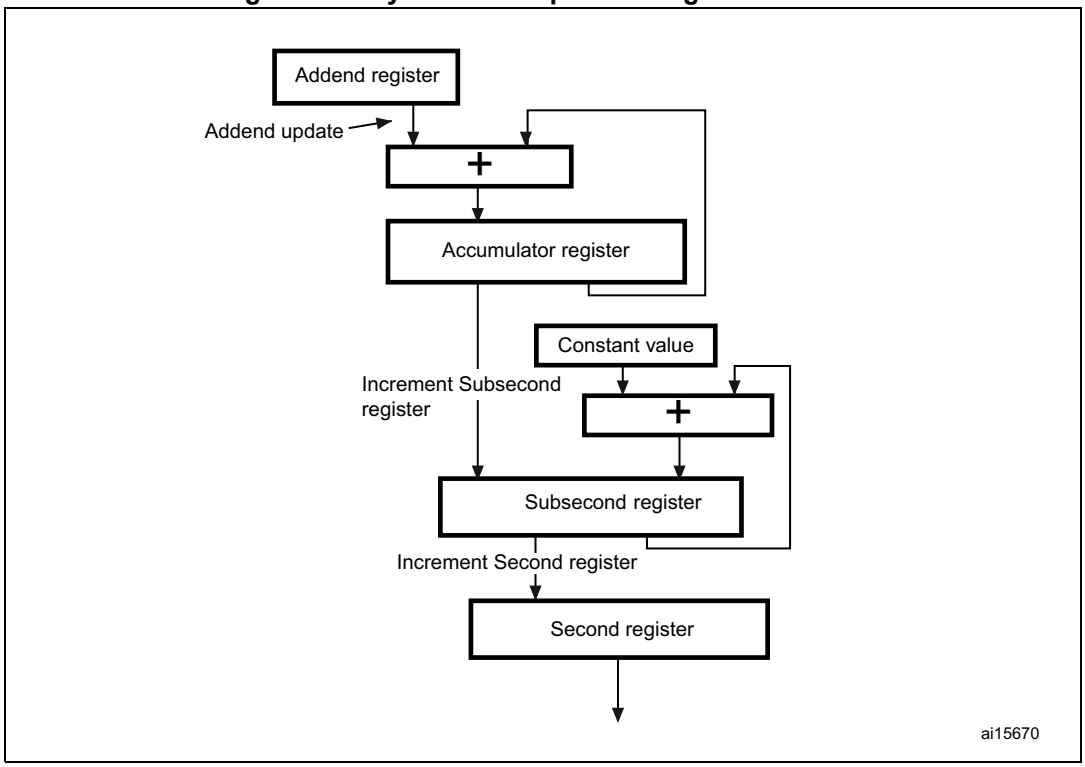
transmitted or received at the MII. The System Time counter can be initialized or corrected using either the Coarse or the Fine correction method.

In the Coarse correction method, the initial value or the offset value is written to the Time stamp update register For initialization, the System Time counter is written with the value in the Time stamp update registers, whereas for system time correction, the offset value (Time stamp update register) is added to or subtracted from the system time.

In the Fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588 specifications) is corrected over a period of time, unlike in the Coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in *Figure 784*. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in *Figure 784*.

Figure 784. System time update using Fine method



The System Time Update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (HCLK) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B7672. When there is not clock drift, the default addend value of 0xC1F07C1F (2^{32} / 1.32) must be programmed.

In *Figure 784*, the constant value used to accumulate the sub-second register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps). When External Time Update is enabled, the optional System Time module is not available. Two different methods are used to update the System Time register depending on the configuration.

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

1. At time MasterSyncTime_n the master sends the slave clock a SYNC message. The slave receives this message when its local clock is SlaveClockTime_n and computes MasterClockTime_n as follows:

$$\text{MasterClockTime}_n = \text{MasterSyncTime}_n + \text{MasterToSlaveDelay}_n$$

2. The master clock counts for current Sync cycle, $\text{MasterClockCount}_n$ is

$$\text{MasterClockCount}_n = \text{MasterClockTime}_n - \text{MasterClockTime}_{n-1}$$

(assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

3. The slave clock count for current Sync cycle, SlaveClockCount_n is

$$\text{SlaveClockCount}_n = \text{SlaveClockTime}_n - \text{SlaveClockTime}_{n-1}$$

4. The difference between master and slave clock counts for current Sync cycle, ClockDiffCount_n is

$$\text{ClockDiffCount}_n = \text{MasterClockTime}_n - \text{SlaveClockTime}_n$$

5. The frequency-scaling factor for slave clock, FreqScaleFactor_n is

$$\text{FreqScaleFactor}_n = (\text{MasterClockCount}_n + \text{ClockDiffCount}_n) / \text{SlaveClockCount}_n$$

6. The frequency compensation value for Addend register, $\text{FreqCompensationValue}_n$ is

$$\text{FreqCompensationValue}_n = \text{FreqScaleFactor}_n \times \text{FreqCompensationValue}_{n-1}$$

In theory, this algorithm achieves lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This

algorithm is self-correcting. If the slave clock is initially set to an incorrect value from the master, the algorithm corrects it at the cost of more Sync cycles.

Refer to *Section 57.9.7: Programming guidelines for IEEE 1588 timestamping* for detailed programming steps.

Transmit path functions

The MAC captures a timestamp when the Start Packet Delimiter (SFD) of a packet is sent on the MII interface. The packets, for which you want to capture timestamps, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. You need to specify the packets for which you want to capture timestamps. You can specify the packets by using the control bits in Transmit Descriptor (see *Section 57.10.3: Transmit descriptor*). The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus connecting the timestamp automatically to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. You can control the snapshot of the time to be sent to the application by using the following options of the *Timestamp control Register (ETH_MACTSCR)*:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type

  This feature controls the type of messages for which snapshots are taken.

The DMA returns the timestamp to the software application inside the corresponding Receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Timestamp correction

According to the IEEE 1588 specification, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network. Because the reference timing source (the PTP clock ***HCLK***) is different from the MAC Tx or Rx clock, the captured timestamp must be corrected for latency issues because of synchronization. In addition, latency issues between the internal snapshot point and the recommended capture point (the boundary between the node and the network), must also be corrected.

Ingress correction

In the Receive side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Ingress Correction Value) programmed in the Ingress Correction register. The value that needs to be programmed in the ingress correction register is calculated as follows:

1. The timestamp correction because of synchronization is compensated by adding INGRESS_SYNC_CORR to the synchronized timestamp value as follows:

   `INGRESS_SYNC_CORR = -(2*PTP_CLK_PER)`

2. The latency correction between the message timestamp point and the internal timestamp snapshot point is done by subtracting the latency value (INGRESS_LATENCY) with the captured timestamp as follows:

   `Ingress Correction = INGRESS_SYNC_CORR - INGRESS_LATENCY`

3. Ingress correction is performed by programming the TSIC field in the MAC Timestamp Ingress correction register. The ingress correction is always negative and is expressed in nanoseconds. The value is represented in complement form as follows:
   - When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set, the accuracy is of 1 ns: it is represented by setting bit 31 to 1 and bits 30:0 containing 10^9 - <ingress_correction_value> represented in binary. For example, if the required correction value is -5 ns, then the programmed value must be 0xBB9A_C9FB.
   - When TSCTRLSSR bit in ETH_MACTSCR register is reset, the accuracy is of ~0.466 ns: it is represented by setting bit 31 to 1 and bits 30:0 containing 2^31 - <ingress_correction_value> represented in binary.

Egress correction

In the Transmit side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Egress correction value) programmed in the Egress Correction register. The value that needs to be programmed in the egress correction register is calculated as follows:

1. The timestamp correction because of synchronization is compensated by adding EGRESS_SYNC_CORR to the synchronized timestamp value as follows:

   When Enable one step timestamp feature is selected,

   `EGRESS_SYNC_CORR = (1*PTP_CLK_PER + 4*TX_CLK_PER)`

   Otherwise,

   `EGRESS_SYNC_CORR = -(2*PTP_CLK_PER)`

2. The egress latency correction between the recommended capture point and the internal timestamp snapshot point is obtained by adding the latency value (EGRESS_LATENCY) to the captured timestamp as follows:

   `Egress Correction = EGRESS_SYNC_CORR + EGRESS_LATENCY`

3. Egress correction is performed by programming the TSEC field in the MAC Timestamp Egress correction register. The egress correction can be positive or negative. It is

expressed in nanoseconds. Negative values are represented in complement form as follows:

– When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set, the accuracy is of 1 ns.

  If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The value must not exceed 0x 3B9A_C9FF.

  If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 10^9 - <egress_correction_value> represented in binary.

  For example, if the required correction value is -5 ns, then the programmed value should be 0xBB9A_C9FB.

– When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset, the accuracy is of ~0.466 ns.

  If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The maximum value is 0x7FFF_FFFF.

  If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 2^31 - <egress_correction_value> represented in binary.

One-step timestamp

The MAC supports the one-step timestamp feature: It identifies the offset in the packet and inserts the timestamp received from the application at that offset.

You can enable the one-step timestamp feature for a packet by setting bit 20 (OSTC) in ATI Control Word. The inserted timestamp consists of the 64-bit TSSL and TSSH received from the application.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

PTP offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network. These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. *Table 513* indicates the PTP message generation criteria.

Table 513. PTP message generation criteria

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
00	0	1	Ordinary or Boundary Slave	SYNC message reception	Delay_Req
00	1	1	Ordinary or Boundary Master	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
01	0	1	Transparent Slave	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req
01	1	1	Transparent Master	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
11	X	X	Peer-to-Peer Transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
All other programming combinations are invalid for PTP Offload feature.					

The PTP offload feature is configured through *PTP Offload control register (ETH_MACPOCR)* register. 80 bits-PTP node identity is given in the three following registers: *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.

57.5.5 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets**.** ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:

1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply

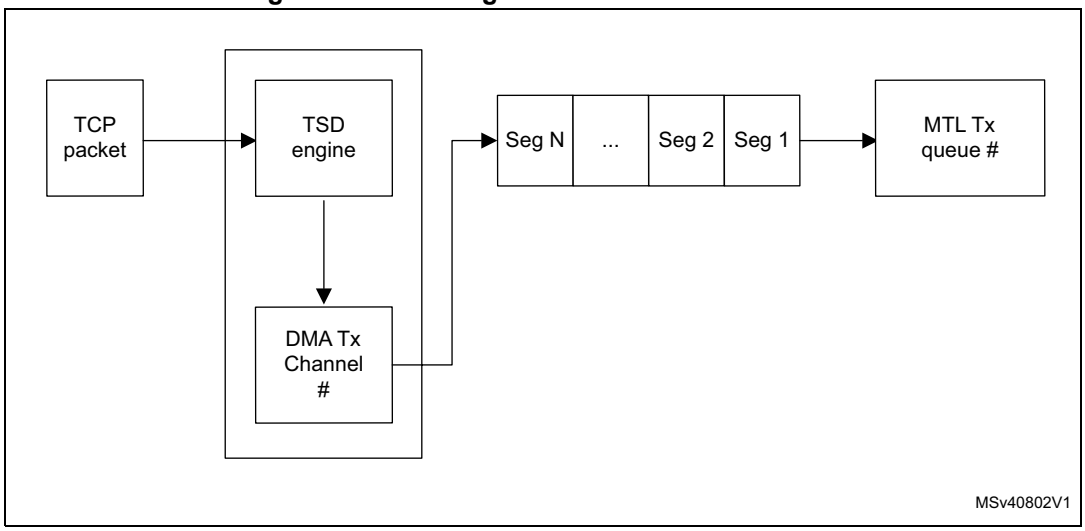
57.5.6 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in *Figure 785*.

This feature is enabled by programming the TSE bit of corresponding ETH_DMACCR register (see *Channel control register (ETH_DMACCR)*). It is only supported when the MAC operates in full-duplex mode.

For detailed programming steps, refer to *Section 57.9.10: Programming guidelines for TSO*.

Figure 785. TCP segmentation offload overview



Enabling the TSO feature

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see *Section 57.10.3: Transmit descriptor*).

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMACCR (see *Channel control register (ETH_DMACCR)*) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMACTXCR register (see *Channel transmit control register (ETH_DMACTXCR)*) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQOMR register (see *Tx queue operat-*

ing mode Register (ETH_MTLTXQOMR)). A MSS plus header equal to half the programmed Tx queue size is recommended.

If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. *Table 514* describes how the TCP and IP headers are updated.

Table 514. TSO: TCP and IP header fields

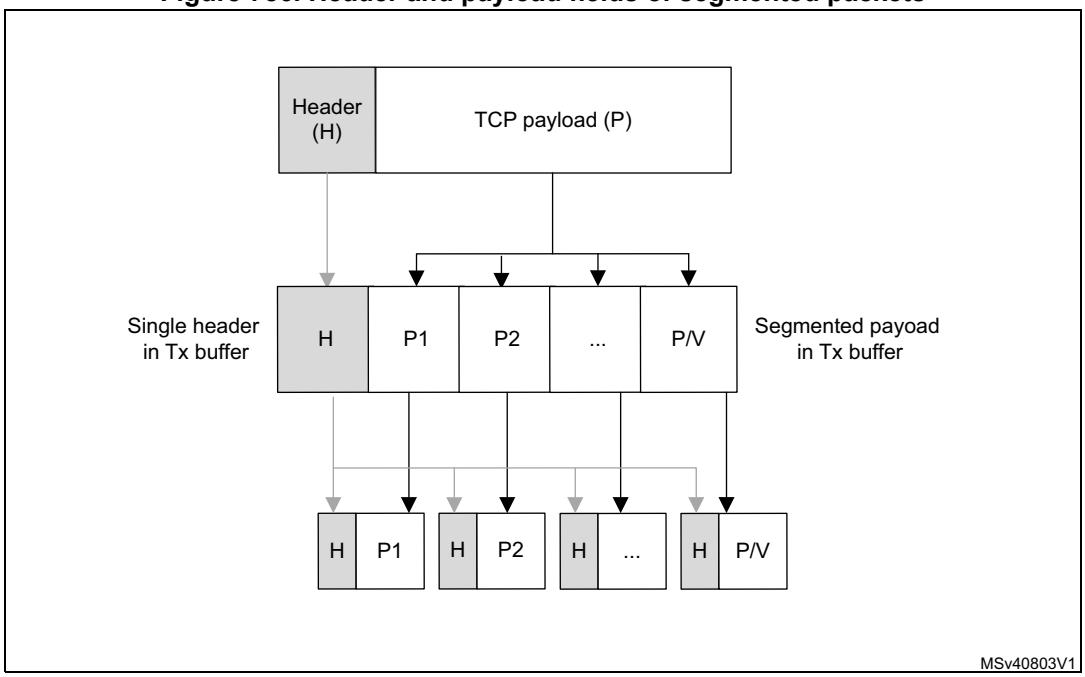
Packet sequence	TCP header	IP header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field is not modified. It is sent as per the header provided by the software. – IPv4 Header Checksum is recalculated. IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field = Previous Identification Field + 1 – IPv4 Header Checksum is recalculated IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = Remaining Payload + TCP Header Length + IP Header Length – Identification Field = Previous Identification Field + 1 – IPv4 header Checksum is recalculated IPv6 Header – Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

Header and payload fields of segmented packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in *Table 514: TSO: TCP and IP header fields*. *Figure 786: Header and payload fields of segmented packets* shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

Figure 786. Header and payload fields of segmented packets



Context descriptor sequence

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to *Section 57.10.3: Transmit descriptor*).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see *Channel interrupt enable register (ETH_DMACIER)*).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see *Channel status register (ETH_DMACSR)*).

57.5.7 Loopback

The MAC supports the Loopback of transmitted packets to its receiver. By default, the MAC Loopback function is disabled, but it can be enabled by programming the LM bit of the *Operating mode configuration register (ETH_MACCR)* register.

The Loopback function is available for all PHY interfaces. The data is always looped back on the MII interface irrespective of which PHY interface is selected.

57.5.8 Flow control

This section describes the flow control for Transmit and Receive paths.

Transmit flow control

The Transmit Flow Control is enabled when TFE bit is set in *Tx Queue flow control register (ETH_MACQTXFCR)*.

Flow control trigger

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end. The application can request the MAC to send a Pause packet or initiate backpressure by using setting the FCB bit in the corresponding *Tx Queue flow control register (ETH_MACQTXFCR)*.

Flow control in full-duplex mode: pause packets control

In full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. *Table 515* describes the fields of a Pause packet.

Table 515. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets; 0101 for PFC packets
PT	Contains Pause time specified in the PT field of the *Tx Queue flow control register (ETH_MACQTXFCR)*

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Flow control in half-duplex mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 516 describes the flow control in the Tx path based on the setting of the following bits:

- EHFC bit of *Rx queue operating mode register (ETH_MTLRXQOMR)*
- TFE bit of *Tx Queue flow control register (ETH_MACQTXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Flow control is similar for all queues.

Table 516. Tx MAC flow control

EFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set.
1	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set. In addition, the MAC Tx performs backpressure when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue operating mode register (ETH_MTLRXQOMR)*.
0	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set.
1	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set. In addition, the MAC Tx sends a Pause packet when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue operating mode register (ETH_MTLRXQOMR)*.

Receive flow control

In the Receive path, the flow control is functional only in full-duplex mode. If any Pause packet is received in half-duplex mode, the packet is considered as a normal control packet.

You can enable the Pause flow control by setting the RFE bit in the *Rx flow control register (ETH_MACRXFCR)*. *Table 517* describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of *Rx flow control register (ETH_MACRXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Table 517. Rx MAC Flow Control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

The following sequence describes the Rx flow control:

1. The MAC checks the destination address of the received Pause packet for either multicast or unicast destination address.
2. The MAC decodes the Type (0x8808) and Opcode (0x0001: Pause packet) fields of the received packet. The Pause time (for Pause packet) is captured to determine the time for which transmitter needs to be blocked.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

57.5.9 Checksum offload engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the Receive path.

Transmit checksum offload engine

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement the Tx FIFO automatically operates in the store-and-forward mode even if the MAC is configured for Threshold (cut-through) mode.

IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status it in the Transmit status (bit 0 in *Table 531: TDES3 normal descriptor (write-back format)*).

TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in *Table 531: TDES3 normal descriptor (write-back format)*). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 518 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Table 518. Transmit checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Authentication – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – No – No – Yes – Yes – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive checksum offload engine

You can enable the Receive Checksum Offload Engine (Rx COE) by selecting the *Enable Receive TCP/IP Checksum Check* option and setting the IPC bit of *Operating mode configuration register (ETH_MACCR)*. When this option is selected, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the *Enable Double VLAN Processing* option is selected and the EDVLP bit of the *VLAN tag register (ETH_MACVTR)* is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 519: Receive checksum offload engine functions for different packet types describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 519. Receive checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No

Table 519. Receive checksum offload engine functions for different packet types (continued)

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – Yes – No – Yes – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

57.5.10 MAC management counters

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in *Section 57.11.4: Ethernet MAC and MMC registers*.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

In addition to control registers, two sets of registers are implemented:

- 6 registers used for collision, error and good packets counters
- 4 registers to record LPI mode transition

57.5.11 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the *Interrupt enable register (ETH_MACIER)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

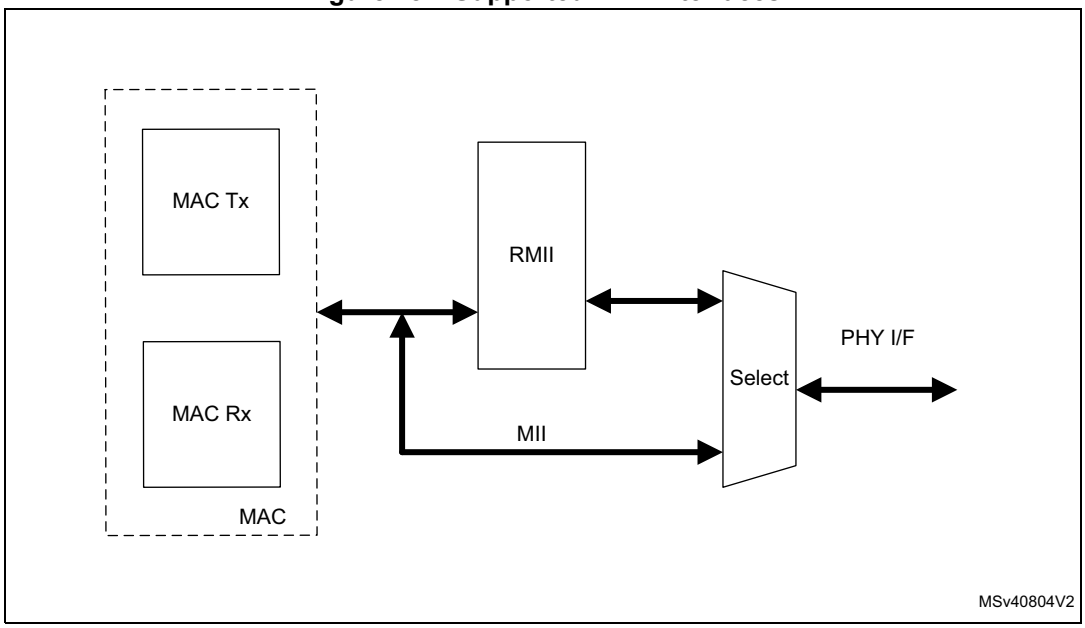
57.5.12 MAC and MMC register descriptions

Refer to *Section 57.11.4: Ethernet MAC and MMC registers*.

57.6 Ethernet functional description: PHY interfaces

The Ethernet peripheral support several PHY interfaces. The root interface is the MII one. All other interfaces are derived from it as shown in *Figure 787*.

Figure 787. Supported PHY interfaces



This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

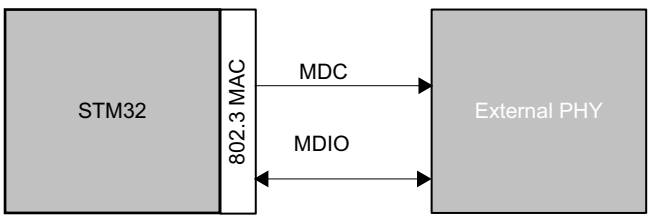
- *Station management agent (SMA)*
- *Media Independent Interface (MII)*
- *Reduced media independent interface (RMII)*

57.6.1 Station management agent (SMA)

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in *Figure 788*.

Figure 788. SMA Interface block



SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock. The division factor depends on the clock range setting in the *MDIO address register (ETH_MACMDIOAR)* register. The MDC clock is selected as follows:

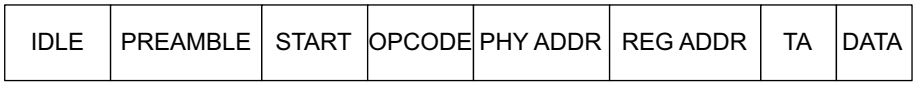
Table 520. MCD clock selection

Selection	CSR clock	MDC clock
0000	60–100 MHz	CSR clock/42
0001	100–150 MHz	CSR clock/62
0010	20–35 MHz	CSR clock/16
0011	35–60 MHz	CSR clock/26
0100	150–250 MHz	CSR clock/102
0101	250–300 MHz	CSR clock/124
0110, 0111	Reserved	-

The data exchange between the MAC and the PHY is performed through mdi_i, mdo_o and mdo_oe signals. This signal group is passed through a three-state buffer and brought out as MDIO line connected to the PHY.

The following figure shows the structure of a packet on the MDIO packet while *Table 521* provides a detailed description of the packet fields.

Figure 789. MDIO packet structure



MSv40806V1

Table 521. MDIO packet field description

Field	Description
IDLE	The MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2’01
OPCODE	2’b10 for Read and 2’b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround is 2’bZ0 for Read and 2’b10 for Write
DATA	Any 16-bit value. In a Write operation, the MAC drives MDIO. In a Read operation, the PHY drives it.

MII management write operations

When bit[3:2] are set to 01 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (*MDIO data register (ETH_MACMDIODR)*) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the MII Management Interface using the Management Packet Format specified in the MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations performed to the *MDIO address register (ETH_MACMDIOAR)* or the *MDIO data register (ETH_MACMDIODR)* during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

Figure 790. Write data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

MSv40807V1

MII management read operation

When bit[3:2] are set to 11 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the MII Management Interface using the Management Packet Format specified in the MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the *MDIO address register (ETH_MACMDIOAR)* or *MDIO data register (ETH_MACMDIODR)* register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the *MDIO data register (ETH_MACMDIODR)* with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

Figure 791. Read data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	10	AAAAA	RRRRR	Z0	DDD...DDD	Z

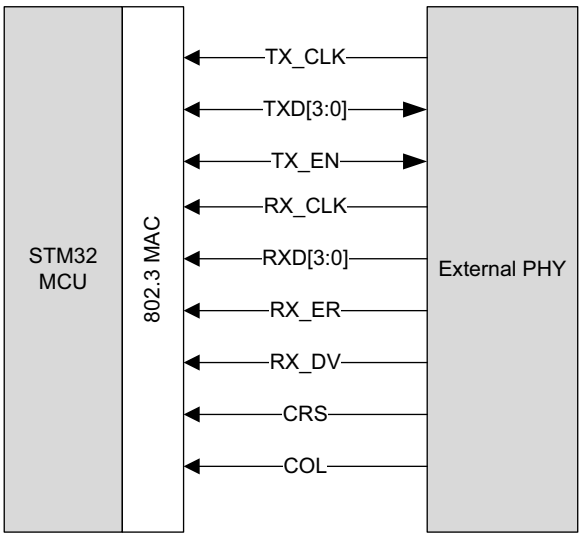
MSv40808V1

57.6.2 Media Independent Interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sub-layer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 792: Media independent interface (MII) signals*.

Figure 792. Media independent interface (MII) signals



MSv40809V1

- TX_CLK: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TXD[3:0]: transmit data.

  TXD is a bundle of 4 data signals driven synchronously by the MAC sub-layer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least

significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted the transmit data must have no effect upon the PHY.

- CRS: carrier sense.

  This signal is asserted by the PHY when either transmit or receive medium is non idle. It shall be deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In full duplex mode the state of this signal is don't care for the MAC sub-layer.

- COL: collision detection signal

  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In full-duplex mode the state of this signal is don't care for the MAC sub-layer.

- RXD[3:0]: reception data

  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY (see *Table 522*).

- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.

- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sub-layer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion as described in *Table 522*.

Table 522. RX interface signal encoding

RX_DV	RX_ERR	RXD[3:0]	Description
0	0	0000 through 1111	Normal inter-frame
0	1	0000	Normal inter-frame
0	1	0001 through 1101	Reserved
0	1	1110	False carrier indication
0	1	1111	Reserved
1	0	0000 through 1111	Normal data reception
1	1	0000 through 1111	Data reception with errors

57.6.3 Reduced media independent interface (RMII)

The Reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU . According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

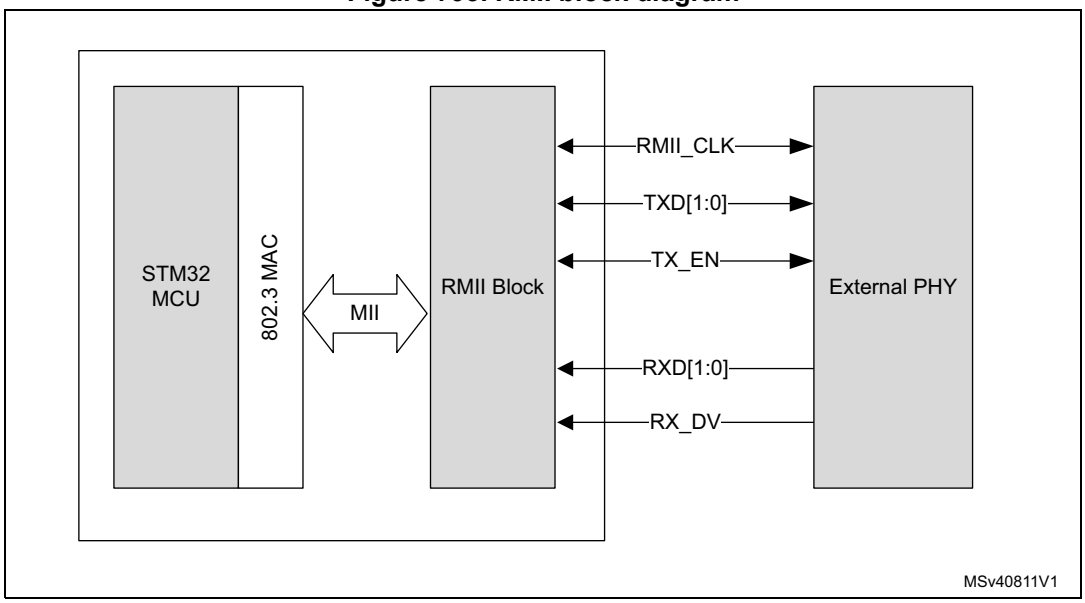
Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps operation.
- Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock references externally.

RMII block diagram

Figure 793: RMII block diagram shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

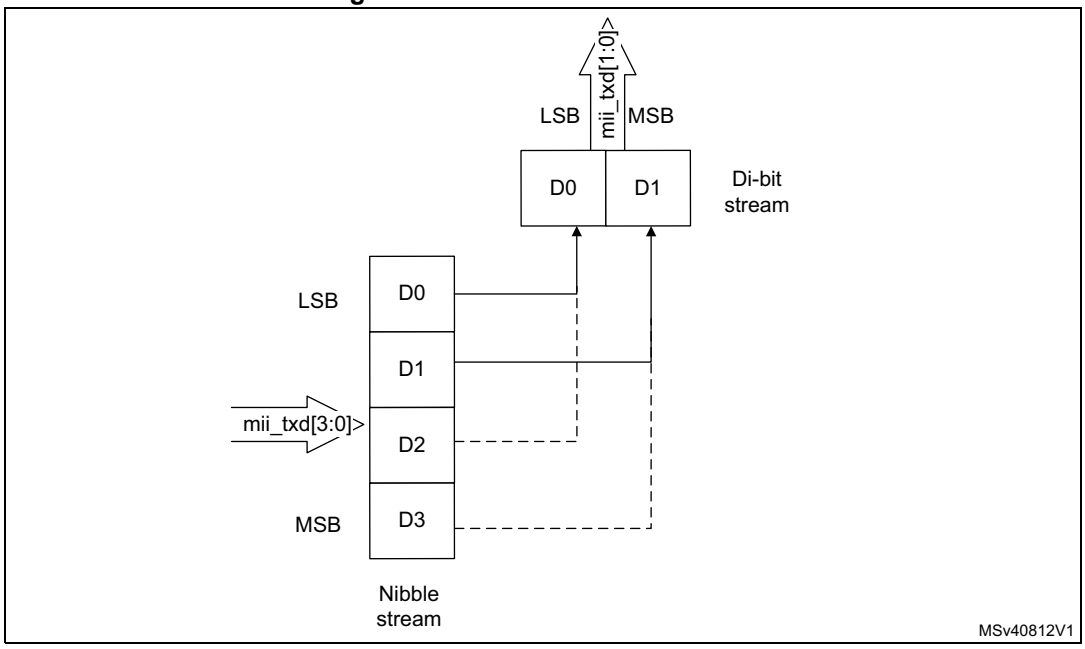
Figure 793. RMII block diagram



Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in *Figure 794: Transmission bit order*. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

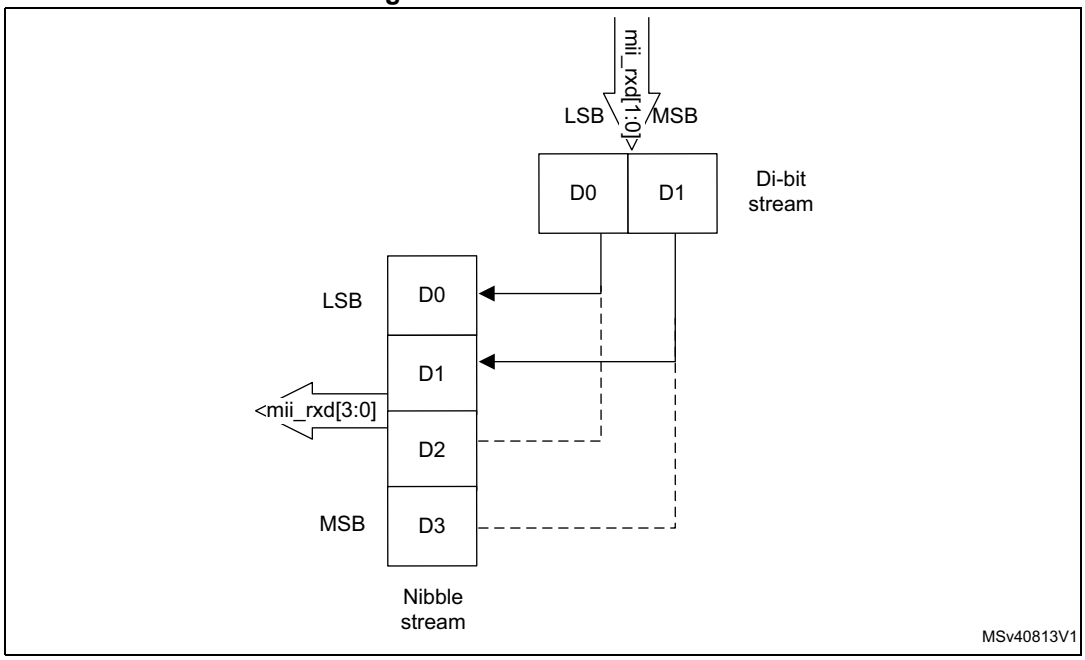
Figure 794. Transmission bit order



Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in *Figure 795: Receive bit order*. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 795. Receive bit order



57.7 Ethernet low-power modes

57.7.1 Energy Efficient Ethernet

Energy Efficient Ethernet (EEE) is an operating mode that enables the MAC sub-layer along with a family of Physical layers to operate in Low-power Idle (LPI) mode. The EEE mode supports operations at 100 Mbps.

The LPI mode allows power saving to be achieved by switching off some of the communication device functions when there is no data to be transmitted and received. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY interface.

The EEE specifies the negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Note: The EEE feature is not supported when the MAC is configured to use the RMII single PHY interface. Even if the MAC supports multiple PHY interfaces, the EEE mode should be activate only when the MAC operates with the MII interface.

The LPI mode is supported only in full-duplex mode.

Transmit path functions

In the Transmit path, the software must set the LPIEN bit of the *LPI control status register (ETH_MACLCSR)* to request the MAC to stop transmission and initiate the LPI protocol.

To exit the PHY from the LPI state, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:

   The MAC cannot start the transmission until the wakeup time specified for the PHY expires. The auto-negotiated wakeup interval is programmed in the TWT field of the *LPI control status register (ETH_MACLCSR)*.
3. Updates the LPI exit status (TLPIEX bit of the *LPI control status register (ETH_MACLCSR)*) and generates an interrupt.

Refer to *Section : Entering and exiting Tx LPI mode* for programming guideline of LPI mode.

Automatically entering/exiting LPI mode in Tx path

The MAC transmitter can be programmed to enter/exit LPI mode automatically based on whether it is IDLE for a specific period of time or has a packet to transfer. These modes are enabled and controlled by *LPI control status register (ETH_MACLCSR)*.

When LPITXA (bit19) and LPITXEN (bit16) of *LPI control status register (ETH_MACLCSR)* are set, the MAC transmitter enters LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter will exit the LPI IDLE state and clear the LPITXEN bit as soon as any of functions in the Tx path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition to the above, when Bit[20] (LPIATE) is also set, the MAC transmitter will enter LPI IDLE state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in *LPI entry timer register (ETH_MACLETR)*. In this mode also, the MAC transmitter will exit the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPITXEN bit is not cleared but remains active so that re-entry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both LPIATE and LPITXA bits are cleared, you can directly control the entry and exit of LPI IDLE state by programming the LPITXEN bit.

Receive path functions

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY goes in LPI mode and MAC updates the RLPIEN bit of the *LPI control status register (ETH_MACLCSR)* and immediately generates an interrupt.

When the PHY receives signals from the link partner to exit the LPI state, the PHY goes out of the LPI mode and MAC updates the RLPIEX bit of the *LPI control status register (ETH_MACLCSR)*. An interrupt is generated immediately.

LPI Interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The LPI interrupt can be cleared by reading the *LPI control status register (ETH_MACLCSR)*.

A sideband signal, lpi_intr_o, is generated together with the interrupt. This signal is used by the wakeup mechanism. It is ORed with pmt_intr_o signal (see *Section : PMT interrupts*) and tied to the EXTI peripheral (line 86).

57.7.2 Power management

The power management (PMT) block supports the reception of network (remote) wakeup packets and magic packets. The PMT block generates interrupts for remote wakeup packets and magic packets that the MAC receives.

When the power-down mode is enabled in the PMT block, the MAC drops all received packets and does not forward any packet to the RxFIFO or the application. The MAC comes out of the power-down mode only when a magic packet or a remote wakeup packet is received and the corresponding detection is enabled.

The PMT block is available in the receive path of MAC. Both types of power management packet (remote wakeup packet and magic packet) can be selected. RWKPKTEN and MGKPKTEN bits of the *PMT control status register (ETH_MACPCSR)* can be set to generate power management events.

The following are the PMT block registers:

- *PMT control status register (ETH_MACPCSR)*
- *Remote wakeup packet filter register (ETH_MACRWKPFR)*

Remote wakeup packet detection

When the MAC is in sleep mode and the remote wakeup bit is enabled in the *PMT control status register (ETH_MACPCSR)*, the normal operation is resumed after a remote wakeup packet is received.

Remote wakeup frame filter register

The PMT block supports 16 programmable filters that allow support of different receive packet patterns. If the incoming packet passes the address filtering of Filter Command, and if Filter CRC-16 matches the CRC of the incoming pattern, the MAC identifies the packet as a wakeup packet.

Remote wakeup packet filter register (ETH_MACRWKPFR) define the filtering management:

- Filter Byte Mask: determines which bytes of the packet must be examined
- Filter Offset: determines the offset from which the packet is to be examined
- Filter CRC-16

The remote wakeup CRC block determines the CRC value that is compared with Filter CRC-16. The remote wakeup packet is checked only for length error, FCS error, dribble bit error, receive error and collision. In addition, the remote wakeup packet is checked to ensure that it is not a runt packet. Even if the remote wakeup packet is more than 512 bytes long, if the packet has a valid CRC value, it is considered valid. The remote wakeup packet detection is updated in the PMT Control and Status register for every remote wakeup packet received. A PMT interrupt to the application triggers a Read to the PMT Control and Status register to determine reception of a remote wakeup packet.

Magic packet detection

The magic packet is based on a method that uses the magic packet technology from Advanced Micro Device to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

The MAC checks only those magic packets that are addressed to the MAC or a multicast address (including broadcast address) to determine whether these packets meet the wakeup requirements. The magic packets that pass the address filtering (unicast or multicast (including broadcast) address) are checked to determine whether they meet the remote wakeup packet data format of 6 bytes of all ones followed by a Unicast MAC Address (that matches the value in MAC Address 0) appearing 16 times.

The application enables the magic packet wakeup by writing 1 to the MGKPKTEN bit of the *PMT control status register (ETH_MACPCSR)*. The PMT block constantly monitors each packet addressed to the node for a specific magic packet pattern. Each packet received is checked for a 48’hFF_FF_FF_FF_FF_FF pattern following the destination and source address field. The PMT block then checks the packet for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again scans the 48’hFF_FF_FF_FF_FF_FF pattern in the incoming packet. The 16 repetitions can be anywhere in the packet, but must be preceded by the synchronization stream (48’hFF_FF_FF_FF_FF_FF). The device can also accept a multicast packet, as long as the 16 duplications of the MAC address are detected. If the number of repetitions of 8'hFF are more than 6, the PMT block checks for 16 repetitions of the MAC address without any breaks or interruptions, after the last 6 repetitions of 8'hFF.

If the MAC address of a node is 48'h00_11_22_33_44_55, the MAC scans for the following data sequence:

Destination Address Source Address …….. FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
…CRC

The MAC checks the remote wakeup packet only for length error, FCS error, dribble bit error, receive or collision error. In addition, the remote wakeup packet is checked to ensure

that it is not a runt packet. Even if the remote wakeup packet is more than 512 bytes long, if the packet has a valid CRC value, MAC considers it a valid packet.

The magic packet detection is updated in the *PMT control status register (ETH_MACPCSR)* for the received magic packet. A PMT interrupt to the Application triggers a read to the *PMT control status register (ETH_MACPCSR)* to determine whether a magic packet has been received.

PMT interrupts

The PMT interrupt signal is asserted when a valid remote wakeup packet is received.

When software resets the PWRDWN bit in *Remote wakeup packet filter register (ETH_MACRWKPFR)*, the MAC comes out of the power-down mode. This event does not generate the PMT interrupt.

As for EEE mode, a sideband signal, pmt_intr_o, is generated together with the wakeup interrupt. It is ORed with lpi_intr_o signal (see *Section : LPI Interrupt*) and tied to the EXTI peripheral (line 86).

57.7.3 Power-down and wakeup sequence

The recommended Power-down and wakeup sequence is as follows.

1. Disable the Transmit DMA and wait for any previous packet transmissions to complete. These transmissions can be detected when Transmit Interrupt (bit 0 of the ETH_DMACSR register) is received.
2. Disable the MAC transmitter and MAC receiver by clearing the corresponding bits (0 and 1) in the MAC Configuration register (ETH_MACCR).
3. Wait until the Receive DMA empties all the packets from the Rx FIFO to system memory. This can be done by reading the corresponding Debug register bits in the DMA (ETH_DMADSR) and MTL (ETH_MTLTXQDR and ETH_MTLRXQDR).
4. Enable Power-down mode by appropriately configuring the PMT registers ETH_MACPCSR and ETH_MACRWKPFR.
5. Enable the MAC Receiver and enter Power-down mode.
6. When receiving a valid remote wakeup packet, the Ethernet peripheral asserts the pmt_intr_o signal and exits Power-down mode.
7. Read the PMT Status register to clear the interrupt, then enable the other modules in the system and resume normal operation.

57.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the *Interrupt status register (ETH_DMAISR)* register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMACSR: Channel Status (see *Channel status register (ETH_DMACSR)*))
- ETH_MTLISR: Interrupt Status (see *Interrupt status Register (ETH_MTLISR)*)
- ETH_MACISR: Interrupt Status (see *Interrupt status register (ETH_MACISR)*)

57.8.1 DMA interrupts

Interrupt registers description

The ETH_DMACSR: Channel Status register (see *Channel status register (ETH_DMACSR)*) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMACIER: Channel Interrupt Enable register (see *Channel interrupt enable register (ETH_DMACIER)*) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts.They are indicated by Bits[15:14] of ETH_DMACSR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the *Interrupt status register (ETH_DMAISR)* for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of *Interrupt status register (ETH_DMAISR)* are cleared.

Periodic scheduling of Transmit and Receive Interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in *Table 526: TDES2 normal descriptor (read format)*) once for every “required” number of packets to be transmitted.
2. Similarly, set the IOC (RDES3[30] in*Table 539: RDES3 normal descriptor (read format)*) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.

In addition to above, an interrupt timer (ETH_DMACRXIWTR: Channel Rx Interrupt Watchdog Timer) is given for flexible control and periodic scheduling of Receive Interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in *Table 539: RDES3 normal descriptor (read format)*) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMACIER register (see *Channel interrupt enable register (ETH_DMACIER)*). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Channel Transfer Complete Interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in the Channel Status register (*Channel status register (ETH_DMACSR)*). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30])).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (*Channel interrupt enable register (ETH_DMACIER)*).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bit[17:16]) in the ETH_DMAMR register (*DMA mode register (ETH_DMAMR)*). *Table 523* explains the behavior of the Transfer Complete interrupt.

Table 523. Transfer complete interrupt behavior

Interrupt Mode	Behavior of TI/RI and interrupt signal
INTM=0	The TI/RI status signals are set whenever the Transfer complete event is detected. These bits are cleared whenever the software driver writes 1 to these bits. The interrupt signal is asserted whenever the corresponding interrupts are also enabled in ETH_DMACIER register.
INTM=1	The TI/RI is set as explained above. However, the interrupt is not asserted for any RI/TI event.
INTM=2	The RI/TI status bits are set whenever the Transfer Complete event is detected and are reset whenever software driver clears them by writing 1. However, if another Transfer complete event is detected before it is cleared (serviced) by the software, then these status bits are automatically set again. However, the interrupt is not generated based on TI/RI.

57.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

The register *Interrupt status Register (ETH_MTLISR)* report the queue number responsible for the event. ETH_MTLQICSR: Queue Interrupt Control Status shall be read for event description.

The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status Register (ETH_MTLISR)* register.

MTL interrupt signal is driven by one of these events:

- Receive Queue Overflow Interrupt
- Transmit Queue Underflow

57.8.3 MAC Interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MACISR)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:

- Receive Status Interrupt
- Transmit Status Interrupt
- Timestamp Interrupt Status
- MMC Interrupt Status
  - MMC Receive Checksum Offload Interrupt Status
  - MMC Transmit Interrupt Status
  - MMC Receive Interrupt Status
- LPI Interrupt Status
- PMT Interrupt Status
- PHY Interrupt

Note: Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wakeup event detection at EXTI level.

57.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see *Section 57.9.1*)
- MTL initialization (see *Section 57.9.2*)
- MAC initialization (see *Section 57.9.3*)
- Performing Normal Receive and Transmit Operation (see *Section 57.9.4*)
- Stopping and Starting Transmission (see *Section 57.9.5*)
- Programming Guidelines for MII Link State Transitions (see *Section 57.9.6*)
- Programming Guidelines for IEEE 1588 Timestamping (see *Section 57.9.7*)
- Programming Guidelines for Energy Efficient Ethernet (see *Section 57.9.8*)
- Programming Guidelines for flexible pulse-per-second (PPS) output (see *Section 57.9.9*)
- Programming Guidelines for TSO (see *Section 57.9.10*)
- Programming Guidelines for VLAN filtering on Receive (see *Section 57.9.11*)

57.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of *DMA mode register (ETH_DMAMR)*).
2. Wait for the completion of the reset process (poll bit 0 of the *DMA mode register (ETH_DMAMR)*, which is cleared when the reset operation is completed).
3. Program the following fields to initialize the *System bus mode register (ETH_DMASBMR)*:
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values in case of AHB bus interface.
   d) If fixed length value is enabled, select the maximum burst length possible on the AXI Bus (bits [7:1])
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to *Section 57.10: Descriptors*.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMACTXRLR and ETH_DMACRXRLR registers (see *Channel Tx descriptor ring length register (ETH_DMACTXRLR)* and *Channel Rx descriptor ring length register (ETH_DMACRXRLR)*). The programmed ring length must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (*Channel Tx descriptor list address register (ETH_DMACTXDLAR)*, *Channel Rx descriptor list address register (ETH_DMACRXDLAR)*). In addition, program the transmit and receive tail pointer registers that inform the DMA about the available descriptors (see *Channel Tx*

descriptor tail pointer register (ETH_DMACTXDTPR) and *Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)*).

7. Program ETH_DMACCR, ETH_DMACTXCR and ETH_DMACRXCR registers (see *Channel control register (ETH_DMACCR)* and *Channel transmit control register (ETH_DMACTXCR)*) to configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ for RxDMA.
8. Enable the interrupts by programming the ETH_DMACIER register (see *Channel interrupt enable register (ETH_DMACIER)*).
9. Start the Receive and Transmit DMAs by setting SR (bit 0) of *Channel receive control register (ETH_DMACRXCR)* and ST (bit 0) of the ETH_DMACTXCR (see *Channel transmit control register (ETH_DMACTXCR)*).

57.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the following fields to initialize the operating mode in the ETH_MTLTXQOMR (see *Tx queue operating mode Register (ETH_MTLTXQOMR)*).
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) if the Threshold mode is used.
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue 0.
   c) Transmit Queue Size (TQS).
2. Program the following fields to initialize the operating mode in the ETH_MTLRXQOMR register (see *Rx queue operating mode register (ETH_MTLRXQOMR)*):
   a) Receive Store and Forward (RSF) or RTC if Threshold mode is used.
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP).
   d) Receive Queue Size (RQS).

57.9.3 MAC initialization

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: *Address x low register (ETH_MACAxLR)* and *Address 0 high register (ETH_MACA0HR)*. If more than one MAC address is enabled in your configuration (up to 3 additional addresses), program the MAC addresses appropriately.
2. Program the following fields to set the appropriate filters for the incoming frames in the *Packet filtering control register (ETH_MACPFR)*:
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the *Tx Queue flow control register (ETH_MACQTXFCR)*:

a) Pause time and other Pause frame control bits.
b) Transmit Flow control bits.
c) Flow Control Busy.

4. Program the *Interrupt enable register (ETH_MACIER)* as required, if it is applicable for your configuration.
5. Program the appropriate fields in the *Operating mode configuration register (ETH_MACCR)* register.

   For example: Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in *Operating mode configuration register (ETH_MACCR)* register to start the MAC transmitter and receiver.

57.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the descriptors and writing the ETH_DMACTXDTPR (see *Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)* and ETH_DMACRXDTPR (see *Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)*).
4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see *Channel current application transmit descriptor register (ETH_DMACCATXDR)* and *Channel current application receive descriptor register (ETH_DMACCARXDR)*).
5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see *Channel current application transmit descriptor register (ETH_DMACCATXDR)* and *Channel current application receive descriptor register (ETH_DMACCARXDR)*).

57.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time :

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMACTXCR register (see *Channel transmit control register (ETH_DMACTXCR)*).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of ETH_MTLTXQDR register (TRCSTS is not 01 and TXQSTS=0).
3. Disable the MAC transmitter and MAC receiver by clearing Bit (RE) and Bit 1(TE) of the *Operating mode configuration register (ETH_MACCR)* Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of *Tx queue debug Register (ETH_MTLTXQDR)*, PRXQ=0 and RXQSTS=00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in *Tx queue debug Register (ETH_MTLTXQDR)* and RXQSTS is set to 0.
6. To restart the operation, first start the DMAs, and then enable the MAC Transmitter and Receiver.

57.9.6 Programming guidelines for MII link state transitions

Transmit and Receive clocks are running when the link is down

Complete the following steps when the link is down while the Transmit and Receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of *Channel control register (ETH_DMACCR)*.
2. Disable the MAC receiver by clearing bit 2 (RE) of *Operating mode configuration register (ETH_MACCR)*.
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue debug Register (ETH_MTLTXQDR)* (TRCSTS is not 01).

   or

   Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing Bit 1(TE) of the *Operating mode configuration register (ETH_MACCR)* Register.
5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in *Tx queue debug Register (ETH_MTLTXQDR)* and RXQSTS to 0 in *Rx queue debug register (ETH_MTLRXQDR)*).
6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.
7. Restart the operation by starting the Tx DMA. Then enable the MAC Transmitter and Receiver.

   The Rx DMA does not need to be enabled: since the Receiver is disabled, there are no data in the Rx FIFO.

Transmit and Receive clocks are stopped when the link is down

Complete the following steps when the link is down and the Transmit and Receive clocks are stopped :

1. Disable the MAC Transmitter and Receiver by clearing RE and TE bits in the *Operating mode configuration register (ETH_MACCR)*. This will not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the *Debug register (ETH_MACDR)* (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC Transmitter is stopped.
4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC Transmitter and Receiver by setting RE and TE bits.

57.9.7 Programming guidelines for IEEE 1588 timestamping

Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the *Timestamp control Register (ETH_MACTSCR)*. However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 16 of *Interrupt enable register (ETH_MACIER)*.
2. Set bit 0 of *Timestamp control Register (ETH_MACTSCR)* to enable timestamping.
3. Program *Sub-second increment register (ETH_MACSSIR)* based on the PTP clock frequency.
4. If you use the Fine Correction method, program *Timestamp addend register (ETH_MACTSAR)* and set bit 5 of *Timestamp control Register (ETH_MACTSCR)*.
5. Poll the *Timestamp control Register (ETH_MACTSCR)* until bit 5 is cleared.
6. Program bit 1 of *Timestamp control Register (ETH_MACTSCR)* to select the Fine Update method (if required).
7. Program *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)* with the appropriate time value.
8. Set bit 2 in *Timestamp control Register (ETH_MACTSCR)*.

   The timestamp counter starts as soon as it is initialized with the value written in the Timestamp Update registers. If one-step timestamping is required:

   a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.

   b) Program *Timestamp Ingress asymmetric correction register (ETH_MACTSIACR)* to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Note: If timestamp operation is disabled by clearing bit 0 of Timestamp control Register (ETH_MACTSCR), repeat all these steps to restart the timestamp operation.

System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the Timestamp Update registers (*System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*).
2. Set bit 3 (TSUPDT) of the *Timestamp control Register (ETH_MACTSCR)*.

   The value in the Timestamp Update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in *Section : System time register module*, calculate at which rate you intend increment or decrement the system time.
2. Update the *Timestamp addend register (ETH_MACTSAR)* with the new value and set bit 5 of the *Timestamp control Register (ETH_MACTSCR)* Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*.
5. Enable the Timestamp interrupt in bit 12 of *Interrupt enable register (ETH_MACIER)*.
6. Set bit 4 in Register *Timestamp control Register (ETH_MACTSCR)*.
7. When this trigger generates an interrupt, read *Interrupt status register (ETH_MACISR)*.
8. Reprogram *Timestamp addend register (ETH_MACTSAR)* with the old value and set bit 5 again.

57.9.8 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).
3. Program bits 25 to 16 and bits 1 to 0 in *LPI timers control register (ETH_MACLTCR)*.
4. Read the PHY link status by using the MDIO interface and update bit 17 of *LPI control status register (ETH_MACLCSR)*.
5. Update *LPI control status register (ETH_MACLCSR)* accordingly. This update should be done whenever the link status in the PHY chip changes.
6. Program *1-microsecond-tick counter register (ETH_MAC1USTCR)* as per the frequency of the clock used for accessing the CSR slave port.
7. Program the LPIET bit in the *LPI entry timer register (ETH_MACLETR)* with the IDLE time for which the MAC should wait before entering the LPI state on its own.

8. Set LPITE and LPITXA (bits 20 to 19) of *LPI control status register (ETH_MACLCSR)*.
9. Update *LPI control status register (ETH_MACLCSR)* to enable LPI auto-entry and MAC auto-exit from LPI state.
10. Program the *1-microsecond-tick counter register (ETH_MAC1USTCR)* according to the frequency of the clock used to access the CSR slave port.
11. Program the LPIET bit in *LPI entry timer register (ETH_MACLETR)* register with the IDLE time for which the MAC should wait before entering the LPI state on its own.
12. Set LPITE and LPITXA (bits 20 and 19) of *LPI control status register (ETH_MACLCSR)* to enable LPI auto-entry and MAC auto-exit from LPI state.
13. Set bit 16 of *LPI control status register (ETH_MACLCSR)* to put the MAC transmitter in LPI state.

    The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.
14. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC Transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.
15. The MAC Transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.
16. Reset LPITXEN bit if the application needs to override the auto-entry/exit modes and directly exit the MAC Transmitter from LPI state.

57.9.9 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:

1. Program TRGTMODSEL bit to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (register 736 and 737) as start time of PPS signal output.
2. Program the start time value in the Target Time registers (register 736 and 737).
3. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)* Register.
4. Program PPSCMD of *PPS control register (ETH_MACPPSCR)* to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the Target Time registers.

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has elapsed. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be higher than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in *PPS width register (ETH_MACPPSWR)*.
3. Program PPSCMD bits of *PPS control register (ETH_MACPPSCR)* to generate a single pulse after the previous pulse is de-asserted. This instructs the MAC to generate

a single pulse on the PPS signal output at the time programmed in Target Time registers.

If this command is given before the previous pulse becomes low, then the new command overwrites the previous command and the QOS may generate only 1 extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:

1. Program TRGTMODSEL bits to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (register 736 and 737) for start time of the PPS signal output.
2. Program the start time value in the Target Time registers (register 736 and 737).
3. Program the interval value between the train of pulses on the PPS signal output in *PPS interval register (ETH_MACPPSIR)*.
4. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)*.
5. Program PPSCMD bits in *PPS control register (ETH_MACPPSCR)* to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

   By default, the PPS pulse train is free-running unless it is stopped by issuing a ‘STOP Pulse train at time’ or ‘STOP Pulse Train immediately’ commands.
6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in *PPS target time nanoseconds register (ETH_MACPPSTTNR)* is reset before programming the Target Time registers again.
7. Program the PPSCMD bits in *PPS control register (ETH_MACPPSCR)* to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

The pulse train can be stopped at any time by programming 0101 in the PPSCMD field.

Similarly, the Stop Pulse train command (given in Step 5) can be canceled by programming PPSCMD bits to 0110 before the time (programmed at step 6) has elapsed.

The pulse train generation can be stopped by programming PPSCMD to 0011 before the start time programmed at step 2) has elapsed.

Generating an interrupt without affecting the PPS

TRGTMODSEL bits in *PPS control register (ETH_MACPPSCR)* enable you to program the Target Time registers (register 736 and 737) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL bits of *PPS control register (ETH_MACPPSCR)* to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

   If TRGTMODSEL bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

57.9.10 Programming guidelines for TSO

Follow the steps below to program TSO:

1. Program TSE bit of the corresponding ETH_DMACTXCR register to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
   – MSS of ETH_DMACCR
   – or MSS in the context descriptor

   If MSS field is programmed in both ETH_DMACCR and in the context descriptor, the latest software programmed sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first descriptor. This buffer must not hold any payload bytes. The payload is allocated to Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.

57.9.11 Programming guidelines to perform VLAN filtering on the receive

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program *VLAN tag register (ETH_MACVTR)* for the following bit to select the filtering method:
   – ETV: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   – VTHM: VLAN Tag Hash Table Match Enable.
   – ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   – ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   – DOVLTC: Ignores VLAN Type for Tag Match
   – VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL bit in *VLAN tag register (ETH_MACVTR)* for the 12-bit or 16-bit VLAN tag.
3. If VLAN tag Hash filtering is enabled, program *VLAN Hash table register (ETH_MACVHTR)*. The upper four bits of the calculated CRC are used to index the

contents of the VLAN Hash table. For example, a Hash value of '1000 selects bit 8 of the VLAN Hash table.

57.10 Descriptors

57.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- Normal descriptors

  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.
- **Context descriptors**

  The context descriptors are used to provide control information applicable to the packet to be transmitted.

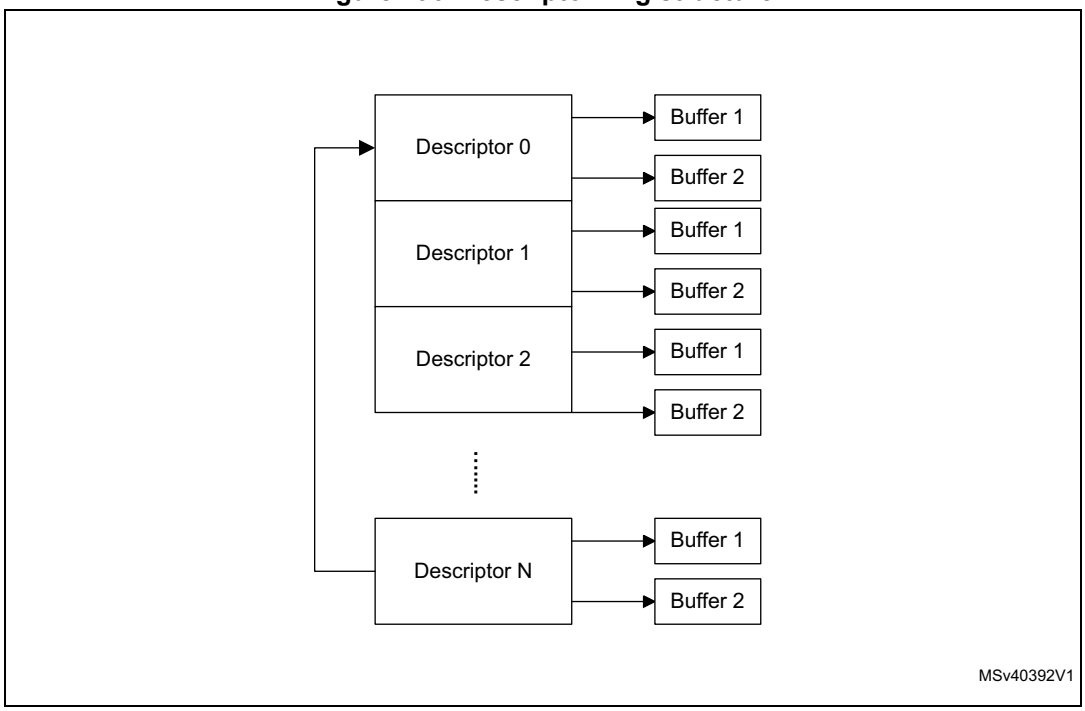
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.

57.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

Figure 796. Descriptor ring structure



In a ring structure, descriptors are separated by the 32-bit word number programmed in the DSL field of the *Channel control register (ETH_DMACCR)*). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- *Channel Tx descriptor ring length register (ETH_DMACTXRLR)*)
- *Channel Rx descriptor ring length register (ETH_DMACRXRLR)*)

The *Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)* or *Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)* contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

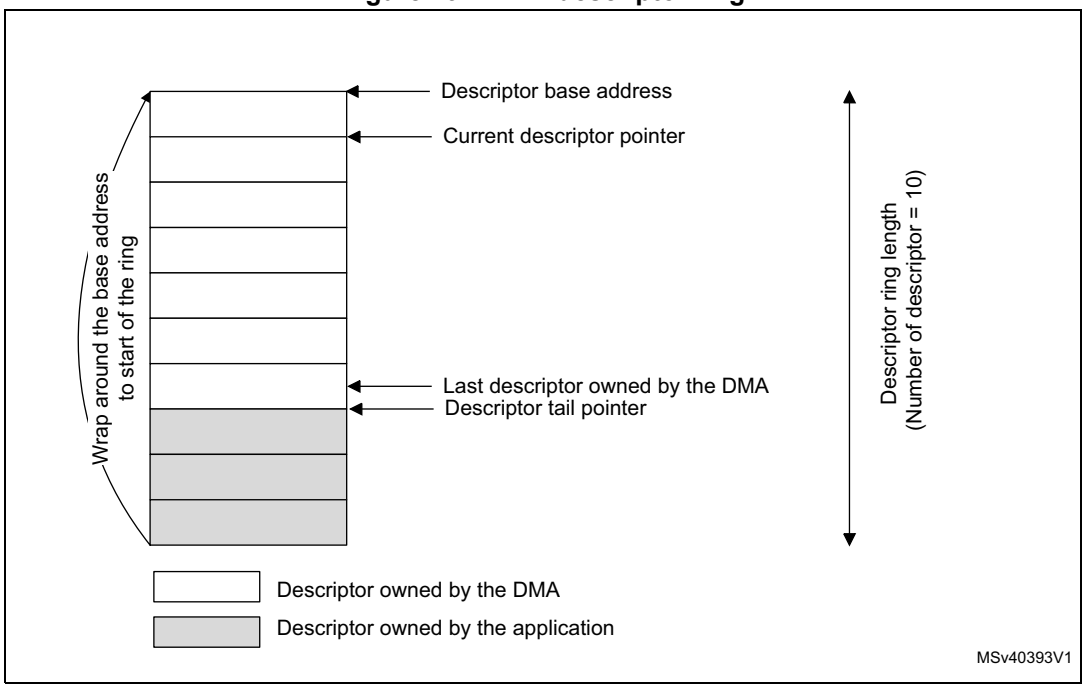
```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the Descriptor tail pointer register and update the tail pointer so that the following condition is met:

```
Current Descriptor Pointer < Descriptor Tail Pointer;
```

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 797: DMA descriptor ring*.

Figure 797. DMA descriptor ring



For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.

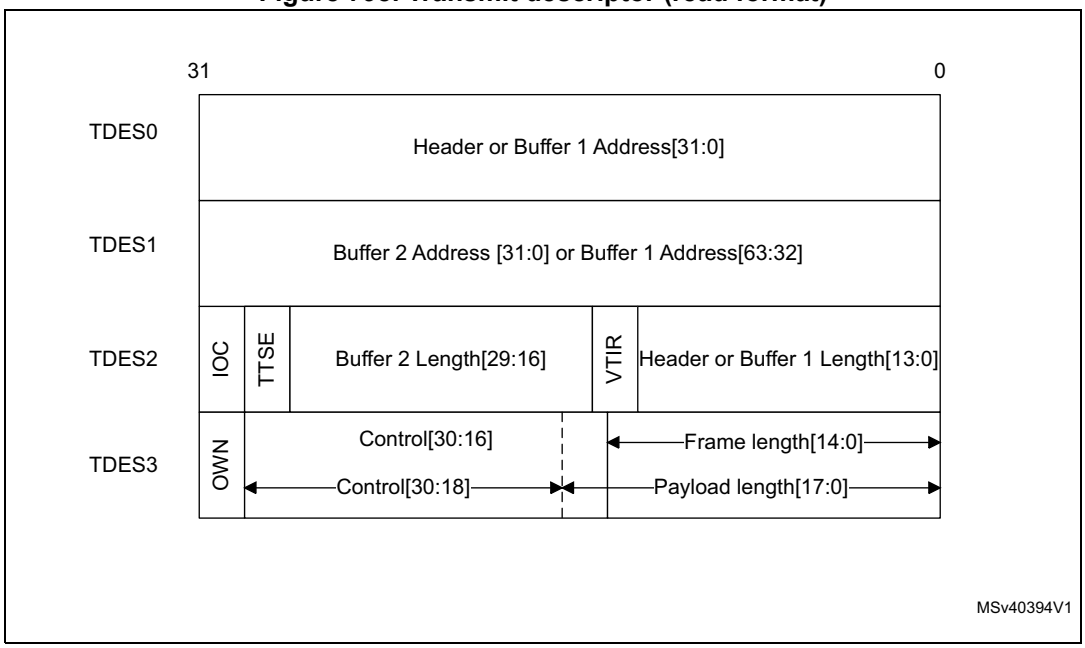
57.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format

Transmit normal descriptor (read format)

Figure 798 shows the Read format for Transmit normal descriptor. *Table 524* to *Table 527* provide a detailed description of all Transmit normal descriptors (read format).

Figure 798. Transmit descriptor (read format)



- TDES0 normal descriptor (read format)

Table 524. TDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer or TSO Header Address Pointer** These bits indicate either the physical address of Buffer 1 or the TSO Header Address pointer when the following bits are set: – TSE bit of TDES3 – FD bit of TDES3

- TDES1 normal descriptor (read format)

Table 525. TDES1 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 or Buffer 1 Address Pointer:** These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment.

- TDES2 normal descriptor (read format)

31	30	29:16	15:14	13:0
IOC	TTSE	B2L	VTIR	HL or B1L

Table 526. TDES2 normal descriptor (read format)

Bits	Name	Description
31	IOC	**Interrupt on Completion:** This bit sets the TI bit in the *Channel status register (ETH_DMACSR)*) when the present packet transmission is complete.
30	TTSE	**Transmit Timestamp Enable**
29:16	B2L	**Buffer 2 Length** The driver sets this field. When set, this field indicates Buffer 2 length.
15:14	VTIR	**VLAN Tag Insertion or Replacement:** These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows: – 00: Do not add a VLAN tag. – 01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. – 10: Insert a VLAN tag with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. – 11: Replace the VLAN tag in packets with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. This option should be used only with the VLAN packets.
13:0	HL or B1L	**Header Length or Buffer 1 Length** For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length. If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes. The maximum header length supported for TSO feature is 1023 bytes. If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length.

- TDES3 normal descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	CPC		SAIC			THL				TSE	TPL	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT/L															

Table 527. TDES3 normal descriptor (Read format)

Bits	Name	Description
31	OWN	**Own bit** – 1: the DMA owns the descriptor. – 0: the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptor.
29	FD	**First Descriptor** When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value.
27:26	CPC	**CRC Pad Control** This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following: – 00: CRC and Pad Insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. – 01: CRC Insertion (Disable Pad Insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. – 10: Disable CRC Insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. – 11: CRC Replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. *Note: When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.*

Table 527. TDES3 normal descriptor (Read format) (continued)

Bits	Name	Description
25:23	SAIC	**SA Insertion Control** These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[24:23]: – 00: Do not include the source address – 01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. – 10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. – 11: Reserved These bits are valid when the First Segment control bit (TDES3 [29]) is set.
22:19	THL	**THL**: TCP Header Length If the TSE bit is set, this field contains the length of the TCP header. The minimum value of this field must be 5.
18	TSE	**TCP Segmentation Enable** When this bit is set, the DMA performs the TCP segmentation for a packet. This bit is valid only if the FD bit is set.
17:16	CIC/TPL	**Checksum Insertion Control or TCP Payload Length** These bits control the checksum calculation and insertion. They can take the following values: – 00: Checksum insertion disabled. – 01: Only IP header checksum calculation and insertion are enabled. – 10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. – 11: IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the TSE bit is reset.When the TSE bit is set, it contains the upper bits [17:16] of the TCP Payload length. This allows the TCP packet length field to be spanned across TDES3[17:0] to provide 256 Kbyte packet length support.

Table 527. TDES3 normal descriptor (Read format) (continued)

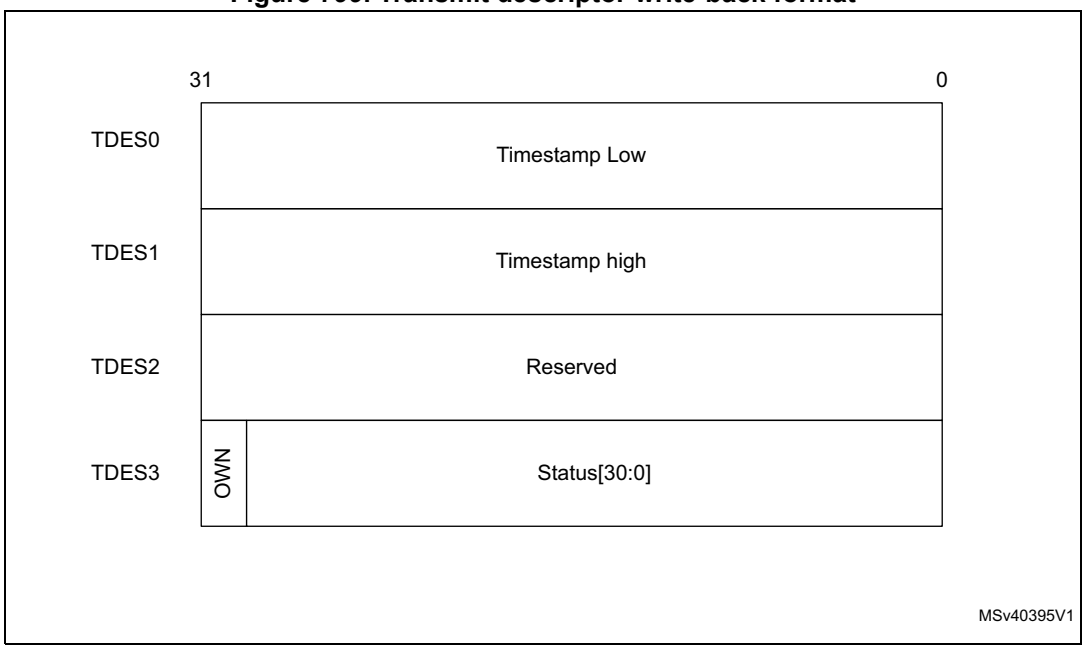
Bits	Name	Description
15	TPL	**Reserved or TCP Payload Length** When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0].
14:0	FL/TPL	**Packet Length or TCP Payload Length** This field is equal to the length of the packet to be transmitted (expressed in bytes). When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted: Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length. This length does not include Ethernet header or TCP/IP header length.

Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 799 shows the write-back format for Transmit normal descriptors. *Table 528* to *Table 531* provide a detailed description of all Transmit Normal descriptors (Write-Back Format).

Figure 799. Transmit descriptor write-back format



- TDES0 normal descriptor (write-back format)

Table 528. DES0 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES1 normal descriptor (write-back format)

Table 529. TDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES2 normal descriptor (write-back format)

Table 530. TDES2 normal descriptor (write-back format)(1)

Bit	Description
31:0	Reserved

1. This format is only applicable to the last descriptor of a packet.

- TDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	Reserved										TTSS	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	PCE	LoC	NC	LC	EC	CC				ED	UF	DB	IHE

Table 531. TDES3 normal descriptor (write-back format)(1)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptors.

Table 531. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
29	FD	**First Descriptor** This bit indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27:18	Reserved	
17	TTSS	**Tx Timestamp Status** This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set.
16	Reserved	
15	ES	**Error Summary:** This bit indicates the logical OR of the following bits: – TDES3[0]: IP Header Error – TDES3[14]: Jabber Timeout – TDES3[13]: Packet Flush – TDES3[12]: Payload Checksum Error – TDES3[11]: Loss of Carrier – TDES3[10]: No Carrier – TDES3[9]: Late Collision – TDES3[8]: Excessive Collision – TDES3[3]: Excessive Deferral – TDES3[2]: Underflow Error
14	JT	**Jabber Timeout** This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the *Operating mode configuration register (ETH_MACCR)* is not set.
13	FF	**Packet Flushed** This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	**Payload Checksum Error** This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode.

Table 531. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
11	LoC	**Loss of Carrier** This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
10	NC	**No Carrier** This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	**Late Collision** This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble. This bit is not valid if Underflow Error is set.
8	EC	**Excessive Collision** This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	**Collision Count** This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	**Excessive Deferral** This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times if DC bit is set in the *Operating mode configuration register (ETH_MACCR)*.
2	UF	**Underflow Error** This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: – The DMA encountered an empty Transmit Buffer while transmitting the packet – The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register.
1	DB	**Deferred Bit** This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
0	IHE	**IP Header Error** When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when Tx Checksum Offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

1. This format is only applicable to the last descriptor of a packet.

Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, VLAN Tag ID for VLAN insertion feature, and SA insertion bit for SA insertion. Write-back is only done on a context descriptor to reset the OWN bit.

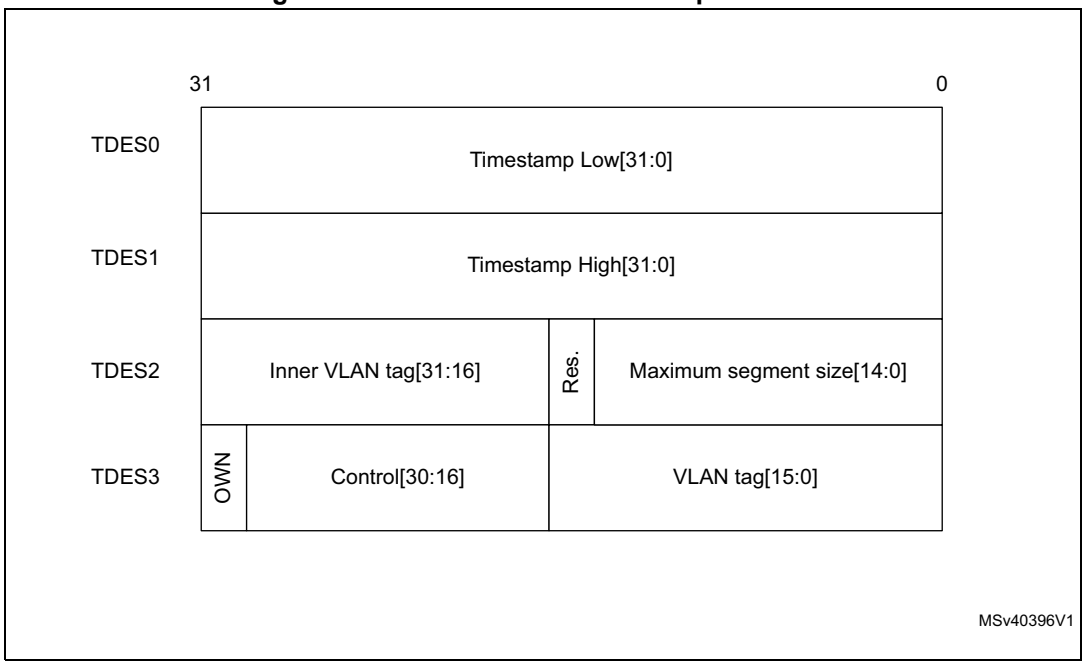
Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.

When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 800 shows the format for Transmit context descriptors. *Table 532* to *Table 535* provide a detailed description of all Transmit context descriptors.

Figure 800. Transmit context descriptor format



- TDES0 context descriptor (read format)

Table 532. TDES0 context descriptor

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)

Table 533. TDES1 context descriptor

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES2 context descriptor (read format)

Table 534. TDES2 context descriptor

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
15:14	Reserved	
13:0	MSS	**Maximum Segment Size** This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

- TDES3 context descriptor (read format)

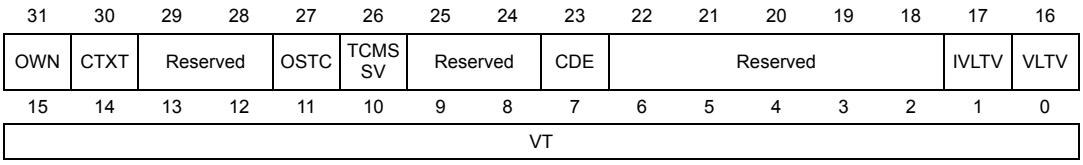


Table 535. TDES3 context descriptor

Bit	Name	Description
31	OWN	**Own bit** – 1: the DMA owns the descriptor. – 0: the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception. – The buffers associated with the descriptor are full.
30	CTXT	**Context Type** This bit should be set to 1 for context descriptor.
29:28	Reserved	
27	OSTC	**One-Step Timestamp Correction Enable** When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
26	TCMSSV	**One-Step Timestamp Correction Input or MSS Valid** When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	Reserved	
23	CDE	**Context Descriptor Error** When this bit is set, it indicates that the context descriptor was provided in the incorrect sequence and the DMA ignored it. The DMA sets this bit during write-back while closing the context descriptor.
22:20	Reserved	
19:18	IVTIR	**Inner VLAN Tag Insert or Replace** When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. This bitfield has the following values: – 00: Do not add the inner VLAN tag. – 01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. – 10: Insert an inner VLAN tag with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. – 11: Replace the inner VLAN tag in packets with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. This option should be used only with the VLAN frames.
17	IVLTV	**Inner VLAN Tag Valid** When this bit is set, it indicates that the IVT field of TDES2 is valid.

Table 535. TDES3 context descriptor (continued)

Bit	Name	Description
16	VLTV	**VLAN Tag Valid** When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	**VLAN Tag** This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the *VLAN inclusion register (ETH_MACVIR)* is reset.

57.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following Receive descriptors are present:

- Normal descriptors with read and write-back formats
- Context descriptors

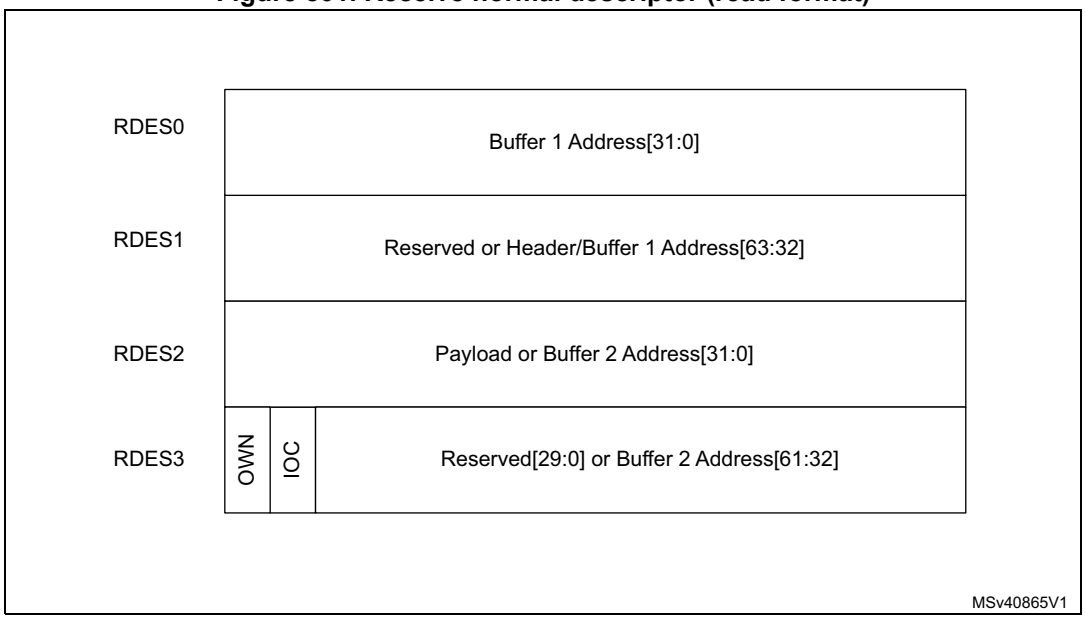
All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see *Figure 801: Receive normal descriptor (read format)* for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in *Figure 802: Receive normal descriptor (write-back format)*.

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA will write the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in *Figure 803: Receive context descriptor*.

Receive normal descriptor (read format)

Figure 801 shows the read format for Receive normal descriptors. *Table 536* to *Table 539* provide a detailed description of all Receive normal descriptors (read format).

Figure 801. Receive normal descriptor (read format)



Note: In the Receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.

- RDES0 normal descriptor (read format)

Table 536. RDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Header or Buffer 1 Address Pointer** The application can program a byte-aligned address for this buffer, which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES0[1:0] (or RDES0[2:0]/[3:0] in case of 64-/128-bit configuration) as zero. However, the packet data is shifted by the actual offset as given in the buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

- RDES1 normal descriptor (read format)

Table 537. RDES1 normal descriptor (read format)

Bit	Name	Description
31:0	Reserved	Field reserved.

- RDES2 normal descriptor (read format)

Table 538. RDES2 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 Address Pointer** These bits indicate Buffer 2 physical address. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores RDES2[1:0]=0 and writes to the complete location.

- RDES3 normal descriptor (read format)

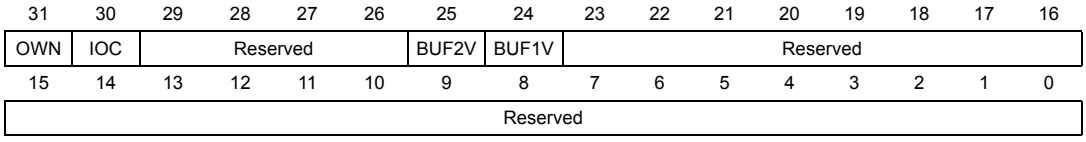


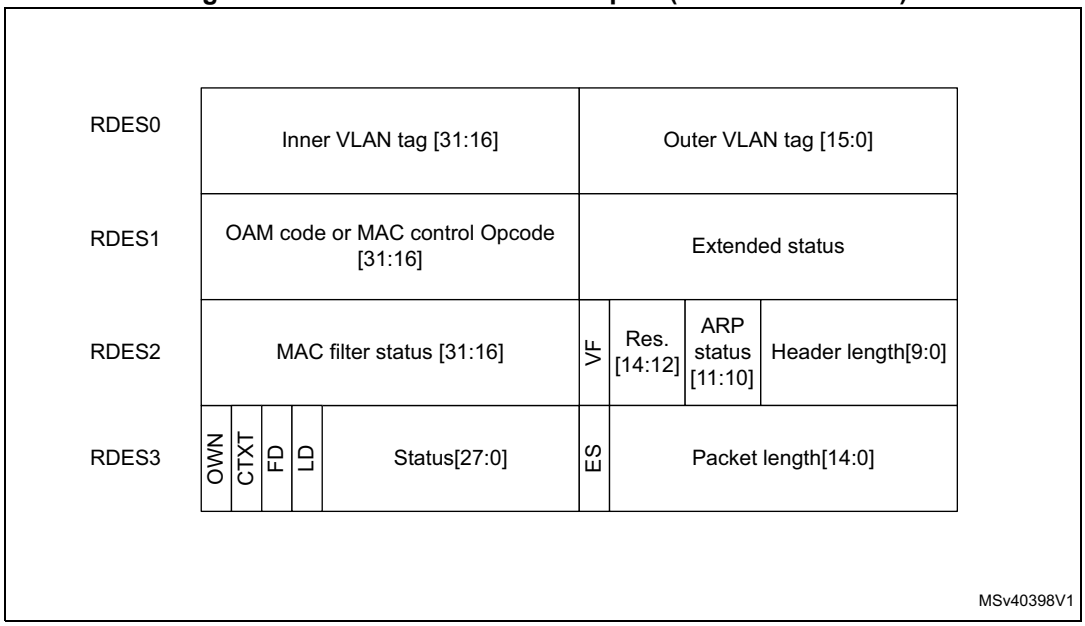
Table 539. RDES3 normal descriptor (read format)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full
30	IOC	**Interrupt Enabled on Completion** When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Reserved	
25	BUF2V	**Buffer 2 Address Valid** When this bit is set, it indicates to the DMA that the buffer 1 address specified in RDES0 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.
24	BUF1V	**Buffer 1 Address Valid** When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address to which Buffer 1 address points in RDES1, can be used by the DMA to write received packet data.
23:0	Reserved	

Receive normal descriptor (write-back format)

Figure 802 shows the write-back format for Receive normal descriptors. *Table 540* to *Table 543* provide a detailed description of all Receive normal descriptors (write-back format).

Figure 802. Receive normal descriptor (write-back format)



- RDES0 normal descriptor (write-back format)

Table 540. RDES0 normal descriptor (write-back format)

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set.
15:0	OVT	**Outer VLAN Tag** This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

- RDES1 normal descriptor (write-back format)

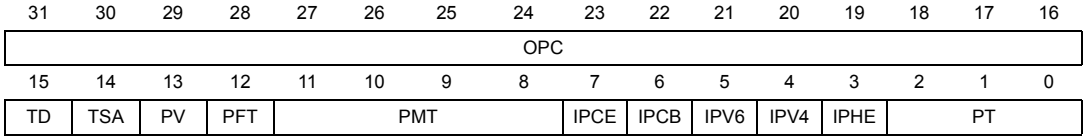


Table 541. RDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:16	OPC	**OAM Sub-Type Code, or MAC Control Packet opcode** OAM Sub-Type Code If bits[18:16] of RDES3 are set to 111, this field contains the OAM sub-type and code fields. MAC Control Packet opcode If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
15	TD	**Timestamp Dropped** This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.
14	TSA	**Timestamp Available** When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	**PTP Version** 1: Received PTP message in IEEE 1588 version 2 format 0: Received PTP message in IEEE 1588 version 1 format
12	PFT	**PTP Packet Type** This bit indicates that the PTP message is sent directly over Ethernet.
11:8	PMT	**PTP Message Type** These bits are encoded to give the type of the message received: – 0000: No PTP message received – 0001: SYNC (all clock types) – 0010: Follow_Up (all clock types) – 0011: Delay_Req (all clock types) – 0100: Delay_Resp (all clock types) – 0101: Pdelay_Req (in peer-to-peer transparent clock) – 0110: Pdelay_Resp (in peer-to-peer transparent clock) – 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) – 1000: Announce – 1001: Management – 1010: Signaling – 1011–1110: Reserved – 1111: PTP packet with Reserved message type These bits are available only when you select the timestamp feature.

Table 541. RDES1 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
7	IPCE	**IP Payload Error** When this bit is set, it indicates either of the following: – The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. – The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. – The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.
6	IPCB	**IP Checksum Bypassed** This bit indicates that the checksum offload engine is bypassed.
5	IPV6	**IPv6 header Present** This bit indicates that an IPV6 header is detected.
4	IPV4	**IPV4 Header Present** This bit indicates that an IPV4 header is detected.
3	IPHE	**IP Header Error** When this bit is set, it indicates either of the following: – The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. – The IP datagram version is not consistent with the Ethernet Type value. – Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.
2:0	PT	**Payload Type** These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): – 000: Unknown type or IP/AV payload not processed – 001: UDP – 010: TCP – 011: ICMP – Others: reserved. If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

- RDES2 normal descriptor (write-back format)

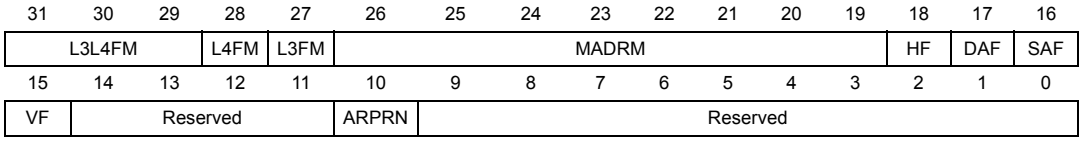


Table 542. RDES2 normal descriptor (write-back format)

Bit	Name	Description
31:29	L3L4FM	**Layer 3 and Layer 4 Filter Number Matched** These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: – 000: Filter 0 – 001: Filter 1 – 010: Filter 2 – 011: Filter 3 – 100: Filter 4 – 101: Filter 5 – 110: Filter 6 – 111: Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter.
28	L4FM	**Layer 4 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: – Layer 3 fields are not enabled and all enabled Layer 4 fields match – All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29].
27	L3FM	**Layer 3 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: – All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed – All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29].
26:19	MADRM	**MAC Address Match or Hash Value** When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register.
18	HF	**Hash Filter Status** When this bit is set, it indicates that the packet passed the MAC address Hash filter. its[26:19] indicate the Hash value.
17	DAF	**Destination Address Filter Fail** When this bit is set, it indicates that the packet failed the DA Filter in the MAC.

Table 542. RDES2 normal descriptor (write-back format) (continued)

Bit	Name	Description
16	SAF	**SA Address Filter Fail** When this bit is set, it indicates that the packet failed the SA Filter in the MAC.
15	VF	**VLAN Filter Status** When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter.
14:11	Reserved	
10	ARPNR	**ARP Reply Not Generated** When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
9:0	Reserved	

- RDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	PL														

Table 543. RDES3 normal descriptor (write-back format)

Bit	Name	Description
31	OWN	**Own bit** 1: The DMA owns the descriptor. 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor.
29	FD	**First Descriptor** When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.

Table 543. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
27	RS2V	**Receive Status RDES2 Valid** When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	**Receive Status RDES1 Valid** When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	**Receive Status RDES0 Valid** When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
24	CE	**CRC Error** When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	**Giant Packet** When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). *Note: Giant packet indicates only the packet length. It does not cause any packet truncation.*
22	RWT	**Receive Watchdog Timeout** When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	**Overflow Error** When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. *Note: This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.*
20	RE	**Receive Error** When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception.
19	DE	**Dribble Bit Error** When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.

Table 543. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
18:16	LT	**Length/Type Field** This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: – 000: The packet is a length packet – 001: The packet is a type packet. – 011: The packet is a ARP Request packet type – 100: The packet is a type packet with VLAN Tag – 101: The packet is a type packet with Double VLAN Tag – 110: The packet is a MAC Control packet type – 111: The packet is a OAM packet type – 010: Reserved
15	ES	**Error Summary** – When this bit is set, it indicates the logical OR of the following bits: – RDES3[24]: CRC Error – RDES3[19]: Dribble Error – RDES3[20]: Receive Error – RDES3[22]: Watchdog Timeout – RDES3[21]: Overflow Error – RDES3[23]: Giant Packet This field is valid only when the LD bit of RDES3 is set.
14:0	PL	**Packet Length** These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and either the Descriptor Error (RDES3[13]) or Overflow Error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.

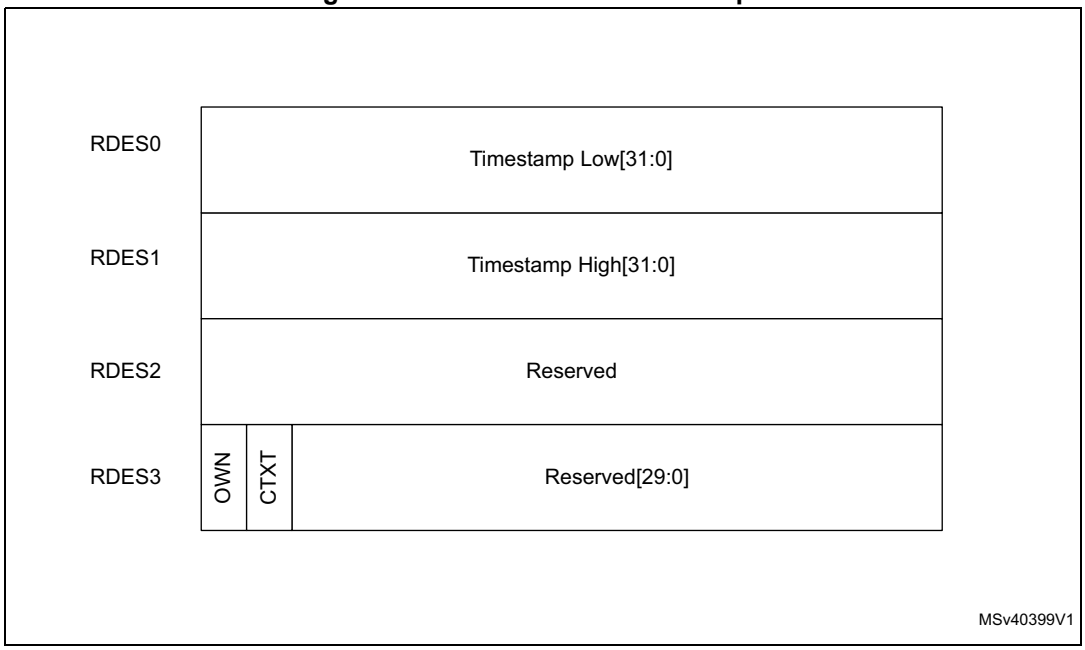
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

Figure 803 shows the format for Receive context descriptors. *Table 544* to *Table 547* provide a detailed description of all Receive context descriptors.

Figure 803. Receive context descriptor



- RDES0 context descriptor

Table 544. RDES0 context descriptor

Bit	Name	Description
31:0	RTSL	**Receive Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

- RDES1 context descriptor

Table 545. RDES1 context descriptor

Bit	Field	Description
31:0	RTSH	**Receive Packet Timestamp High** The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

- RDES2 context descriptor

Table 546. RDES2 context descriptor

Bit	Description
31:0	Reserved

- RDES3 context descriptor

Table 547. RDES3 context descriptor

Bit	Name	Description
31	OWN	**Own Bit** 1; The DMA owns the descriptor 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor.
29:0	Reserved	

57.11 Ethernet registers

57.11.1 Ethernet registers maps

This section provides the following register maps:

- DMA registers (see *Section 57.11.2: Ethernet DMA registers*)
- MTL registers (see *Section 57.11.3: Ethernet MTL registers*)
- MAC registers including MMC register (see *Section 57.11.4: Ethernet MAC and MMC registers*)

57.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000

Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PR[2:0]			TXPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.			DA	SWR
	r	r	r	r							r	r	r	r	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **INTM[1:0]**: Interrupt Mode

This field defines the interrupt mode of the Ethernet peripheral.
The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMACSR register changes depending on the INTM value (refer to *Table 523: Transfer complete interrupt behavior*).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PR[2:0]**: Priority ratio

These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set.
000: The priority ratio is 1:1
001: The priority ratio is 2:1
010: The priority ratio is 3:1
011: The priority ratio is 4:1
100: The priority ratio is 5:1
101: The priority ratio is 6:1
110: The priority ratio is 7:1
111: The priority ratio is 8:1

Bit 11 **TXPR**: Transmit priority

When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.

Bits 10:2 Reserved, must be kept at reset value.

Bit 1 **DA**: DMA Tx or Rx Arbitration Scheme

This bit specifies the arbitration scheme between the Transmit and Receive paths of all channels:

0: Weighted Round-Robin with Rx:Tx or Tx:Rx

The priority between the paths is according to the priority specified in Bits[14:12] and the priority weight is specified in the TXPR bit.

1: Fixed priority

The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path.

Bit 0 **SWR**: Software Reset

When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all clock domains. Before reprogramming any register, a value of zero should be read in this bit.

Note: The reset operation is complete only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.

System bus mode register (ETH_DMASBMR)

Address offset: 0x1004

Reset value: 0x0101 0000

The System bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB	MB	Res.	AAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FB
r	r		rw												rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RB**: Rebuild INCRx Burst

When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.

Bit 14 **MB**: Mixed Burst

When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **AAL**: Address-Aligned Beats

When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **FB**: Fixed Burst Length

When this bit is set to 1, the AHB master will initiate burst transfers of specified length (INCRx or SINGLE).

When this bit is set to 0, the AHB master will initiate transfers of unspecified length (INCR) or SINGLE transfers.

Interrupt status register (ETH_DMAISR)

Address offset: 0x1008

Reset value: 0x0000 0000

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC0IS
															r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **MACIS**: MAC Interrupt Status

This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

Bit 16 **MTLIS**: MTL Interrupt Status

This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **DC0IS**: DMA Channel Interrupt Status

This bit indicates an interrupt event in DMA Channel. To reset this bit to 0, the software must read the corresponding register in DMA Channel to get the exact cause of the interrupt and clear its source.

Debug status register (ETH_DMADSR)

Address offset: 0x100C

Reset value: 0x0000 0000

The Debug status register gives the Receive and Transmit process status for DMA Channel for debugging purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				Res.			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXW HSTS

															r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **TPS0[3:0]**: DMA Channel Transmit Process State
This field indicates the Tx DMA FSM state for Channel:
000: Stopped (Reset or Stop Transmit Command issued)
001: Running (Fetching Tx Transfer Descriptor)
010: Running (Waiting for status)
011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
100: Timestamp write state
101: Reserved for future use
110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
111: Running (Closing Tx Descriptor)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPS0[3:0]**: DMA Channel Receive Process State
This field indicates the Rx DMA FSM state for Channel:
000: Stopped (Reset or Stop Receive Command issued)
001: Running (Fetching Rx Transfer Descriptor)
010: Reserved for future use
011: Running (Waiting for Rx packet)
100: Suspended (Rx Descriptor Unavailable)
101: Running (Closing the Rx Descriptor)
110: Timestamp write state
111: Running (Transferring the received packet data from the Rx buffer to the system memory)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **AXWHSTS**: AHB Master Write Channel
When high, this bit indicates that the write channel of the AHB master FMSs are in non-idle state.

Channel control register (ETH_DMACCR)

Address offset: 0x1100

Reset value: 0x0000 0000

The DMA Channel Control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBL X8
											rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Res.	Res.	MSS[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:18 **DSL[2:0]**: Descriptor Skip Length

This bit specifies the 32-bit word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.

When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PBLX8**: 8xPBL mode

When this bit is set, the PBL value programmed in Bits[21:16] in ETH_DMACTXCR is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **MSS[13:0]**: Maximum Segment Size

This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of ETH_DMACTXCR register is set.

The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

This field is reserved when the Enable TCP Segmentation Offloading for TCP/IP Packets option is not selected.

Channel transmit control register (ETH_DMACTXCR)

Address offset: 0x1104

Reset value: 0x0000 0000

The DMA Channel Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	TXPBL[5:0]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	Res.	Res.	Res.	ST
			rw								rw				rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **TXPBL[5:0]**: Transmit Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

1. Set the PBLx8 mode in ETH_DMACCR.
2. Set the PBL.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TSE**: TCP Segmentation Enabled

When this bit is set, the DMA performs the TCP segmentation for packets in Channel i. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

This field is reserved if Enable TCP Segmentation Offloading for TCP/IP Packets option is not selected.

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **OSF**: Operate on Second Packet

When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ST**: Start or Stop Transmission Command

When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted.

The DMA tries to acquire descriptor from either of the following positions:

- The current position in the list: this is the base address of the Transmit list set by the ETH_DMACTXDLAR register.
- The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMACSR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMACTXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMACTXDLAR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

Channel receive control register (ETH_DMACRXCR)

Address offset: 0x1108

Reset value: 0x0000 0000

The DMA Channel Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBL[5:0]					
rw										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RBSZ[13:0]														SR
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RPF**: DMA Rx Channel Packet Flush

When this bit is set to 1, the DMA will automatically flush the packet from the Rx queues destined to DMA Rx Channel when the DMA Rx Channel is stopped after a system bus error has occurred.
The flushing happens on the Read side of the Rx queue.
When this bit is set to 0 the EQOS will not flush the packet in the Rx queue destined to DMA Rx Channel after the DMA is stopped due to a system bus error.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:16 **RXPBL[5:0]**: Receive Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.
To transfer more than 32 beats, perform the following steps:
1. Set the PBLx8 mode in the ETH_DMACCR.
2. Set the PBL.

Bit 15 Reserved, must be kept at reset value.

Bits 14:1 **RBSZ[13:0]**: Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes. The buffer size is applicable to payload buffers when split headers are enabled.

Note: The buffer size must be a multiple of 4, 8, or 16 depending on the bus widths (32, 64, or 128 respectively). This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 4, 8, or 16, it may result into undefined behavior.

The LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0 **SR**: Start or Stop Receive

When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the address set by the ETH_DMACRXDLAR register.

– The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMACSR is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the ETH_DMACRXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

Channel Tx descriptor list address register (ETH_DMACTXDLAR)

Address offset: 0x1114

Reset value: 0x0000 0000

Channel Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, the ST bit is set to zero in ETH_DMACiTXCR register. When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly-programmed descriptor base address. If this register is not changed when the ST bit is set to 0, the DMA takes the descriptor address where it was stopped earlier.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDESLA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **TDESLA[31:2]**: Start of Transmit List

This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

The width of this field depends on the configuration:

31:2 for 32-bit configuration

Bits 1:0 Reserved, must be kept at reset value.

Channel Rx descriptor list address register (ETH_DMACRXDLAR)

Address offset: 0x111C

Reset value: 0x0000 0000

The Channel Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in ETH_DMACRXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDESLA[29:14]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDESLA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **RDESLA[29:0]**: Start of Receive List

This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

The width of this field depends on the configuration:

31:2 for 32-bit configuration

Bits 1:0 Reserved, must be kept at reset value.

Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)

Address offset: 0x1120

Reset value: 0x0000 0000

The ChannelTx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDT[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **TDT[31:2]**: Transmit Descriptor Tail Pointer

This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

The width of this field depends on the configuration:

31:2 for 32-bit configuration

Bits 1:0 Reserved, must be kept at reset value.

Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)

Address offset: 0x1128

Reset value: 0x0000 0000

The Channel Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDT[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **RDT[**31:2**]**: Receive Descriptor Tail Pointer

This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.
The width of this field depends on the configuration:
31:2 for 32-bit configuration

Bits 1:0 Reserved, must be kept at reset value.

Channel Tx descriptor ring length register (ETH_DMACTXRLR)

Address offset: 0x112C

Reset value: 0x0000 0000

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TDRL[9:0]**: Transmit Descriptor Ring Length

This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. It is recommended to put a minimum ring descriptor length of 4.

Channel Rx descriptor ring length register (ETH_DMACRXRLR)

Address offset: 0x1130

Reset value: 0x0000 0000

The Channel Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **RDRL[9:0]**: Receive Descriptor Ring Length

This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors.

Channel interrupt enable register (ETH_DMACIER)

Address offset: 0x1134

Reset value: 0x0000 0000

The Channel Interrupt Enable register enables the interrupts reported by the Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NIE**: Normal Interrupt Summary Enable

When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the ETH_DMACSR:
Bit 0: Transmit Interrupt
Bit 2: Transmit Buffer Unavailable
Bit 6: Receive Interrupt
Bit 11: Early Receive Interrupt
When this bit is reset, the normal interrupt summary is disabled.

Bit 14 **AIE**: Abnormal Interrupt Summary Enable

When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the ETH_DMACSR:
Bit 1: Transmit Process Stopped
Bit 7: Rx Buffer Unavailable
Bit 8: Receive Process Stopped
Bit 9: Receive Watchdog Timeout
Bit 10: Early Transmit Interrupt
Bit 12: Fatal Bus Error
When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 **CDEE**: Context Descriptor Error Enable

When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 **FBEE**: Fatal Bus Error Enable

When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 **ERIE**: Early Receive Interrupt Enable

When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled.

Bit 10 **ETIE**: Early Transmit Interrupt Enable

When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 **RWTE**: Receive Watchdog Timeout Enable

When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled.

Bit 8 **RSE**: Receive Stopped Enable

When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.

Bit 7 **RBUE**: Receive Buffer Unavailable Enable

When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.

Bit 6 **RIE**: Receive Interrupt Enable

When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBUE**: Transmit Buffer Unavailable Enable

When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

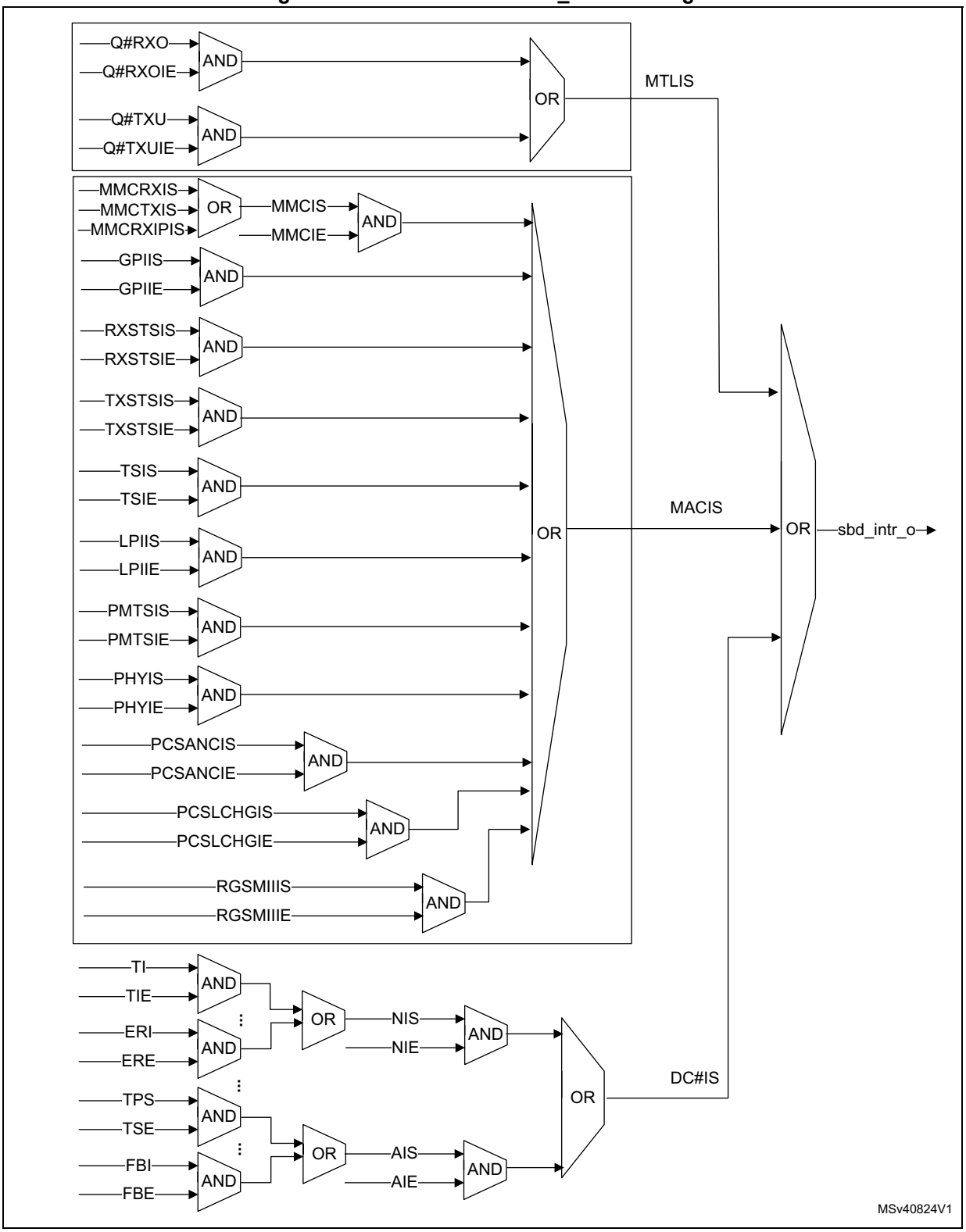
Bit 1 **TXSE**: Transmit Stopped Enable

When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0 **TIE**: Transmit Interrupt Enable

When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.

Figure 804. Generation of ETH_DMAISR flags



Channel Rx interrupt watchdog timer register (ETH_DMACRXIWTR)

Address offset: 0x1138

Reset value: 0x0000 0000

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the ETH_DMACSR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RWT[7:0]**: Receive Interrupt Watchdog Timer Count

This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.

The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMACSR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].

When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

Channel current application transmit descriptor register (ETH_DMACCATXDR)

Address offset: 0x1144

Reset value: 0x0000 0000

The Channel Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTDESAPTR[31:0]**: Application Transmit Descriptor Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel current application receive descriptor register (ETH_DMACCARXDR)

Address offset: 0x114C

Reset value: 0x0000 0000

The Channel Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRDESAPTR[31:0]**: Application Receive Descriptor Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel current application transmit buffer register (ETH_DMACCATXBR)

Address offset: 0x1154

Reset value: 0x0000 0000

The Channel Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTBUFAPTR[31:0]**: Application Transmit Buffer Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel current application receive buffer register (ETH_DMACCARXBR)

Address offset: 0x115C

Reset value: 0x0000 0000

The Channel Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRBUFAPTR[31:0]**: Application Receive Buffer Address Pointer

The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel status register (ETH_DMACSR)

Address offset: 0x1160

Reset value: 0x0000 0000

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]		
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **REB[2:0]**: Rx DMA Error Bits

This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface.
Bit 21
1: Error during data transfer by Rx DMA
0: No Error during data transfer by Rx DMA
Bit 20
1: Error during descriptor access
0: Error during data buffer access
Bit 19
1: Error during read transfer
0: Error during write transfer
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bits 18:16 **TEB[2:0]**: Tx DMA Error Bits

This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface.
Bit 18
1: Error during data transfer by Tx DMA
0: No Error during data transfer by Tx DMA
Bit 17
1: Error during descriptor access
0: Error during data buffer access
Bit 16
1: Error during read transfer
0: Error during write transfer
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15 **NIS**: Normal Interrupt Summary

Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
Bit 0: Transmit Interrupt
Bit 2: Transmit Buffer Unavailable
Bit 6: Receive Interrupt
Bit 11: Early Receive Interrupt
Only unmasked bits (interrupts for which interrupt enable is set in ETH_DMACIER register) affect the Normal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14 **AIS**: Abnormal Interrupt Summary

Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
Bit 1: Transmit Process Stopped
Bit 7: Receive Buffer Unavailable
Bit 8: Receive Process Stopped
Bit 10: Early Transmit Interrupt
Bit 12: Fatal Bus Error
Bit 13: Context Descriptor Error
Only unmasked bits affect the Abnormal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13 **CDE**: Context Descriptor Error

This bit indicates that the DMA Tx engine received a context descriptor in the middle of a packet (in an intermediate descriptor), and the DMA Tx engine ignored it.

Bit 12 **FBE**: Fatal Bus Error

This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.

Bit 11 **ERI**: Early Receive Interrupt

This bit indicates that the DMA filled the first data buffer of the packet. The RI bit of this register automatically clears this bit.

Bit 10 **ETI**: Early Transmit Interrupt

This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO.

Bit 9 **RWT**: Receive Watchdog Timeout

This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.

Bit 8 **RPS**: Receive Process Stopped

This bit is asserted when the Rx process enters the Stopped state.

Bit 7 **RBU**: Receive Buffer Unavailable

This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor.

Bit 6 **RI**: Receive Interrupt

This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

The reception remains in the Running state.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBU**: Transmit Buffer Unavailable

This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPS0 field of the DMA_Debug_Status0 register explains the Transmit Process state transitions.

To resume processing the Transmit descriptors, the application should do the following:

1. Change the ownership of the descriptor by setting Bit 31 of TDES3.
2. Issue a Transmit Poll Demand command.

For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel.

Bit 1 **TPS**: Transmit Process Stopped

This bit is set when the transmission is stopped.

Bit 0 **TI**: Transmit Interrupt

This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

Channel missed frame count register (ETH_DMACMFCR)

Address offset: 0x116C

Reset value: 0x0000 0000

This register has the number of packet counter that got dropped by the DMA either due to Bus Error or due to programing RPF field in ETH_DMACRXCR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
r					r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MFCO**: Overflow status of the MFC Counter

When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MFC[10:0]**: Dropped Packet Counters

This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing
RPF field in ETH_DMACRXCR register. The counter gets cleared when this register is read.

Ethernet DMA register map and reset values

Table 548. ETH_DMA common register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000	ETH_DMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM		Res.	PR[2:0]			TXPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.			DA	SWR
	Reset value															0	0		0	0	0	0							0	0	0	0	0
0x1004	ETH_DMASBMR	Res.	Res.	Res.	Res.	Res.				Res.	Res.	Res.	Res.	RD_OSR_LMT[1:0]				RB	MB	Res.	AAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FB
	Reset value													0	0	0	1	0	0		0												0
0x1008	ETH_DMAISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC0IS
	Reset value															0	0																0
0x100C	ETH_DMADSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				Res.				TPS0				RPS0				Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXWHSTS
	Reset value																	0	0	0	0	0	0	0	0								0
0x1010 - 0x101C	Reserved																																

Table 549. ETH_DMA_CH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1100	ETH_DMACCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL			Res.	PBLX8	Res.	Res.	MSS													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1104	ETH_ DMACTXCR	Res.	Res.	Res.	Res.	Res.				Res.	Res.	TXPBL[5:0]						IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	Res.			ST
	Reset value											0	0	0	0	0	0	0			0									0				0
0x1108	ETH_ DMACRXCR	RPF	Res.	Res.	Res.	Res.				Res.	Res.	RXPBL[5:0]						Res.	RBSZ[10:0]											Res.	Res.	Res.	SR
	Reset value	0										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0				0
0x110C - 01110	Reserved																																

Table 549. ETH_DMA_CH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1114	ETH_ DMACTXDLAR	TDESLA																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x1118	Reserved																																
0x111C	ETH_ DMACRXDLAR	RDESLA																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x1120	ETH_ DMACTXDTPR	TDT																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x1124	Reserved																																
0x1128	ETH_ DMACRXDTPR	RDT																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x112C	ETH_ DMACTXDRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1130	ETH_ DMACRXDRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1134	ETH_DMACIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1138	ETH_ DMACRXIWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x1140	Reserved																																
0x1144	ETH_ DMACCATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1148	Reserved																																
0x0114C	ETH_ DMACCARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1150	Reserved																																
0x1154	ETH_ DMACCATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1158	Reserved																																
0x115C	ETH_ DMACCARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1160	ETH_DMACSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0

Table 549. ETH_DMA_CH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1164 - 0x1168	Reserved																																
0x116C	ETH_ DMACMFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

57.11.3 Ethernet MTL registers

Operating mode Register (ETH_MTLOMR)

Address offset: 0x0C00

Reset value: 0x0000 0000

The Operating Mode register establishes the Transmit and Receive operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CNTCLR	CNTPRST	Res.	Res.	Res.	Res.	Res.	Res.	DTXSTS	Res.
						rw	rw							rw	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CNTCLR**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNT_PRESET bit, CNT_PRESET has precedence.

Bit 8 **CNTPRST**: Counters Preset

When this bit is set:

- ETH_MTLTXQUR register is initialized/preset to 0x7F0.
- Missed Packet and Overflow Packet counters in ETH_MTLRXQMPOCR register is initialized/preset to 0x7F0

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 Reserved, must be kept at reset value.

Bit 1 **DTXSTS**: Drop Transmit Status

When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

This bit is reserved and read-only when the Disable Transmit Status in MTL feature is selected while configuring the core.

Bit 0 Reserved, must be kept at reset value.

Interrupt status Register (ETH_MTLISR)

Address offset: 0x0C20

Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q0IS
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **Q0IS**: Queue interrupt status

This bit indicates that an interrupt has been generated by Queue. To reset this bit, read ETH_MTLQICSR register to identify the interrupt cause and clear the source.

Tx queue operating mode Register (ETH_MTLTXQOMR)

Address offset: 0x0D00

Reset value: 0x0007 0008

The Queue Transmit Operating Mode register establishes the Transmit queue operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
									rw	rw	rw	r	r	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **TQS[8:0]**: Transmit Queue Size

This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The sixteenth bit is the starting bit of this field. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits:
LOG2(2048/256) = LOG2(8) = 3 bits
TQS = 0 corresponds to 256 bytes.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **TTC[2:0]**: Transmit Threshold Control

These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.
000: 32
001: 64
010: 96
011: 128
100: 192
101: 256
110: 384
111: 512

Bits 3:2 **TXQEN[1:0]**: Transmit Queue Enable

This field is used to enable/disable the transmit queue.
00: Not enabled
01: Reserved
10: Enabled
11: Reserved

Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.

Bit 1 **TSF**: Transmit Store and Forward

When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped.

Bit 0 **FTQ**: Flush Transmit Queue

When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the ETH_MTLTXQOMR register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (eth_mii_tx_clk) should be active.

Tx queue underflow register (ETH_MTLTXQUR)

Address offset: 0x0D04

Reset value: 0x0000 0000

The Queue Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **UFCNTOVF**: Overflow Bit for Underflow Packet Counter

This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened.

Bits 10:0 **UFFRMCNT[10:0]**: Underflow Packet Counter

This field indicates the number of packets aborted by the controller because of Tx queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read.

Tx queue debug Register (ETH_MTLTXQDR)

Address offset: 0x0D08

Reset value: 0x0000 0000

The Queue Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]		
									r	r	r		r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTS FSTS	TXQ STS	TWC STS	TRCSTS[1:0]		TXQPA USED
										r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **STXSTSF[2:0]**: Number of Status Words in Tx Status FIFO of Queue

This field indicates the current number of status in the Tx Status FIFO of this queue. This field is reserved when the Disable Transmit Status in MTL feature is selected while configuring the core.

When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PTXQ[2:0]**: Number of Packets in the Transmit Queue

This field indicates the current number of packets in the Tx queue. This field is reserved when the Disable Transmit Status in MTL feature is selected while configuring the core.

When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of packets in the Transmit queue.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TXSTSFSTS**: MTL Tx Status FIFO Full Status

When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.

This field is reserved when the Disable Transmit Status in MTL feature is selected while configuring the core.

Bit 4 **TXQSTS**: MTL Tx Queue Not Empty Status

When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission.

Bit 3 **TWCSTS**: MTL Tx Queue Write Controller Status

When high, this bit indicates that the MTL Tx queue Write Controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx Queue Read Controller Status

This field indicates the state of the Tx Queue Read Controller:

00: Idle state

01: Read state (transferring data to the MAC transmitter)

10: Waiting for pending Tx Status from the MAC transmitter

11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit Queue in Pause

When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the full-duplex only mode) because of the following:

- Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
- Reception of 802.3x Pause packet when PFC is disabled

Queue interrupt control status Register (ETH_MTLQICSR)

Address offset: 0x0D2C

Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	Res	TXUNFIS
						rw	rw							rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive Queue Overflow Interrupt Enable

When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **RXOVFIS**: Receive Queue Overflow Interrupt Status

This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TXUIE**: Transmit Queue Underflow Interrupt Enable

When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **TXUNFIS**: Transmit Queue Underflow Interrupt Status

This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.

Rx queue operating mode register (ETH_MTLRXQOMR)

Address offset: 0x0D30

Reset value: 0x0070 0000

The Queue Receive operating Mode register establishes the Receive queue operating modes and command.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[2:0]			Res.	Res.	Res.	RFD [2]
									r	r	r				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFD[1:0]		Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
rw	rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **RQS[2:0]**: Receive Queue Size

This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx queues more than one and the reset value is 0x0.

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:14 **RFD[2:0]**: Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes)

These bits control the threshold (fill-level of Rx queue) at which the flow control is de-asserted after activation:

0: Full minus 1 Kbyte, that is, FULL 1 Kbyte

1: Full minus 1.5 Kbyte, that is, FULL 1.5 Kbyte

2: Full minus 2 Kbytes, that is, FULL 2 Kbytes

3: Full minus 2.5 Kbytes, that is, FULL 2.5 Kbytes

4: Full minus 2.5 Kbytes, that is, FULL 3 Kbytes

5: Full minus 2.5 Kbytes, that is, FULL 3.5 Kbytes

The de-assertion is effective only after flow control is asserted.

Note: The value must be programmed in such a way to make sure that the threshold is a positive number.

When the EHFC is set high, these values are applicable only when the Rx queue size determined by the RQS field of this register, is equal to or greater than 4 Kbytes.

For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 Kbyte) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the RQS field of this register.

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:8 **RFA[2:0]**: Threshold for Activating Flow Control (in half-duplex and full-duplex

These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, see RFD.

Bit 7	**EHFC**: Enable Hardware Flow Control When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled. This bit is not used (reserved and always reset) when the Rx queue is less than 4 Kbytes.
Bit 6	**DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset.
Bit 5	**RSF**: Receive Queue Store and Forward When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.
Bit 4	**FEP**: Forward Error Packets When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.
Bit 3	**FUP**: Forward Undersized Good Packets When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.
Bit 2	Reserved, must be kept at reset value.
Bits 1:0	**RTC[1:0]**: Receive Queue Threshold Control These bits control the threshold level of the MTL Rx queue (in bytes): 00: 64 01: 32 10: 96 11: 128 The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

Rx queue missed packet and overflow counter register (ETH_MTLRXQMPOCR)

Address offset: 0x=0D34

Reset value: 0x0000 0000

The Queue missed packet and overflow counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]										
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTOVF**: Missed Packet Counter Overflow Bit
When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed Packet Counter
This field indicates the number of packets missed by the Ethernet peripheral because the application asserted ari_pkt_flush_i[] for this queue. This counter is reset when this register is read.
This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **OVFCNTOVF**: Overflow Counter Overflow Bit
When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow Packet Counter
This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.

Rx queue debug register (ETH_MTLRXQDR)

Address offset: 0x0D38

Reset value: 0x0000 0000

The Queue Receive Debug register gives the debug status of various blocks related to the Receive queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PRXQ[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
										r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PRXQ[13:0]**: Number of Packets in Receive Queue

This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256Kbyte/16bytes = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **RXQSTS[1:0]**: MTL Rx Queue Fill-Level Status

This field gives the status of the fill-level of the Rx queue:
00: Rx queue empty
01: Rx queue fill-level below flow-control deactivate threshold
10: Rx queue fill-level above flow-control activate threshold
11: Rx queue full

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **RRCSTS[1:0]**: MTL Rx Queue Read Controller State

This field gives the state of the Rx queue Read controller:
00: Idle state
01: Reading packet data
10: Reading packet status (or timestamp)
11: Flushing the packet data and status

Bit 0 **RWCSTS**: MTL Rx Queue Write Controller Active Status

When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.

Ethernet MTL register map and reset values

Table 550. ETH_MTL register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C00	ETH_MTLOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTCLR	CNTPRST	Res.	Res.			Res.	Res.	DTXSTS	Res.
	Reset value																							0	0							0	
0x0C04 - 0x0C1C	Reserved																																
0x0C20	ETH_MTLISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q0TXOIE
	Reset value																																0
0x0C24 - 0x0CFC	Reserved																																
0x0D00	ETH_ MTLTXQOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC [2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value													0	1	1	1										0	0	0	1	0	0	0
0x0D04	ETH_MTLTXQUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D08	ETH_MTLTXQDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D0C - 0x0D14	Reserved																																
0x0D18 - 0x0D28	Reserved																																
0x0D2C	ETH_MTLQICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	TXUNFIS
	Reset value								0								0							0	0							0	0

Table 550. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D30	ETH_ MTLRXQOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS2	RQS1	RQS0	Res.	Res.	Res.	RFD						RFA			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC	
	Reset value										1	1	1				0	0	0				0	0	0	0	0	0	0	0		0	0
0x0D34	ETH_ MTLRXQMPOCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]										Res.	Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0D38	ETH_ MTLRXQDR	Res.	Res.	PRXQ[13:0]													Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0		0	0	0
0xD3C to 0xD7C	Reserved																																
0x0D68	Reserved																																

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

57.11.4 Ethernet MAC and MMC registers

Operating mode configuration register (ETH_MACCR)

Address offset: 0x0000

Reset value: 0x0000 0000

The MAC Configuration Register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	Res.	JD	JE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **ARPEN**: ARP Offload Enable

When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It will forward the ARP packet to the application and also indicate the events in the RxStatus.

When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.

This bit is available only when the Enable IPv4 ARP Offload is selected.

Bits 30:28 **SARC[2:0]**: Source Address Insertion or Replacement Control

This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:

0x: Reserved.

10:

- If Bit 30 is set to 0, the MAC inserts the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets.
- If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected while configuring the core, the MAC inserts the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets.

11:

- If Bit 30 is set to 0, the MAC replaces the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets.
- If Bit 30 is set to 1 and the MAC Address Register 1 is enabled, the MAC replaces the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets.

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

These bits are reserved and RO when the Enable SA and VLAN Insertion on Tx feature is not selected while configuring the core.

Bit 27 **IPC**: Checksum Offload

When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.

If the IP Checksum Offload feature is not enabled while configuring the core, this bit is reserved and RO (with default value).

The Layer 3 and Layer 4 Packet Filter and Enable Split Header features automatically selects the IPC Full Checksum Offload Engine on the Receive side. When any of these features are enabled, you must set the IPC bit.

Bits 26:24 **IPG[2:0]**: Inter-Packet Gap

These bits control the minimum IPG between packets during transmission.
000: 96 bit times
001: 88 bit times
010: 80 bit times
...
111: 40 bit times
This range of minimum IPG is valid in full-duplex mode.
In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.
When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.
The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23 **GPSLCE**: Giant Packet Size Limit Control Enable

When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.
When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).
The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22 **S2KP**: IEEE 802.3as Support for 2K Packets

When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.
When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see Gaint Packet Status based on S2KP and JE Bits.

Note: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21 **CST**: CRC stripping for Type packets

When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.

Note: For information about how the settings of the ACS bit and this bit impact the packet length, see Packet Length based on the CST and ACS Bits.

Bit 20 **ACS**: Automatic Pad or CRC Stripping

When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.

When this bit is reset, the MAC passes all incoming packets to the application, without any modification.

Note: For information about how the settings of CST bit and this bit impact the packet length, see the Table, Packet Length based on the CST and ACS Bit.

Bit 19 **WD**: Watchdog Disable

When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.

When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **JD**: Jabber Disable

When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.

When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.

Bit 16 **JE**: Jumbo Packet Enable

When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **FES**: MAC Speed

This bit selects the speed in the 10/100 Mbps mode:

0: 10 Mbps

1: 100 Mbps

In the 1000 Mbps-only configurations, this bit is read-only with the reset value. In the 10 or 100 Mbps-only or default 10/100 Mbps configurations, this bit is read-write. The mac_speed_o[0] signal reflects the value of this bit.

Bit 13 **DM**: Duplex Mode

When this bit is set, the MAC operates in the full-duplex mode in which it can transmit and receive simultaneously. This bit is RO with default value of 1'b1 in the full-duplex-only configurations.

Bit 12 **LM**: Loopback Mode

When this bit is set, the MAC operates in the loopback mode at MII. The MII Rx clock input (eth_mii_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Bit 11 **ECRSFD**: Enable Carrier Sense Before Transmission in Full-Duplex Mode

When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low.

When this bit is reset, the MAC transmitter ignores the status of the CRS signal.

Bit 10 **DO**: Disable Receive Own

When this bit is set, the MAC disables the reception of packets when the ETH_TX_EN is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.

This bit is not applicable in the full-duplex mode. This bit is reserved and read-only (RO) with default value in the full-duplex-only configurations.

Bit 9 **DCRS**: Disable Carrier Sense During Transmission

When this bit is set, the MAC transmitter ignores the (G)MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.

When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

This bit is reserved and read-only (RO) in the full-duplex-only configurations.

Bit 8 **DR**: Disable Retry

When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.

When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the half-duplex mode. This bit is reserved and read-only (RO) in the full-duplex-only configurations.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **BL[1:0]**: Back-Off Limit

The back-off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps; 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision.

00: k= min (n, 10)

01: k = min (n, 8)

10: k = min (n, 4)

11: k = min (n, 1)

where - = retransmission attempt

The random integer r takes the value in the range 0 <= r < 2^k

This bit is applicable only in the half-duplex mode. This bit is reserved and read-only (RO) in the full-duplex-only configurations.

Bit 4 **DC**: Deferral Check

When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode.

If the MAC is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII.

The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.

When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.

This bit is applicable only in the half-duplex mode. This bit is reserved and read-only (RO) in the full-duplex-only configurations.

Bits 3:2 **PRELEN[1:0]**: Preamble Length for Transmit packets

These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode.

00: 7 bytes of preamble

01: 5 bytes of preamble

10: 3 bytes of preamble

11: Reserved

Bit 1 **TE**: Transmitter Enable

When this bit is set, the Tx state machine of the MAC is enabled for transmission on the MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 **RE**: Receiver Enable

When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the MII interface.

Table 551 shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

Table 551. Giant Packet Status based on S2KP and JE Bits

Length/Type Field	Received Packet Length	S2KP	JE	Giant Packet Status
Untagged packet	> 1,518	0	0	1
	> 2,000	1	0	1
	> 9,018	x	1	1
VLAN tagged packet	> 1,522	0	0	1
	> 2,000	1	0	1
	> 9,022	x	1	1
Note: For all other combinations, the Giant Packet status is 0.				

Table 552 shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

Table 552. Packet Length based on the CST and ACS bits

Receive Checksum Offload Engine	Received Packet Length	CST	ACS	FCS Stripping Done
IPCHKSUM_EN = 0 and IPC_FULL_OFFLOAD = 0 or IPCHKSUM_EN = 1 and IPC_FULL_OFFLOAD = 1	< 1,536	x	0	No
		x	1	Yes (for Ethernet packets)
	>= 1,536	0	x	No
		1	x	Yes (for Type packets)
IPCHKSUM_EN = 1 and IPC_FULL_OFFLOAD = 0	< 1,536	x	0	No
		x	1	Yes (for Ethernet packets)
	>= 1,536	x	x	No

Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004

Reset value: 0x0000 0000

The MAC Extended Configuration Register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC
		rw	rw	rw	rw	rw	rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GPSL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **EIPG[4:0]**: Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits), along with IPG field in ETH_MACCR, gives the minimum IPG greater than 96 bit times in steps of 8 bit times:

{EIPG, IPG}

0x00:104 bit times

0x01: 112 bit times

0x02: 120 bit times

..

0xFF: 2144 bit times

Bit 24 **EIPGEN**: Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in ETH_MACCR together as minimum IPG greater than 96 bit times in steps of 8 bit times.

When this bit is reset, the MAC ignores EIPG field and interprets IPG field in ETH_MACCR as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-Duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-Duplex mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **USP**: Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the ETH_MACA0HR and ETH_MACA0LR registers. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).

When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.

Bit 17 **SPEN**: Slow Protocol Detection Enable

When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid sub-types.

When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16 **DCRCC**: Disable CRC Checking for Received Packets

When this bit is set, the MAC receiver does not check the CRC field in the received packets.

When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **GPSL[13:0]**: Giant Packet Size Limit

If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.

For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. When the Enable Double VLAN Processing option is selected, the MAC adds 8 bytes to the programmed value for double VLAN tagged packets. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

Packet filtering control register (ETH_MACPFR)

Address offset: 0x0008

Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE
rw										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RA**: Receive All

When this bit is set, the MAC Receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word.

When this bit is reset, the Receiver module passes only those packets to the application that pass the SA or DA address filter.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DNTU**: Drop Non-TCP/UDP over IP Packets

When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.

If the Enable Layer 3 and Layer 4 Packet Filter option is not selected, this bit is reserved (RO with default value).

Bit 20 **IPFE**: Layer 3 and Layer 4 Filter Enable

When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.

When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

If the Enable Layer 3 and Layer 4 Packet Filter option is not selected, this bit is reserved (RO with default value).

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **VTFE**: VLAN Tag Filter Enable

When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HPF**: Hash or Perfect Filter

When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.

When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter. This bit is reserved (and RO) if the Enable Address Filter Hash Table option is not selected.

Bit 9 **SAF**: Source Address Filter Enable

When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.

When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

Note: According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8 **SAIF**: SA Inverse Filtering

When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.

When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6 **PCF[1:0]**: Pass Control Packets

These bits control the forwarding of all control packets (including unicast and multicast Pause packets).

00: The MAC filters all control packets from reaching the application.

01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.

10: The MAC forwards all control packets to the application even if they fail the Address filter.

11: The MAC forwards the control packets that pass the Address filter.

Bit 5 **DBF**: Disable Broadcast Packets

When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.

When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4 **PM**: Pass All Multicast

When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.

Bit 3 **DAIF**: DA Inverse Filtering

When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast

When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.

When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

If the Enable Address Filter Hash Table option is not selected, this bit is reserved (and RO).

Bit 1 **HUC**: Hash Unicast

When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.

When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

If the Enable Address Filter Hash Table option is not selected, this bit is reserved (and RO).

Bit 0 **PR**: Promiscuous Mode

When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

Watchdog timeout register (ETH_MACWTR)

Address offset: 0x000C

Reset value: 0x0000 0000

The Watchdog Timeout register controls the watchdog timeout for received packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
							rw					rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PWE**: Programmable Watchdog Enable

When this bit is set and the WD bit of the ETH_MACCR register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in ETH_MACCR register.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WTO[3:0]**: Watchdog Timeout

When the PWE bit is set and the WD bit of the ETH_MACCR register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes
0x1: 3 Kbytes
0x2: 4 Kbytes
0x3: 5 Kbytes
..
0xC: 14 Kbytes
0xD: 15 Kbytes
0xE: 16383 Bytes
0xF: Reserved

Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

Hash Table 0 register (ETH_MACHT0R)

Address offset: 0x0010

Reset value: 0x0000 0000

The Hash Table Register 0 contains the first 32 bits of the Hash table (64 bits).

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1).

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

If the Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT31T0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT31T0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT31T0[31:0]**: MAC Hash Table First 32 Bits

This field contains the first 32 Bits [31:0] of the Hash table.

Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014

Reset value: 0x0000 0000

The Hash Table Register 1contains the last 32 bits of the Hash table (64 bits).

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1).

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

If the Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT63T32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT63T32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT63T32[31:0]**: MAC Hash Table Second 32 Bits

This field contains the second 32 Bits [63:32] of the Hash table.

VLAN tag register (ETH_MACVTR)

Address offset: 0x0050

Reset value: 0x0000 0000

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS [1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV
rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EIVLRXS**: Enable Inner VLAN Tag in Rx Status

When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **EIVLS[1:0]**: Enable Inner VLAN Tag Stripping on Receive

This field indicates the stripping operation on inner VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 27 **ERIVLT**: Enable Inner VLAN Tag

When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.

Bit 26 **EDVLP**: Enable Double VLAN Processing

When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).

Bit 25 **VTHM**: VLAN Tag Hash Table Match Enable

When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHTR register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table.
When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison.
When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison.
When this bit is reset, the VLAN Hash Match operation is not performed. If the VLAN Hash feature is not enabled, this bit is reserved (RO with default value).

Bit 24 **EVLRXS**: Enable VLAN Tag in Rx status

When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EVLS[1:0]**: Enable VLAN Tag Stripping on Receive

This field indicates the stripping operation on the outer VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 20 **DOVLTC**: Disable VLAN Type Check

When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19 **ERSVLM**: Enable Receive S-VLAN Match

When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18 **ESVL**: Enable S-VLAN

When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17 **VTIM**: VLAN Tag Inverse Match Enable

When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16 **ETV**: Enable 12-Bit VLAN Tag Comparison

When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:0 **VL[15:0]**: VLAN Tag Identifier for Receive Packets

This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag
When the ETV bit is set, only the VID is used for comparison.
If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.

VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058

Reset value: 0x0000 0000

When the ERSVLM bit of ETH_MACHT1R register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of ETH_MACVTR register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

If the VLAN Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLHT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VLHT[15:0]**: VLAN Hash Table
This field contains the 16-bit VLAN Hash Table.

VLAN inclusion register (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC [1:0]	
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.

10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Inner VLAN inclusion register (ETH_MACIVIR)

Address offset: 0x0064

Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC [1:0]	
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion

The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.

10: VLAN tag insertion

The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement

The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Tx Queue flow control register (ETH_MACQTXFCR)

Address offset: 0x0070

Reset value: 0x0000 0000

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
								rw	rw	rw	rw			rw	rw

Bits 31:16 **PT[15:0]**: Pause Time

This field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the (G)MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DZPQ**: Disable Zero-Quanta Pause

When this bit is set, it disables the automatic generation of the zero-quanta Pause packets. When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Bits 6:4 **PLT[2:0]**: Pause Low Threshold

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

000: Pause Time minus 4 Slot Times (PT -4 slot times)

001: Pause Time minus 28 Slot Times (PT -28 slot times)

010: Pause Time minus 36 Slot Times (PT -36 slot times)

011: Pause Time minus 144 Slot Times (PT -144 slot times)

100: Pause Time minus 256 Slot Times (PT -256 slot times)

101: Pause Time minus 512 Slot Times (PT -512 slot times)

110-111: Reserved

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TFE**: Transmit Flow Control Enable

Full-Duplex Mode: when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

Half-Duplex Mode: when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0 **FCB_BPA**: Flow Control Busy or Backpressure Activate

This bit initiates a Pause packet in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set.

Full-Duplex Mode: this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

Half-Duplex Mode: When this bit is set (and TFE bit is set) in the half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled.

Rx flow control register (ETH_MACRXFCR)

Address offset: 0x0090

Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UP**: Unicast Pause Packet Detect

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in ETH_MACA0HR and ETH_MACA0LR.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011.

Bit 0 **RFE**: Receive Flow Control Enable

When this bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.

Interrupt status register (ETH_MACISR)

Address offset: 0x00B0

Reset value: 0x0000 0000

The Interrupt Status register contains the status of interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	Res.
	r	r	r		r	r	r			r	r	r			

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIS**: Receive Status Interrupt

This bit indicates the status of received packets. This bit is set when the RWT bit is set in the ETH_MACISR register. This bit is cleared when the corresponding interrupt source bit is read in the ETH_MACISR register.

Bit 13 **TXSTSIS**: Transmit Status Interrupt

This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the ETH_MACISR register:

- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber Timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read in the ETH_MACISR register.

Bit 12 **TSIS**: Timestamp Interrupt Status

If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:

- The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.
- There is an overflow in the Seconds register.
- The Target Time Error occurred, that is, programmed target time already elapsed.

If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the ETH_MACTXTSSNR and ETH_MACTXTSSSR registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the ETH_MACTXTSSNR and ETH_MACTXTSSSR registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the corresponding interrupt source bit is read in the ETH_MACTSSR register.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC Transmit Interrupt Status

This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.

This bit is valid only when you select the Enable MAC Management Counters (MMC) option.

Bit 9 **MMCRXIS**: MMC Receive Interrupt Status

This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.

This bit is valid only when you select the Enable MAC Management Counters (MMC) option.

Bit 8 **MMCIS**: MMC Interrupt Status

This bit is set high when Bit 11, Bit 10, or Bit 9 is set high. This bit is cleared only when all these bits are low. This bit is valid only when you select the Enable MAC Management Counters (MMC) option.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **LPIIS**: LPI Interrupt Status

When the Energy Efficient Ethernet feature is enabled, this bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared when the TLPIEN bit of ETH_MACLCSR register is read. In all other modes, this bit is reserved.

Bit 4 **PMTIS**: PMT Interrupt Status

This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the ETH_MACPCSR register.

This bit is valid only when you select the Enable Power Management option.

Bit 3 **PHYIS**: PHY Interrupt

This bit is set when rising edge is detected on the phy_intr_i input. This bit is cleared when this register is read.

Bits 2:0 Reserved, must be kept at reset value.

Interrupt enable register (ETH_MACIER)

Address offset: 0x00B4

Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	Res.
	rw	rw	rw							rw	rw	rw			

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIE**: Receive Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the ETH_MACISR register.

Bit 13 **TXSTSIE**: Transmit Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the ETH_MACISR register.

Bit 12 **TSIE**: Timestamp Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in ETH_MACISR register.

This bit is valid only when the Enable IEEE 1588 Timestamp Support option is selected. Otherwise, this bit is reserved.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **LPIIE**: LPI Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in ETH_MACISR register.

This bit is valid only when the Enable Energy Efficient Ethernet (EEE) option is selected. Otherwise, this bit is reserved.

Bit 4 **PMTIE**: PMT Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in ETH_MACISR register.

Bit 3 **PHYIE**: PHY Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in ETH_MACISR register.

Bits 2:0 Reserved, must be kept at reset value.

Rx Tx status register (ETH_MACRXTXSR)

Address offset: 0x00B8

Reset value: 0x0000 0000

The Receive Transmit Status register contains the Receive and Transmit Error status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
							r			r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RWT**: Receive Watchdog Timeout

This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the ETH_MACCR register. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the ETH_MACCR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **EXCOL**: Excessive Collisions

When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the ETH_MACCR register, this bit is set after the first collision and the packet transmission is aborted.

Bit 4 **LCOL**: Late Collision

When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode.

This bit is not valid if the Underflow error occurs.

Bit 3 **EXDEF**: Excessive Deferral

When the DTXSTS bit is set in the MTL_Operation_Mode register and the DC bit is set in the ETH_MACCR register, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000 Mbps mode or when Jumbo packet is enabled).

Bit 2 **LCARR**: Loss of Carrier

When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the loss of carrier occurred during packet transmission, that is, the phy_crs_i signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.

Bit 1 **NCARR**: No Carrier

When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.

Bit 0 **TJT**: Transmit Jabber Timeout

This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the ETH_MACCR register. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the ETH_MACCR register.

PMT control status register (ETH_MACPCSR)

Address offset: 0x00C0

Reset value: 0x0000 0000

The PMT Control and Status Register is present only when you select the PMT module in coreConsultant.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw			r	r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
					rw	rw			r	r			rw	rw	rw

Bit 31 **RWKFILTRST**: Remote wakeup Packet Filter Register Pointer Reset

When this bit is set, the remote wakeup packet filter register pointer is reset to 3'b000. It is automatically cleared after 1 clock cycle.

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:24 **RWKPTR[4:0]**: Remote wakeup FIFO Pointer

This field gives the current value (0 to 7) of the Remote wakeup Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wakeup Packet Filter Register are transferred to the eth_mii_rx_clk domain when a Write occurs to that register.

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **RWKPFE**: Remote wakeup Packet Forwarding Enable

When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wakeup frame. All frames after that event including the received wakeup frame are forwarded to application. This bit is then self-cleared on receiving the wakeup packet.

The application can also clear this bit before the expected wakeup frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high.

Note: If Magic Packet Enable and wakeup Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wakeup frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.

Bit 9 **GLBLUCAST**: Global Unicast

When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wakeup packet.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RWKPRCVD**: Remote wakeup Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a remote wakeup packet. This bit is cleared when this register is read.

Bit 5 **MGKPRCVD**: Magic Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RWKPKTEN**: Remote wakeup Packet Enable

When this bit is set, a power management event is generated when the MAC receives a remote wakeup packet.

Bit 1 **MGKPKTEN**: Magic Packet Enable

When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0 **PWRDWN**: Power Down

When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wakeup packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wakeup packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wakeup Packet Enable bit is set high.

Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

Remote wakeup packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACRWKPFR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACRWKPFR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MACRWKPFR[31:0]**: Remote wakeup packet filter
Refer to *Table 553* and *Table 554* for details on register content and programming sequence.

The ETH_MACRWKPFR register at address 0x00C4 loads the wakeup Packet Filter register.

To load values in a wakeup Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

Table 553. Remote wakeup packet filter register

ETH_MACRWKPFR value #	Field							
0	Filter 0 byte mask							
1	Filter 1 byte mask							
2	Filter 2 byte mask							
3	Filter 3 byte mask							
4	Reserved	Filter 3 command	Reserved	Filter 2 command	Reserved	Filter 1 command	Reserved	Filter 0 command
5	Filter 3 offset		Filter 2 offset		Filter 1 offset		Filter 0 offset	
6	Filter 1 CRC - 16				Filter 0 CRC - 16			
7	Filter 3 CRC - 16				Filter 2 CRC - 16			

Table 554. ETH_MACRWKPFR

Register	Description
Filter *i* Byte mask	The filter *i* byte mask register defines the bytes of the packet that are examined by filter *i* (0, 1, 2 or 3) to determine whether or not a packet is a wakeup packet. – The MSB (31st bit) must be zero. – Bit j[30:0] is the byte mask. – If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter *i* Offset + j of the incoming packet; otherwise Filter *i* Offset + j is ignored.
Filter *i* Command	The 4-bit filter *i* command controls the filter *i* operation. – Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. – Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value. Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wakeup packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2". – Bit 1 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. – Bit 0 is the enable for filter *i*. If Bit 0 is not set, filter *i* is disabled.
Filter *i* Offset	This filter *i* offset register defines the offset (within the packet) from which the filter *i* examines the packets. – This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. – The minimum allowed offset is 12, which refers to the 13th byte of the packet. – The offset value 0 refers to the first byte of the packet.
Filter *i* CRC-16	This filter *i* CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wakeup filter register block. – The 16-bit CRC calculation uses the following polynomial: G(x) = x^16 + x^15 + x^2 + 1 Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following: – 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation. – 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

The below table lists the remote wakeup scenarios in which PMT interrupt is generated.

Table 555. Remote Wakeup Packet and PMT Interrupt Generation(1)

Filter i Command			Frame Type and CRC Status		Interrupt Generation
CAST	INV	EN	Received Frame Cast Type	CRC Status	RWK INTR
0	0	1	Unicast	MATCH	Remote Wakeup packet is detected and PMT interrupt is generated
0	1	1	Unicast	MISMATCH	Remote Wakeup packet is detected and PMT interrupt is generated
1	0	1	Multicast	MATCH	Remote Wakeup packet is detected and PMT interrupt is generated
1	1	1	Multicast	MISMATCH	Remote Wakeup packet is detected and PMT interrupt is generated

1. In all other combinations, the Remote Wakeup packet is not detected and PMT interrupt is not generated.

LPI control status register (ETH_MACLCSR)

Address offset: 0x00D0

Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITE	LPITXA	PLSEN	PLS	LPIEN
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
						r	r					r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **LPITE**: LPI Timer Enable

This bit controls the automatic entry of the MAC Transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPITXEN bits are set, the MAC Transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.

After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the Transmitter exits LPI state but does not clear LPITXEN bit. This enables the re-entry into LPI state when it is IDLE again.

When LPITE is 0, the LPI Auto timer is disabled and MAC Transmitter enters LPI state based on the settings of LPITXA and LPITXEN bit descriptions.

Bit 19 **LPITXA**: LPI Tx Automate

This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side.

If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTxQOMR, when the MAC is in the LPI mode, it exits the LPI mode.

When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18 **PLSEN**: PHY Link Status Enable

This bit enables the link status received on the Receive paths to be used for activating the LPI LS TIMER.

When this bit is set, the MAC uses the PLS bit for the LPI LS Timer trigger. When this bit is reset, the MAC ignores the link-status bits of the ETH_MACPHYCSR register and takes only the PLS bit.

This bit is RO and reserved if you have not selected the PHY interface.

Bit 17 **PLS**: PHY Link Status

This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.

When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16 **LPIEN**: LPI Enable

When this bit is set, it instructs the MAC Transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.

This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **RLPIST**: Receive LPI State

When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.

Bit 8 **TLPIST**: Transmit LPI State

When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **RLPIEX**: Receive LPI Exit

When this bit is set, it indicates that the MAC Receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register.

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 2 **RLPIEN**: Receive LPI Entry

When this bit is set, it indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register.

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 1 **TLPIEX**: Transmit LPI Exit

When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register.

Bit 0 **TLPIEN**: Transmit LPI Entry

When this bit is set, it indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register.

LPI timers control register (ETH_MACLTCR)

Address offset: 0x00D4

Reset value: 0x03E8 0000

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TWT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **LST[9:0]**: LPI LS Timer

This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.

Bits 15:0 **TWT[15:0]**: LPI TW Timer

This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

LPI entry timer register (ETH_MACLETR)

Address offset: 0x00D8

Reset value: 0x0000 0000

The LPI Entry Timer Register is used to store the LPI Idle Timer Value in Micro-Seconds.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[16:13]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPIET[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:3 **LPIET[16:0]**: LPI Entry Timer

This field specifies the time in microseconds the MAC will wait to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1.

Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

Bits 2:0 Reserved, must be kept at reset value.

1-microsecond-tick counter register (ETH_MAC1USTCR)

Address offset: 0x00DC

Reset value: 0x0000 0000

This register controls the generation of the Reference time (1-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **TIC_1US_CNTR[11:0]**: 1 µs tick Counter

The application must program this counter so that the number of clock cycles of CSR clock is 1 µs.

(Subtract 1 from the value before programming).

For example if the CSR clock is 100 MHz then this field needs to be programmed to 100 - 1 = 99 (which is 0x63).

This is required to generate the 1 µs events that are used to update some of the EEE related counters.

Version register (ETH_MACVR)

Address offset: 0x0110

Reset value: 0x0000 3041

The version register identifies the version of the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USERVER[7:0]								SNPSVER[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **USERVER[7:0]**: ST-defined version

Bits 7:0 **SNPSVER[7:0]**: IP version

Debug register (ETH_MACDR)

Address offset: 0x0114

Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
													r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **TFCSTS[1:0]**: MAC Transmit Packet Controller Status

This field indicates the state of the MAC Transmit Packet Controller module:
00: Idle state
01: Waiting for one of the following:
– Status of the previous packet OR
– IPG or backoff period to be over
10: Generating and transmitting a Pause control packet (in full-duplex mode)
11: Transferring input packet for transmission

Bit 16 **TPESTS**: MAC MII Transmit Protocol Engine Status

When this bit is set, it indicates that the MAC MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **RFCFCSTS[1:0]**: MAC Receive Packet Controller FIFO Status

When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.

Bit 0 **RPESTS**: MAC MII Receive Protocol Engine Status

When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.

HW feature 1 register (ETH_MACHWF1R)

Address offset: 0x0120

Reset value: 0x1184 1904

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		Res.	Res.	Res.	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN
	r	r	r	r		r	r				r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					Res.	RXFIFOSIZE[4:0]				
r	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
This field indicates the total number of L3 or L4 filters:
0000: No L3 or L4 Filter
0001: 1 L3 or L4 Filter
0010: 2 L3 or L4 Filters
..
1000: 8 L3 or L4

Bit 26 Reserved, must be kept at reset value.

Bits 25:24 **HASHTBLSZ[1:0]**: Hash Table Size
This field indicates the size of the Hash table:
00: No Hash table
01: 64
10: 128
11: 256

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **AVSEL**: AV Feature Enable
This bit is set to 1 when the Enable Audio Video Bridging option is selected.

Bit 19 **DBGMEMA**: DMA Debug Registers Enable
This bit is set to 1 when the Debug Mode Enable option is selected

Bit 18 **TSOEN**: TCP Segmentation Offload Enable
This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Bit 17 **SPHEN**: Split Header Feature Enable
This bit is set to 1 when the Enable Split Header Structure option is selected

Bit 16 **DCBEN**: DCB Feature Enable
This bit is set to 1 when the Enable Data Center Bridging option is selected

Bits 15:14 **ADDR64[1:0]**: Address width
This field indicates the configured address width.
00: 32 bits
Others: Reserved

Bit 13 **ADVTHWORD**: IEEE 1588 High Word Register Enable
This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected

Bit 12 **PTOEN**: PTP Offload Enable
This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11 **OSTEN**: One-Step Timestamping Enable
This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.

Bits 10:6 **TXFIFOSIZE[4:0]**: MTL Transmit FIFO Size
This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011-11111: Reserved

Bit 5 Reserved, must be kept at reset value.

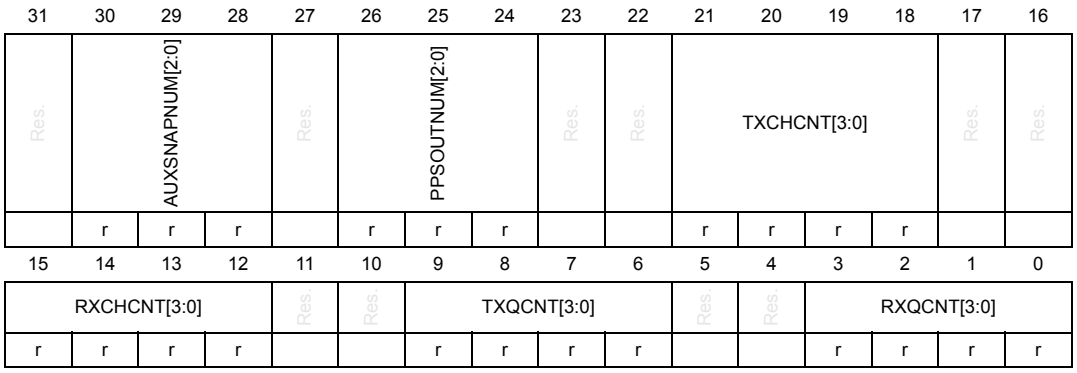
Bits 4:0 **RXFIFOSIZE[4:0]**: MTL Receive FIFO Size
This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32,767 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011: 256 Kbytes
01100-11111: Reserved

HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124

Reset value: 0x4100 0000

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.



Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **AUXSNAPNUM[2:0]**: Number of Auxiliary Snapshot Inputs
This field indicates the number of auxiliary snapshot inputs:
000: No auxiliary input
001: 1 auxiliary input
010: 2 auxiliary inputs
011: 3 auxiliary inputs
100: 4 auxiliary inputs
101-111: Reserved

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **PPSOUTNUM[2:0]**: Number of PPS Outputs
This field indicates the number of PPS outputs:
000: No PPS output
001: 1 PPS output
010: 2 PPS outputs
011: 3 PPS outputs
100: 4 PPS outputs
101-111: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **TXCHCNT[3:0]**: Number of DMA Transmit Channels
This field indicates the number of DMA Transmit channels:
0000: 1 DMA Tx Channel
0001: 2 DMA Tx Channels
..
0111: 8 DMA Tx

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:12 **RXCHCNT[3:0]**: Number of DMA Receive Channels
This field indicates the number of DMA Receive channels:
0000: 1 DMA Rx Channel
0001: 2 DMA Rx Channels
..
0111: 8 DMA Rx

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **TXQCNT[3:0]**: Number of MTL Transmit Queues
This field indicates the number of MTL Transmit queues:
0000: 1 MTL Tx queue
0001: 2 MTL Tx queues
..
0111: 8 MTL Tx

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **RXQCNT[3:0]**: Number of MTL Receive Queues
This field indicates the number of MTL Receive queues:
0000: 1 MTL Rx queue
0001: 2 MTL Rx queues
..
0111: 8 MTL Rx

MDIO address register (ETH_MACMDIOAR)

Address offset: 0x0200

Reset value: 0x0000 0000

The MDIO Address register controls the management cycles to external PHY through a management interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]				
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	MB
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PSE**: Preamble Suppression Enable

When this bit is set, the SMA will suppress the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.

When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.

Bit 26 **BTB**: Back to Back transactions

When this bit is set and the NTC has value greater than 0, then the MAC will inform the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which will be executed immediately irrespective of the number trailing clocks generated for the previous frame.

When this bit is reset, then the read/write command completion (MII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.

This bit must not be set when NTC=0.

Bits 25:21 **PA[4:0]**: Physical Layer Address

This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.
This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Bits 20:16 **RDA[4:0]**: Register/Device Address

These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **NTC[2:0]**: Number of Training Clocks

This field controls the number of trailing clock cycles generated on ETH_MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3'h3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.

Bits 11:8 **CR[3:0]**: CSR Clock Range

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design:
0000: CSR clock = 60-100 MHz; MDC clock = CSR clock/42
0001: CSR clock = 100-150 MHz; MDC clock = CSR clock/62
0010: CSR clock = 20-35 MHz; MDC clock = CSR clock/16
0011: CSR clock = 35-60 MHz; MDC clock = CSR clock/26
0100: CSR clock = 150-250 MHz; MDC clock = CSR clock/102
0101: CSR clock = 250-300 MHz; MDC clock = CSR clock/124
0110, 0111: Reserved
The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits as 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks:
1000: CSR clock/4
1001: CSR clock/6
1010: CSR clock/8
1011: CSR clock/10
1100: CSR clock/12
1101: CSR clock/14
1110: CSR clock/16
1111: CSR clock/18

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SKAP**: Skip Address Packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Bits 3:2 **GOC[1:0]**: MII Operation Command
This bit indicates the operation command to the PHY.
00: Reserved
01: Write
10: Post Read Increment Address for Clause 45 PHY
11: Read
When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E**: Clause 45 PHY Enable
When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.

Bit 0 **MB**: MII Busy
The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in ETH_MACMDIOAR and ETH_MACMDIODR registers as long as this bit is set.
For write transfers, the application must first write 16-bit data in the MD field (and also RA field when C45E is set) in ETH_MACMDIODR register before setting this bit. When C45E is set, it should also write into the RA field of ETH_MACMDIODR register before initiating a read transfer. When a read transfer is completed (MII busy=0), the data read from the PHY register is valid in the MD field of the ETH_MACMDIODR register.
Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit.

MDIO data register (ETH_MACMDIODR)

Address offset: 0x0204

Reset value: 0x0000 0000

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in ETH_MACMDIOAR. This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **RA[15:0]**: Register Address
This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

Bits 15:0 **MD[15:0]**: MII Data
This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

ARP address register (ETH_MACARPAR)

Address offset: 0x0AE0

Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPPA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARPPA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARPPA[31:0]**: ARP Protocol Address

This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.
This field is available only when the Enable IPv4 ARP Offload option is selected.

Address 0 high register (ETH_MACA0HR)

Address offset: 0x0300

Reset value: 0x8000 FFFF

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211.

If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

This bit is always set to 1.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address0[47:32]

This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

Address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)

Reset value: 0xFFFF FFFF

The MAC Address x Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)

This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

Address x high register (ETH_MACAxHR)

Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)

Reset value: 0x0000 FFFF

The MAC Address x High register holds the upper 16 bits of the second 6-byte MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30 **SA**: Source Address

When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.

Bits 29:24 **MBC[5:0]**: Mask Byte Control

These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:

Bit 29: Register 194[15:8]

Bit 28: Register 194[7:0]

Bit 27: Register 195[31:24]

..

Bit 24: Register 195[7:0]

You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address1 [47:32]

This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MMC control register (MMC_CONTROL)

Address offset: 0x0700

Reset value: 0x0000 0000

This register configures the MMC operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
							rw			rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets

The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of ETH_MACPFR register.

When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset

When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF_FFF0 (Half 16).

When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (Full 2Kbytes) and all packet-counters get preset to 0xFFFF_FFF0 (Full 16).

For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.

Bit 4 **CNTPRST**: Counters Preset

When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.

This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze

When this bit is set, it freezes all MMC counters to their current value.

Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Bit 2 **RSTONRD**: Reset on Read

When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover

When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

MMC Rx interrupt register (MMC_RX_INTERRUPT)

Address offset: 0x0704

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Receive statistics counters.

The MMC Receive Interrupt register maintains the interrupts that are generated when the following occur:

- Receive statistic counters reach half of their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter).
- Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32 bit counter and 0xFFFF for 16 bit counter).

When the Counter Stop Rollover is set, interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.
				r	r									r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
									r	r					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIS**: MMC Receive LPI transition counter interrupt status

This bit is set when the Rx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIS**: MMC Receive LPI microsecond counter interrupt status

This bit is set when the Rx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIS**: MMC Receive Unicast Good Packet Counter Interrupt Status
This bit is set when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIS**: MMC Receive Alignment Error Packet Counter Interrupt Status
This bit is set when the rxalignmenterror counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIS**: MMC Receive CRC Error Packet Counter Interrupt Status
This bit is set when the rxcrcerror counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt register (MMC_TX_INTERRUPT)

Address offset: 0x0708

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Transmit statistics counters.

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The MMC Transmit Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.
				r	r					r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIS**: MMC Transmit LPI transition counter interrupt status
This bit is set when the Tx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIS**: MMC Transmit LPI microsecond counter interrupt status
This bit is set when the Tx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIS**: MMC Transmit Good Packet Counter Interrupt Status
This bit is set when the txpacketcount_g counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIS**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Status
This bit is set when the txmulticol_g counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIS**: MMC Transmit Single Collision Good Packet Counter Interrupt Status
This bit is set when the txsinglecol_g counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

MMC Rx interrupt mask register (MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C

Reset value: 0x0000 0000

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.
				r	rw									rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
									rw	rw					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIM**: MMC Receive LPI transition counter interrupt Mask
Setting this bit masks the interrupt when the Rx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIM**: MMC Receive LPI microsecond counter interrupt Mask
Setting this bit masks the interrupt when the Rx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIM**: MMC Receive Unicast Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIM**: MMC Receive Alignment Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the rxalignmenterror counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIM**: MMC Receive CRC Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the rxcrcerror counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt mask register (MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710

Reset value: 0x0000 0000

This register maintains the masks for interrupts generated from all Transmit statistics counters.

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32 bit wide. This register is present only when any one of the MMC Transmit Counters is selected during core configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.
				r	rw					rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIM**: MMC Transmit LPI transition counter interrupt Mask
Setting this bit masks the interrupt when the Tx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIM**: MMC Transmit LPI microsecond counter interrupt Mask
Setting this bit masks the interrupt when the Tx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value.

Bits 25:21 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIM**: MMC Transmit Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the txpacketcount_g counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIM**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the txmulticol_g counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIM**: MMC Transmit Single Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the txsinglecol_g counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

Tx single collision good packets register (TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXSNGLCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXSNGLCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXSNGLCOLG[31:0]**: Tx Single Collision Good Packets

This field indicates the number of successfully transmitted packets after a single collision in the half-duplex mode.

Tx multiple collision good packets register (TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMULTCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMULTCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMULTCOLG[31:0]**: Tx Multiple Collision Good Packets

This field indicates the number of successfully transmitted packets after multiple collisions in the half-duplex mode.

Tx packet count good register (TX_PACKET_COUNT_GOOD)

Address offset: 0x0768

Reset value: 0x0000 0000

This register provides the number of good packets transmitted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPKTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPKTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPKTG[31:0]**: Tx Packet Count Good

This field indicates the number of good packets transmitted.

Rx CRC error packets register (RX_CRC_ERROR_PACKETS)

Address offset: 0x0794

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with CRC error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRCERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRCERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRCERR[31:0]**: Rx CRC Error Packets

This field indicates the number of packets received with CRC error.

Rx alignment error packets register (RX_ALIGNMENT_ERROR_PACKETS)

Address offset: 0x0798

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXALGNERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXALGNERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXALGNERR[31:0]**: Rx Alignment Error Packets

This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Rx unicast packets good register (RX_UNICAST_PACKETS_GOOD)

Address offset: 0x07C4

Reset value: 0x0000 0000

This register provides the number of good unicast packets received by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXUCASTG[31:0]**: Rx Unicast Packets Good

This field indicates the number of good unicast packets received.

Tx LPI microsecond timer register (TX_LPI_USEC_CNTR)

Address offset: 0x07EC

Reset value: 0x0000 0000

This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPIUSC[31:0]**: Tx LPI Microseconds Counter

This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Tx LPI transition counter register (TX_LPI_TRAN_CNTR)

Address offset: 0x07F0

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPITRC[31:0]**: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate Mode (because of LPITXA bit set in the LPI Control and Status register), the counter will increment.

Rx LPI microsecond counter register (RX_LPI_USEC_CNTR)

Address offset: 0x07F4

Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPIUSC[31:0]**: Rx LPI Microseconds Counter

This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (RX_LPI_TRAN_CNTR)

Address offset: 0x07F8

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPITRC[31:0]**: Rx LPI Transition counter

This field indicates the number of times Rx LPI Entry has occurred.

L3 and L4 control 0 register (ETH_MACL3L4C0R)

Address offset: 0x0900

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4. This register is reserved if the Layer 3 and Layer 4 Filtering feature is not selected during core configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0
										rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM0**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM0 bit is set high.

Bit 20 **L4DPM0**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM0**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM0 bit is set high.

Bit 18 **L4SPM0**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN0**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA Higher Bits Match

IPv4 Packets:

This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

IPv6 Packets:

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.

1: LSb[0] is masked.

2: Two LSbs [1:0] are masked

..

127: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA Higher Bits Match

IPv4 Packets:

This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

IPv6 Packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.

Bit 2 **L3SAM0**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN0**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

Layer4 address filter 0 register (ETH_MACL4A0R)

Address offset: 0x0904

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP0[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in ETH_MACL3L4C0R register, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP0[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in ETH_MACL3L4C0R register, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer 3 Address 0 filter 0 register (ETH_MACL3A00R)

Address offset: 0x0910

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 Register 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A00[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A00[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A00[31:0]**: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 Register 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A10[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A10[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A10[31:0]**: Layer 3 Address 1 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 Address 2 filter 0 register (ETH_MACL3A20)

Address offset: 0x0918

Reset value: 0x0000 0000

The Layer 3 Address 2 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A20[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A20[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A20[31:0]**: Layer 3 Address 2 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the ETH_MACL3L4C0R register, this field is not used.

Layer3 Address 3 filter 0 register (ETH_MACL3A30)

Address offset: 0x091C

Reset value: 0x0000 0000

The Layer 3 Address 3 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A30[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A30[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A30[31:0]**: Layer 3 Address 3 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the ETH_MACL3L4C0R register, this field is not used.

L3 and L4 control 1 register (ETH_MACL3L4C1R)

Address offset: 0x0930

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1
										rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM1**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM0 bit is set high.

Bit 20 **L4DPM1**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM1**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM0 bit is set high.

Bit 18 **L4SPM1**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN1**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Bits 15:11 **L3HDBM1[4:0]**: Layer 3 IP DA Higher Bits Match

IPv4 Packets:

This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

IPv6 Packets:

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.

1: LSb[0] is masked.

2: Two LSbs [1:0] are masked

..

127: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM1[4:0]**: Layer 3 IP SA Higher Bits Match

IPv4 Packets:

This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

IPv6 Packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM1**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.

Bit 2 **L3SAM1**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN1**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934

Reset value: 0x0000 0000

The Layer 4 Address 0 register and registers 580 through 667 are reserved (RO with default value) if Enable Layer 3 and Layer 4 Packet Filter option is not selected while configuring the core.

You can configure the Layer 3 and Layer 4 Address Registers to be double-synchronized by selecting the Synchronize Layer 3 and Layer 4 Address Registers to Rx Clock Domain option while configuring the core. When you select this option, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform consecutive writes to same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP1[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in ETH_MACL3L4C0R register, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP1[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in ETH_MACL3L4C0R register, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 Register 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A01[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A01[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A01[31:0]**: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 Register 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A11[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A11[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the ETH_MACL3L4C0R register, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 1 Register (ETH_MACL3A21R)

Address offset: 0x0948

Reset value: 0x0000 0000

The Layer 3 Address 2 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A21[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A21[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A21[31:0]**: Layer 3 Address 2 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the ETH_MACL3L4C0R register, this field is not used.

Layer3 address 3 filter 1 register (ETH_MACL3A31R)

Address offset: 0x94C

Reset value: 0x0000 0000

The Layer 3 Address 3 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A31[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A31[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A31[31:0]**: Layer 3 Address 3 Field

When the L3PEN0 and L3SAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the ETH_MACL3L4C0R register, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the ETH_MACL3L4C0R register, this field is not used.

Timestamp control Register (ETH_MACTSCR)

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the Receiver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	CSC	TSENMACADDR	SNAPTYPSEL[1:0]	
							rw					r	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	Res.	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw			rw		rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TXTSSTSM**: Transmit Timestamp Status Mode

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSTSMIS bit of the ETH_MACTXTSSNR register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSTSHI bit of the ETH_MACTXTSSSR register.

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CSC**: Enable checksum correction during OST for PTP over UDP/IPv4 packets

When this bit is set, the last two bytes of PTP message sent over UDP/IPv4 is updated to keep the UDP checksum correct, for changes made to origin timestamp and/or correction field as part of one step timestamp operation. The application shall form the packet with these two dummy bytes.

When reset, no updates are done to keep the UDP checksum correct. The application shall form the packet with UDP checksum set to 0.

Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering

When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.

When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.

For normal time stamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.

For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTYPSEL[1:0]**: Select PTP packets for Taking Snapshots

These bits, along with Bits 15 and 14, decide the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Timestamp Snapshot Dependency on Register Bits Table.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master

When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages

When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see Timestamp Snapshot Dependency on Register Bits Table.

Bit 13 **TSIPV4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPV6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets

When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format

When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control

When this bit is set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets

When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **TSADDREG**: Update Addend Register

When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TSUPDT**: Update Timestamp

When this bit is set, the system time is updated (added or subtracted) with the value specified in ETH_MACSTSUR and ETH_MACSTNUR.

This bit should be zero before updating it. This bit is reset when the update is complete in hardware. The Timestamp Higher Word register (if enabled during core configuration) is not updated.

Bit 2 **TSINIT**: Initialize Timestamp

When this bit is set, the system time is initialized (overwritten) with the value specified in the MAC Register 80 (System Time Seconds Update Register) and MAC Register 81 (System Time Nanoseconds Update Register).

This bit should be zero before it is updated. This bit is reset when the initialization is complete. The Timestamp Higher Word register (if enabled during core configuration) can only be initialized.

Bit 1 **TSCFUPDT**: Fine or Coarse Timestamp Update

When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0 **TSENA**: Enable Timestamp

When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.

On the Receive side, the MAC processes the 1588 packets only if this bit is set.

Table 556 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPSEL field in ETH_MACTSCR register.

Table 556. Timestamp Snapshot Dependency on Register Bits

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP Messages
00	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	X	X	SYNC, Delay_Req
11	X	X	Pdelay_Req, Pdelay_Resp

Sub-second increment register (ETH_MACSSIR)

Address offset: 0x0B04

Reset value: 0x0000 0000

The Sub-second Increment register is present only when the IEEE 1588 timestamp feature is selected without an external timestamp input. In Coarse Update mode [Bit 1 in ETH_MACTSCR register, the value in this register is added to the system time every clock cycle of HCLK. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SNSINC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SSINC[7:0]**: Sub-second Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [Bit 9 (TSCTRLSSR) is set in ETH_MACTSCR]. When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:8 **SNSINC[7:0]**:Sub-nanosecond Increment Value

This field contains the sub-nanosecond increment value, represented in nanoseconds multiplied by 2^8.

This value is accumulated with the sub-nanoseconds field of the sub-second register.

For example, when TSCTRLSSR field in the ETH_MACTSCR register is set. and if the required increment is 5.3ns, then SSINC should be 0x05 and SNSINC should be 0x4C.

Bits 7:0 Reserved, must be kept at reset value.

System time seconds register (ETH_MACSTSR)

Address offset: 0x0B08

Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from HCLK to CSR clock).

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TSS[31:0]**: Timestamp Second

The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

System time nanoseconds register (ETH_MACSTNR)

Address offset: 0x0B0C

Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSSS[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **TSSS[30:0]**: Timestamp Sub-seconds

The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When Bit 9 is set in ETH_MACTSCR, each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls-over to zero.

System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10

Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in ETH_MACTSCR register.

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSS[31:0]**: Timestamp Seconds

The value in this field is the sub-second part of the update. When ADDSUB is reset, this field must be programmed with the sub-second part of the update value, with an accuracy based on the TSCTRLSSR bit of the ETH_MACTSCR register. When ADDSUB is set, then this field must be programmed with the complement of the sub-second part of the update value as described below.

When TSCTRLSSR is set, then the programmed value must be 10^9 - <sub-second value>. When TSCTRLSSR is reset, then the programmed value must be 2^31 - <sub-second_value>

For example, when TSCTRLSSR bit is set and if 2.000000001 seconds need to be subtracted from the system time, then the TSS field in the MAC_Timestamp Seconds update register must be 0xFFFF_FFFE (that is, 2^32 - 2), ADDSUB bit in this register should be set, and the TSSS field must be 0x3B9A_C9FF (that is, 10^9 - 1).

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14

Reset value: 0x0000 0000

This register is present only when the IEEE 1588 timestamp feature is selected without external timestamp input.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDSUB	TSSS[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADDSUB**: Add or Subtract Time

When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0 **TSSS[30:0]**: Timestamp Sub-seconds

The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When the TSCTRLSSR bit is set in the ETH_MACTSCR register, each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF.

Timestamp addend register (ETH_MACTSAR)

Address offset: 0x0B18

Reset value: 0x0000 0000

The Timestamp Addend register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input. This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the ETH_MACTSCR register). The content of this register is added to a 32-bit accumulator in every clock cycle (of HCLK) and the system time is updated whenever the accumulator overflows.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSAR[31:0]**: Timestamp Addend Register

This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Timestamp status register (ETH_MACTSSR)

Address offset: 0x0B20

Reset value: 0x0000 0000

The Timestamp Status register is present only when the IEEE 1588 Timestamp feature is selected. All bits except Bits[27:25] gets cleared when the application reads this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]			
		r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
r												r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **ATSNS[4:0]**: Number of Auxiliary Timestamp Snapshots

This field indicates the number of Snapshots available in the FIFO. A value equal to the selected depth of FIFO (4, 8, or 16) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set. This bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected.

Bit 24 **ATSSTM**: Auxiliary Timestamp Snapshot Trigger Missed

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO. This bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATSSTN[3:0]**: Auxiliary Timestamp Snapshot Trigger Identifier

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

Bit 16: Auxiliary trigger 0

Bit 17: Auxiliary trigger 1

Bit 18: Auxiliary trigger 2

Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

Bit 15 **TXTSSIS**: Tx Timestamp Status Interrupt Status

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the ETH_MACTXTSSNR and ETH_MACTXTSSSR registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the ETH_MACTXTSSNR and ETH_MACTXTSSSR registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the ETH_MACTXTSSSR register is read.

This bit is reserved in all other configurations.

Bits 14:4 Reserved, must be kept at reset value.

Bit 3 **TSTRGTERR0**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPS_Target_Time_seconds and ETH_MACPPS_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit.

Bit 2 **AUXTSTRIG**: Auxiliary Timestamp Trigger Snapshot

This bit is set high when the auxiliary snapshot is written to the FIFO. This bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected.

Bit 1 **TSTARGT0**: Timestamp Target Time Reached

When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPS_Target_Time_seconds and ETH_MACPPS_Target_Time_Nanoseconds registers.

Bit 0 **TSSOVF**: Timestamp Seconds Overflow

When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF.

Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)

Address offset: 0x0B30

Reset value: 0x0000 0000

This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSMIS	TXTSSLO[30:16]														
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSLO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **TXTSSMIS**: Transmit Timestamp Status Missed

When this bit is set, it indicates one of the following:

– The timestamp of the current packet is ignored if TXTSSTSM bit of the ETH_MACTSCR register is reset

– The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the ETH_MACTSCR register is set.

Bits 30:0 **TXTSSLO[30:0]**: Transmit Timestamp Status Low

This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.

Tx timestamp status seconds register (ETH_MACTXTSSSR)

Address offset: 0x0B34

Reset value: 0x0000 0000

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXTSSHI[31:0]**: Transmit Timestamp Status High

This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

Auxiliary control register (ETH_MACACR)

Address offset: 0x0B40

Reset value: 0x0000 0000

The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATSEN3**: Auxiliary Snapshot 3 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[3] input is enabled. When this bit is reset, the events on this input are ignored.

This bit is reserved when one of the following is true:

– The Add IEEE 1588 Auxiliary Snapshot option is not selected while configuring the core.

– The selected number in the Number of IEEE 1588 Auxiliary Snapshot Inputs option is less than four.

Bit 6 **ATSEN2**: Auxiliary Snapshot 2 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[2] input is enabled. When this bit is reset, the events on this input are ignored.

This bit is reserved when one of the following is true:

– The Add IEEE 1588 Auxiliary Snapshot option is not selected while configuring the core.

– The selected number in the Number of IEEE 1588 Auxiliary Snapshot Inputs option is less than 3.

Bit 5 **ATSEN1**: Auxiliary Snapshot 1 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[1] input is enabled. When this bit is reset, the events on this input are ignored.

This bit is reserved when one of the following is true:

– The Add IEEE 1588 Auxiliary Snapshot option is not selected while configuring the core.

– The selected number in the Number of IEEE 1588 Auxiliary Snapshot Inputs option is less than 2.

Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[0] input is enabled. When this bit is reset, the events on this input are ignored.

This bit is reserved when the Add IEEE 1588 Auxiliary Snapshot option is not selected while configuring the core.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear

When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

This bit is reserved when the Add IEEE 1588 Auxiliary Snapshot option is not selected while configuring the core.

Auxiliary timestamp nanoseconds register (ETH_MACATSNR)

Address offset: 0x0B48

Reset value: 0x0000 0000

The Auxiliary Timestamp Nanoseconds register, along with ETH_MACATSSR, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in ETH_MACTSSR indicate the fill-level of the FIFO. The top of the FIFO is removed only when the last byte of MAC Register 91 (Auxiliary Timestamp - Seconds Register) is read. In the little-endian mode, this means when Bits[31:24] are read and in big-endian mode, Bits[7:0] are read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	AUXTSLO[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSLO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp

Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

Auxiliary timestamp seconds register (ETH_MACATSSR)

Address offset: 0x0B4C

Reset value: 0x0000 0000

The Auxiliary Timestamp - Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUXTSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUXTSHI[31:0]**: Auxiliary Timestamp

Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

Timestamp Ingress asymmetric correction register (ETH_MACTSIACR)

Address offset: 0x0B50

Reset value: 0x0000 0000

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTIAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTIAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction

This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^16. For example, 2.5 ns is represented as 0x00028000.

The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Timestamp Egress asymmetric correction register (ETH_MACTSEACR)

Address offset: 0x0B54

Reset value: 0x0000 0000

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTEAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTEAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction

This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^16.

For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD_8000, which is the 2's complement of 0x0002_8000(2.5 * 216). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003_4CCC (3.3 * 216).

Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)

Address offset: 0x0B58

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSIC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSIC[31:0]**: Timestamp Ingress Correction

This field contains the ingress path correction value as defined by the Ingress Correction expression.

Timestamp Egress correction nanosecond register (ETH_MACTSECNR)

Address offset: 0x0B5C

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSEC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSEC[31:0]**: Timestamp Egress Correction

This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

The PPS Control register is present only when the Timestamp feature is selected and External Timestamp is not enabled.

Bits[30:24] of this register are valid only when four Flexible PPS outputs are selected. Bits[22:16] are valid only when three or more Flexible PPS outputs are selected. Bits[14:8] are valid only when two or more Flexible PPS outputs are selected. Bits[6:4] are valid only when Flexible PPS feature is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (MAC registers 96 and 97) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Reserved

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD. When this bit is reset, Bits[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCTRL[3:0]**: PPS Output Frequency Control

This field controls the frequency of the PPS output (ptp_pps_o) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:

0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.

0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.

0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.

0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

..

1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

Note: In the binary rollover mode, the PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- *When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms*
- *When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of*
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high)
- *When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of*
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

The PPS Control register is present only when the Timestamp feature is selected and External Timestamp is not enabled.

Bits[30:24] of this register are valid only when four Flexible PPS outputs are selected. Bits[22:16] are valid only when three or more Flexible PPS outputs are selected. Bits[14:8] are valid only when two or more Flexible PPS outputs are selected. Bits[6:4] are valid only when Flexible PPS feature is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (MAC registers 96 and 97) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Reserved

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD. When this bit is reset, Bits[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCMD[3:0]:** Flexible PPS Output (ptp_pps_o[0]) Control

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are 'all-zero'. The following list describes the values of PPSCMD0:

0000: No Command

0001: START Single Pulse

This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.

0010: START Pulse Train

This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.

0011: Cancel START

This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.

0100: STOP Pulse train at time

This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programmed in the Target Time registers elapses.

0101: STOP Pulse Train immediately

This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010).

0110: Cancel STOP Pulse train

This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.

0111-1111: Reserved

PPS target time seconds register (ETH_MACPPSTTSR)

Address offset: 0x0B80

Reset value: 0x0000 0000

The PPS Target Time Seconds register, along with PPS Target Time Nanoseconds register, is used to schedule an interrupt event [Bit 1 of ETH_MACTSSR] when the system time exceeds the value programmed in these registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSTRH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTRH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSTRH0[31:0]**: PPS Target Time Seconds Register

This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the ETH_MACPPSCR register.

PPS target time nanoseconds register (ETH_MACPPSTTNR)

Address offset: 0x0B84

Reset value: 0x0000 0000

The PPS Target Time Nanoseconds register is present only when more than one Flexible PPS output is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRGTBUSY0	TTSL0[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TRGTBUSY0**: PPS Target Time Register Busy

The MAC sets this bit when the PPSCMD0 field in the ETH_MACPPSCR register is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain.

The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

Bits 30:0 **TTSL0[30:0]**: Target Time Low for PPS Register

This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in ETH_MACPPSCR.

When the TSCTRLSSR bit is set in the ETH_MACTSCR register, this value should not exceed 0x3B9A_C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value.

PPS interval register (ETH_MACPPSIR)

Address offset: 0x0B88

Reset value: 0x0000 0000

The PPS Interval register contains the number of units of sub-second increment value between the rising edges of PPS signal output (ptp_pps_o[0]).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSINT0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSINT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSINT0[31:0]**: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of sub-second increment value.

You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of sub-second increment value), you should program value 4 (5-1) in this register.

PPS width register (ETH_MACPPSWR)

Address offset: 0x0B8C

Reset value: 0x0000 0000

The PPS Width register contains the number of units of sub-second increment value between the rising and corresponding falling edges of PPS signal output (ptp_pps_o).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSWIDTH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSWIDTH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSWIDTH0[31:0]**: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of sub-second increment value.

You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of sub-second increment value), you should program value 3 (4-1) in this register.

Note: The value programmed in this register must be lesser than the value programmed in ETH_MACPP0IR register.

PTP Offload control register (ETH_MACPOCR)

Address offset: 0x0BC0

Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation. This register is available only when the Enable PTP Timestamp Offload feature is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DN[7:0]								Res.	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DN[7:0]**: Domain Number
This field indicates the domain Number in which the PTP node is operating.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DRRDIS**: Disable PTO Delay Request/Response response generation
When this bit is set, the Delay Request and Delay response will not be generated for received SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 **APDREQTRIG**: Automatic PTP Pdelay_Req message Trigger
When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the APDREQEN bit for this operation.

Bit 4 **ASYNCTRIG**: Automatic PTP SYNC message Trigger
When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the ASYNCEN bit for this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **APDREQEN**: Automatic PTP Pdelay_Req message Enable
When this bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer Transparent mode.

Bit 1 **ASYNCEN**: Automatic PTP SYNC message Enable
When this bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Clock Master mode.

Bit 0 **PTOEN**: PTP Offload Enable
When this bit is set, the PTP Offload feature is enabled.

PTP Source Port Identity 0 Register (ETH_MACSPI0R)

Address offset: 0x0BC4

Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node. This register is available only when the Enable PTP Timestamp Offload feature is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI0[31:0]**: Source Port Identity 0
This field indicates bits [31:0] of sourcePortIdentity of PTP node.

PTP Source port identity 1 register (ETH_MACSPI1R)

Address offset: 0x0BC8

Reset value: 0x0000 0000

This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node. This register is available only when the Enable PTP Timestamp Offload feature is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI1[31:0]**: Source Port Identity 1
This field indicates bits [63:32] of sourcePortIdentity of PTP node.

PTP Source port identity 2 register (ETH_MACSPI2R)

Address offset: 0x0BCC

Reset value: 0x0000 0000

This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SPI2[15:0]**: Source Port Identity 2
This field indicates bits [79:64] of sourcePortIdentity of PTP node.

Log message interval register (ETH_MACLMIR)

Address offset: 0x0BD0

Reset value: 0x0000 0000

This register contains the periodic intervals for automatic PTP packet generation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LMPDRI[7:0]**: Log Min Pdelay_Req Interval

This field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11 Reserved, must be kept at reset value.

Bits 10:8 **DRSYNCR[2:0]**:

Delay_Req to SYNC Ratio

In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.

0: DelayReq generated for every received SYNC

1: DelayReq generated every alternate reception of SYNC

2: for every 4 SYNC messages

3: for every 8 SYNC messages

4: for every 16 SYNC messages

5: for every 32 SYNC messages

6-7: Reserved

The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0 **LSI[7:0]**:

Log Sync Interval

This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Ethernet MAC register map and reset values

Table 557. Ethernet MAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	ETH_MACCR	ARPEN	SARC [2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	Res.	JD	JE	Res.	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL [1:0]		DC	PRELEN[1:0]		TE	RE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0004	ETH_MACECR	Res.	Res.	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC	Res.	Res.	GPSL[13:0]														
	Reset value			0	0	0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	ETH_MACPFR	RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE	Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
	Reset value	0										0	0				0						0	0	0	0	0	0	0	0	0	0	0
0x000C	ETH_MACWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
	Reset value																								0					0	0	0	0
0x0010	ETH_MACHT0R	HT31T0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	ETH_MACHT1R	HT63T32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018 - 0x004C	Reserved																																
0x0050	ETH_MACVTR	EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV	VL[15:0]																
	Reset value	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	Reserved																																
0x0058	ETH_MACVHTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLHT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	Reserved																																
0x0060	ETH_MACVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC [1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	ETH_MACIVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC [1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068 - 0x006C	Reserved																																
0x0070	ETH_ MACQTXFCR	PT[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0			0	0
0x0074 - 0x008C	Reserved																																

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0090	ETH_MACRXFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
	Reset value																															0	0
0x0094	Reserved																																
0x00A0 - 0x00A8	Reserved																																
0x00AC	Reserved																																
0x00B0	ETH_MACISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	Res. S
	Reset value																		0	0	0		0	0	0			0	0	0			
0x00B4	ETH_MACIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	Res.
	Reset value																		0	0	0							0	0	0			0
0x00B8	ETH_ MACRXTXSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
	Reset value																								0			0	0	0	0	0	
0x00BC	Reserved																																
0x00C0	ETH_MACPCSR	RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
	Reset value	0			0	0	0	0	0														0	0			0	0			0	0	0
0x00C4	ETH_ MACRWKPFR	WKUPFRMFTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8 - 0x00CC	Reserved																																
0x00D0	ETH_MACLCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIATE	LPITXA	PLSEN	PLS	LPIEN	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
	Reset value												0	0	0	0	0						0	0					0	0	0	0
0x00D4	ETH_MACLTCR	Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]											TWT[15:0]														
	Reset value							1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								0x03E8											0														
0x00D8	ETH_MACLETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[16:0]																	Res.	Res.	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00DC	ETH_ MAC1USTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x00E0 - 0x00F8	Reserved																																

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00FC - 0x010C	Reserved																																
0x0110	ETH_MACVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERVER[7:0]								SNPSVER[7:0]							
	Reset value																	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1
0x0114	ETH_MACDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
	Reset value														0	0	0														0	0	0
0x0118	Reserved																																
0x0120	ETH_MACHWF1R	Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN	ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					SPRAM		RXFIFOSIZE[4:0]			
	Reset value		0	0	1	0		0	1	1		0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0
0x0124	ETH_MACHWF2R	Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			Res.	Res.	TXCHCNT[3:0]				Res.	Res.	RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
	Reset value		1	0	0		0	0	1			0	0	0	0			0	0	0	0			0	0	0	0			0	0	0	0
0x012C - 0x01FC	Reserved																																
0x0200	ETH_MACMDIOAR	Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]					Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	MB
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0
0x0204	ETH_MACMDIODR	RA[15:0]																MD[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0208 - 0x020C	Reserved																																
0x0AE0	ETH_MACARPAR	ARPPA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0214 - 0x02FC	Reserved																																
0x0300	ETH_MACA0HR	AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	1																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0304	ETH_MACA0LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0308	ETH_MACA1HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30C	ETH_MACA1LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0310	ETH_MACA2HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0314	ETH_MACA2LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0318	ETH_MACA3HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x031C	ETH_MACA3LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0320 - 0x06FC	Reserved																																
0x0700	MMC_CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
	Reset value																								0			0	0	0	0	0	0
0x0704	MMC_RX_ INTERRUPT	Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0708	MMC_TX_ INTERRUPT	Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x070C	MMC_RX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0710	MMC_TX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0714 - 0x0748	Reserved																																
0x074C	TX_SINGLE_ COLLISION_ GOOD_PACKETS	TXSNGLCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0750	TX_MULTIPLE_ COLLISION_ GOOD_PACKETS	TXMULTCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0754 - 0x0764	Reserved																																
0x0768	TX_PACKET_ COUNT_GOOD	TXPKTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x076C - 0x0790	Reserved																																
0x0794	RX_CRC_ERROR_ PACKETS	RXCRCERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0798	RX_ALIGNMENT_ ERROR_PACKETS	RXALGNERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x079C - 0x07C0	Reserved																																
0x07C4	RX_UNICAST_ PACKETS_GOOD	RXUCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C8 - 0x07E8	Reserved																																
0x07EC	TX_LPI_USEC_ CNTR	TXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F0	TX_LPI_TRAN_ CNTR	TXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F4	RX_LPI_USEC_ CNTR	RXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F8	RX_LPI_TRAN_ CNTR	RXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07FC - 0x08FC	Reserved																																

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0900	ETH_ MACL3L4C0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0	L3HDBM0 [4:0]					L3HSBM0 [4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0904	ETH_MACL4A0R	L4DP0[15:0]																L4SP0[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0908 - 0x090C	Reserved																																
0x0910	ETH_MACL3A00R	L3A00[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0914	ETH_MACL3A10R	L3A10[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0918	ETH_MACL3A20R	L3A20[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x091C	ETH_MACL3A30R	L3A30[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0920 - 0x092C	Reserved																																
0x0930	ETH_MACL3L4C1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1	L3HDBM1 [4:0]					L3HSBM 1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0934	ETH_MACL4A1R	L4DP1[15:0]																L4SP1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0938 - 0x093C	Reserved																																
0x0940	ETH_MACL3A01R	L3A01[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0944	ETH_MACL3A11R	L3A11[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0948	ETH_MACL3A21R	L3A21[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094C	ETH_MACL3A31R	L3A31[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0950 - 0x0AFC	Reserved																																
0x0B00	ETH_MACTSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	CSC	TSENMACADDR	SNAPTYPSEL[1:0]		TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	Res.	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
	Reset value				0				0					0	0	0	0	0	0	1	0	0	0	0	0			0		0	0	0	0

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B04	ETH_MACSSIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]								RESERVED_SNSINC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0B08	ETH_MACSTSR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0C	ETH_MACSTNR	Res.	TSSS[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B10	ETH_MACSTSUR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B14	ETH_MACSTNUR	ADDSUB	TSSS[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B18	ETH_MACTSAR	TSAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B1C	Reserved																																
0x0B20	ETH_MACTSSR	Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]				TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
	Reset value			0	0	0	0	0	0					0	0	0	0	0												0	0	0	0
0x0B24 - 0x0B2C	Reserved																																
0x0B30	ETH_MACTXTSSNR	TXTSSMIS	TXTSSLO[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B34	ETH_MACTXTSSSR	TXTSSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B38 - 0x0B3C	Reserved																																
0x0B40	ETH_MACACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
	Reset value																									0	0	0	0				0
0x0B44	Reserved																																

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B48	ETH_MACATSNR	Res.	AUXTSLO[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4C	ETH_MACATSSR	AUXTSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B50	ETH_MACTSIACR	OSTIAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B54	ETH_MACTSEACR	OSTEAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B58	ETH_ MACTSICNR	TSIC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B5C	ETH_ MACTSECNR	TSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B60 - 0x0B6C	Reserved																																
0x0B70	ETH_MACPPSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
	Reset value																										0	0	0	0	0	0	0
0x0B70	ETH_MACPPSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
	Reset value																										0	0	0	0	0	0	0
0x0B74 - 0x0B7C	Reserved																																
0x0B80	ETH_ MACPPSTTSR	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B84	ETH_ MACPPSTTNR	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B88	ETH_MACPPSIR	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 557. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B8C	ETH_MACPPSWR	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B90 - 0x0BBC	Reserved																																
0x0BC0	ETH_MACPOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DN[7:0]								Res.	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
	Reset value																	0	0	0	0	0	0	0	0		0	0	0		0	0	0
0x0BC4	ETH_MACSPI0R	SPI0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC8	ETH_MACSPI1R	SPI1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BCC	ETH_MACSPI2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BD0	ETH_MACLMIR	LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
	Reset value	0	0	0	0	0	0	0	0														0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

58 HDMI-CEC controller (HDMI-CEC)

58.1 Introduction

Consumer electronics control (CEC) is part of HDMI (high-definition multimedia interface) standard as appendix supplement 1. It contains a protocol that provides high-level control functions between various audiovisual products. CEC operates at low speeds, with minimum processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

58.2 HDMI-CEC controller main features

- Complies with HDMI-CEC v1.4 specification
- Independent 32 kHz CEC kernel (refer to *Section RCC kernel clock distribution*)
- Works in Stop mode for ultra-low-power applications
- Configurable signal-free time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable peripheral address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration lost detection (ARBLST)
  - With automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)

58.3 HDMI-CEC functional description

58.3.1 HDMI-CEC pin and internal signals

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up must be added externally to the microcontroller.

To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This can be done by using a MOS transistor, as shown on *Figure 805*.

Table 559 lists the internal signals that are exchanged between the HDMI-CEC and other functional blocks (such as RCC and EXTI).

Table 558. HDMI pin

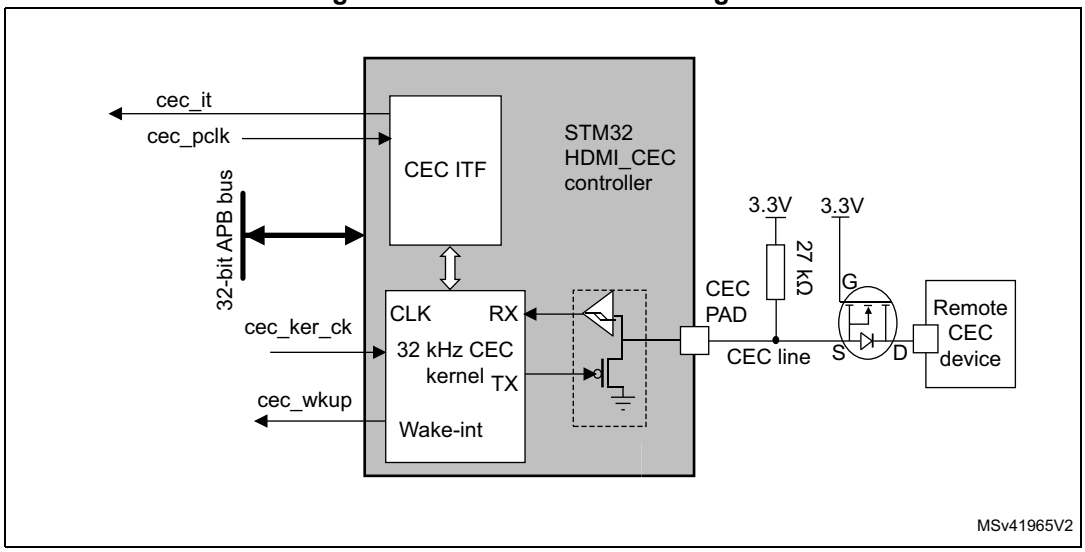
Name	Signal type	Remarks
CEC	Bidirectional	Two states: – 1 = high impedance – 0 = low impedance A 27 kΩ resistor must be added externally.

Table 559. HDMI-CEC internal input/output signals

Signal name	Signal type	Description
cec_wkup	Digital output	HDMI-CEC wakeup signal
cec_it	Digital output	HDMI-CEC interrupt signal
cec_pclk	Digital input	APB clock
cec_ker_ck	Digital input	HDMI-CEC kernel clock

58.3.2 HDMI-CEC block diagram

Figure 805. HDMI-CEC block diagram



58.3.3 Message description

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgment bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.

All these blocks are made of a 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In case a message contains additional blocks after an EOM is indicated, those additional blocks must be ignored. The EOM bit may be set in the header block to ‘ping’ other devices, to make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header, or by the follower that needs to reject a broadcast message.

The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

Figure 806. Message structure

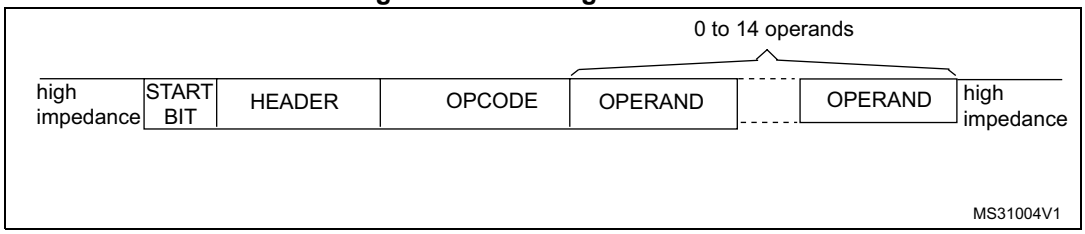
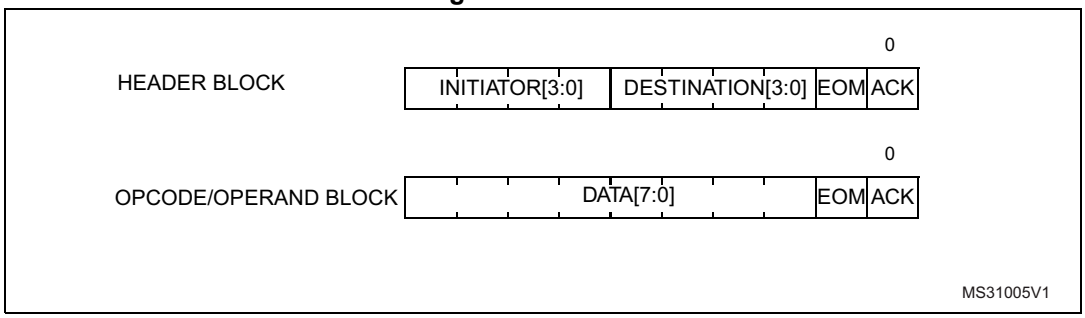


Figure 807. Blocks

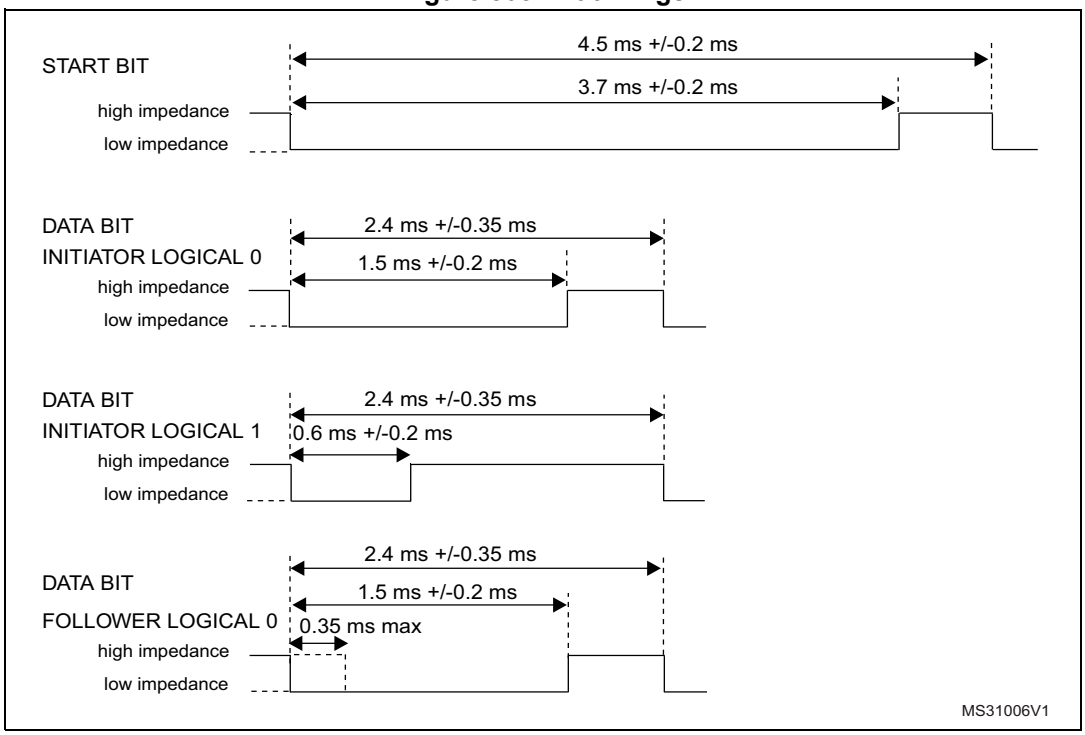


58.3.4 Bit timing

The format of the start bit is unique and identifies the start of a message. It must be validated by its low duration and its total duration.

All remaining data bits in the message, after the start bit, have consistent timing. The high-to-low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.

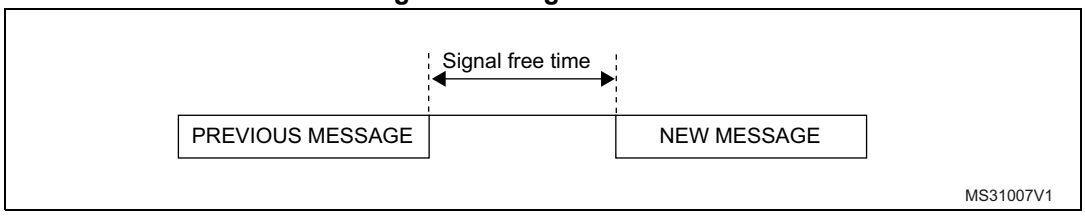
Figure 808. Bit timings



58.4 Arbitration

All devices transmitting - or retransmitting - a message onto the CEC line must ensure that it has been inactive for a number of bit periods. This signal-free time is defined as the time starting from the final bit of the previous frame and depends on the initiating device and the current status as shown in the table below.

Figure 809. Signal free time



Since only one initiator is allowed at any one time, an arbitration mechanism is provided to avoid conflict when more than one initiator begins transmitting at the same time.

CEC line arbitration starts with the leading edge of the start bit and continues until the end of the initiator address bits within the header block. During this period, the initiator must monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0. Assuming then it has lost arbitration, it stops transmitting and becomes a follower.

Figure 810. Arbitration phase

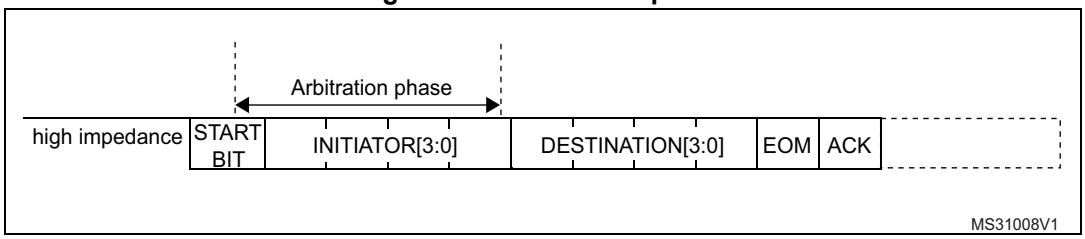
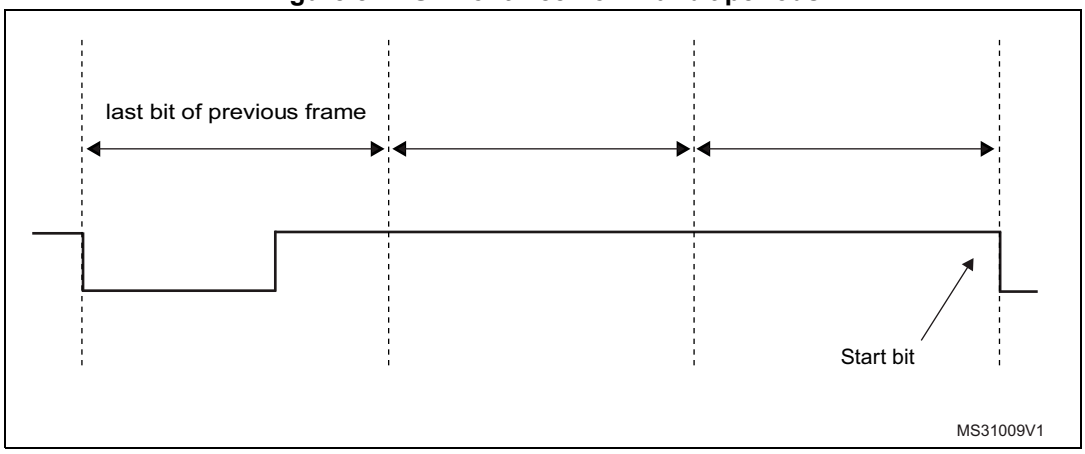


Figure 811 shows an example for a SFT of three nominal bit periods

Figure 811. SFT of three nominal bit periods



A configurable time window is counted before starting the transmission.

In the SFT = 0 configuration, HDMI-CEC performs automatic SFT calculation ensuring compliance with the HDMI-CEC standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission

This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFT bits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

58.4.1 SFT option bit

In case of SFTOPT = 0 configuration, SFT starts being counted when the start-of-transmission command is set by software (TXSOM=1).

In case of SFTOPT=1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT timer is still running instead, the system waits until the timer elapses before transmission can start.

In case of SFTOPT = 1 a bus-event condition starting the SFT timer is detected in the following cases:

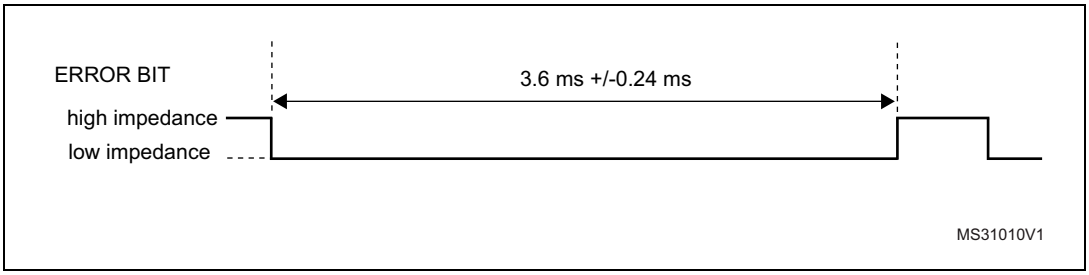
- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR=1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACKE bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any uncodified low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

58.5 Error handling

58.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period, i.e. 3.6 ms nominally.

Figure 812. Error bit timing



58.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

58.5.3 Bit Rising Error (BRE)

BRE (bit rising error): is set when a bit rising edge is detected outside the windows where it is expected (see *Figure 813*). BRE flag also generates a CEC interrupt if the BREIE=1.

In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP=1 an error bit is generated even if BREGEN=0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN=0, BRDNOGEN=1.

58.5.4 Short Bit Period Error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see *Figure 813*). SBPE flag also generates a CEC interrupt if the SBPEIE=1.

An error bit is always generated on the line in case of a SBPE error detection. An Error Bit is not generated upon SBPE detection only when Listen mode is set (LSTN=1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN=1

58.5.5 Long Bit Period Error (LBPE)

LBPE is set when a bit falling edge is not detected in a valid window (see *Figure 813*). LBPE flag also generates a CEC interrupt if the LBPEIE=1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN=0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN=0, BRDNOGEN=1.

Note: The BREGEN=1, BRESTP=0 configuration must be avoided

Figure 813. Error handling

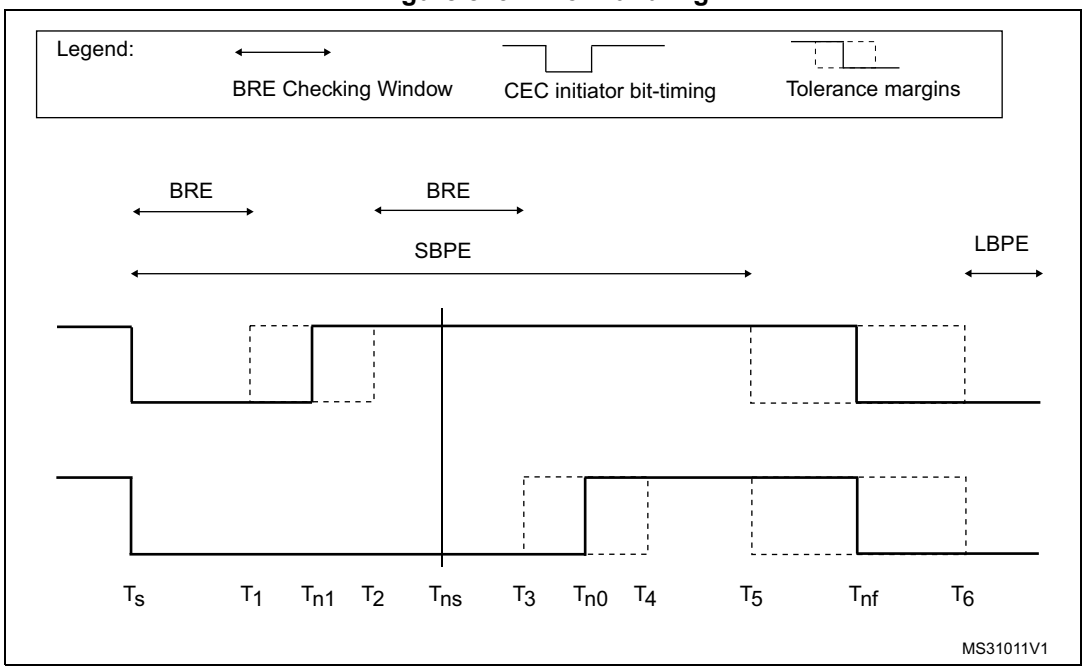


Table 560. Error handling timing parameters

Time	RXTOL	ms	Description
T_s	x	0	Bit start event.
T_1	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
T_{n1}	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
T_2	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
T_{ns}	x	1.05	Nominal sampling time.
T_3	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	
T_{n0}	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).

Table 560. Error handling timing parameters (continued)

Time	RXTOL	ms	Description
T_4	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T_5	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T_{nf}	x	2.4	The nominal data bit period.
T_6	0	2.75	The latest time for the start of a following bit.
	1	2.95	

58.5.6 Transmission Error Detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE=1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to 5 times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

Figure 814. TXERR detection

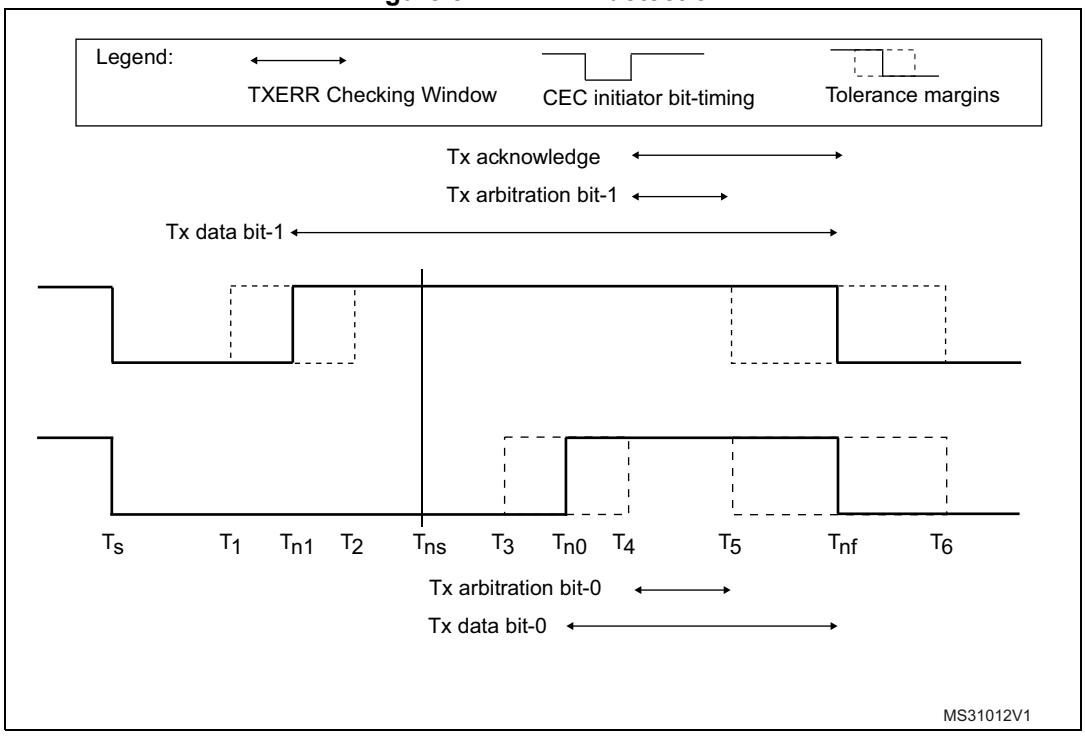


Table 561. TXERR timing parameters

Time	RXTOL	ms	Description
T_s	x	0	Bit start event.
T_1	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
T_{n1}	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
T_2	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
T_{ns}	x	1.05	Nominal sampling time.
T_3	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	

Table 561. TXERR timing parameters (continued)

Time	RXTOL	ms	Description
T_{n0}	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
T_4	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T_5	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T_{nf}	x	2.4	The nominal data bit period.
T_6	0	2.75	The latest time for the start of a following bit.
	1	2.95	

58.6 HDMI-CEC interrupts

An interrupt can be produced:

- during reception if a Receive Block Transfer is finished or if a Receive Error occurs.
- during transmission if a Transmit Block Transfer is finished or if a Transmit Error occurs.

Table 562. HDMI-CEC interrupts

Interrupt event	Event flag	Enable Control bit
Rx-Byte Received	RXBR	RXBRIE
End of reception	RXEND	RXENDIE
Rx-Overrun	RXOVR	RXOVRIE
RxBit Rising Error	BRE	BREIE
Rx-Short Bit Period Error	SBPE	SBPEIE
Rx-Long Bit Period Error	LBPE	LBPEIE
Rx-Missing Acknowledge Error	RXACKE	RXACKEIE
Arbitration lost	ARBLST	ARBLSTIE
Tx-Byte Request	TXBR	TXBRIE
End of transmission	TXEND	TXENDIE
Tx-Buffer Underrun	TXUDR	TXUDRIE
Tx-Error	TXERR	TXERRIE
Tx-Missing Acknowledge Error	TXACKE	TXACKEIE

58.7 HDMI-CEC registers

Refer to *Section 1.2 on page 97* for a list of abbreviations used in register descriptions.

58.7.1 CEC control register (CEC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEOM	TXSOM	CECEN
													rs	rs	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TXEOM***:* Tx End Of Message

The TXEOM bit is set by software to command transmission of the last byte of a CEC message.
TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.
0: TXDR data byte is transmitted with EOM=0
1: TXDR data byte is transmitted with EOM=1

Note: TXEOM must be set when CECEN=1.
TXEOM must be set before writing transmission data to TXDR.
If TXEOM is set when TXSOM=0, transmitted message will consist of 1 byte (HEADER) only (PING message)

Bit 1 **TXSOM**: Tx Start Of Message

TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM.
Start-Bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission will start after the end of reception.
TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND=1), in case of transmission underrun (TXUDR=1), negative acknowledge (TXACKE=1), and transmission error (TXERR=1). It is also cleared by CECEN=0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST=1).
TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.
0: No CEC transmission is on-going
1: CEC transmission command

Note: TXSOM must be set when CECEN=1.
TXSOM must be set when transmission data is available into TXDR.
HEADER first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC_CFGR.OAR which is used only for reception.

Bit 0 **CECEN:** CEC enable

The CECEN bit is set and cleared by software. CECEN=1 starts message reception and enables the TXSOM control. CECEN=0 disables the CEC peripheral, clears all bits of CEC_CR register and aborts any on-going reception or transmission.
0: CEC peripheral is off
1: CEC peripheral is on

58.7.2 CEC configuration register (CEC_CFGR)

This register is used to configure the HDMI-CEC controller.

Address offset: 0x04

Reset value: 0x0000 0000

Caution: It is mandatory to write CEC_CFGR only when CECEN=0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSTN	OAR[14:0]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFT OP	BRDN OGEN	LBPE GEN	BRE GEN	BRE STP	RX TOL	SFT[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LSTN:** Listen mode

LSTN bit is set and cleared by software.

0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.

1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

Bits 30:16 **OAR[14:0]:** Own addresses configuration

The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.

At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN=1), but without acknowledge sent. Broadcast messages are always received.

Example:

OAR = 0b000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5. Consequently, each message directed to one of these addresses is received.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **SFTOP**: SFT Option Bit

The SFTOPT bit is set and cleared by software.

0: SFT timer starts when TXSOM is set by software

1: SFT timer starts automatically at the end of message transmission/reception.

Bit 7 **BRDNOGEN:** Avoid Error-Bit Generation in Broadcast

The BRDNOGEN bit is set and cleared by software.

0: BRE detection with BRESTP=1 and BREGEN=0 on a broadcast message generates an Error-Bit on the CEC line. LBPE detection with LBPEGEN=0 on a broadcast message generates an Error-Bit on the CEC line

1: Error-Bit is not generated in the same condition as above. An Error-Bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.

Bit 6 **LBPEGEN:** Generate Error-Bit on Long Bit Period Error

The LBPEGEN bit is set and cleared by software.

0: LBPE detection does not generate an Error-Bit on the CEC line

1: LBPE detection generates an Error-Bit on the CEC line

Note: If BRDNOGEN=0, an Error-bit is generated upon LBPE detection in broadcast even if LBPEGEN=0

Bit 5 **BREGEN:** Generate Error-Bit on Bit Rising Error

The BREGEN bit is set and cleared by software.

0: BRE detection does not generate an Error-Bit on the CEC line

1: BRE detection generates an Error-Bit on the CEC line (if BRESTP is set)

Note: If BRDNOGEN=0, an Error-bit is generated upon BRE detection with BRESTP=1 in broadcast even if BREGEN=0

Bit 4 **BRESTP**: Rx-Stop on Bit Rising Error

The BRESTP bit is set and cleared by software.

0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.

1: BRE detection stops message reception

Bit 3 **RXTOL:** Rx-Tolerance

The RXTOL bit is set and cleared by software.

0: Standard tolerance margin:

- Start-Bit, +/- 200 µs rise, +/- 200 µs fall.
- Data-Bit: +/- 200 µs rise. +/- 350 µs fall.

1: Extended Tolerance

- Start-Bit: +/- 400 µs rise, +/- 400 µs fall
- Data-Bit: +/-300 µs rise, +/- 500 µs fall

Bits 2:0 **SFT[2:0]**: Signal Free Time

SFT bits are set by software. In the SFT=0x0 configuration the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.

- 0x0
  - 2.5 Data-Bit periods if CEC is the last bus initiator with unsuccessful transmission (ARBLST=1, TXERR=1, TXUDR=1 or TXACKE= 1)
  - 4 Data-Bit periods if CEC is the new bus initiator
  - 6 Data-Bit periods if CEC is the last bus initiator with successful transmission (TXEOM=1)
- 0x1: 0.5 nominal data bit periods
- 0x2: 1.5 nominal data bit periods
- 0x3: 2.5 nominal data bit periods
- 0x4: 3.5 nominal data bit periods
- 0x5: 4.5 nominal data bit periods
- 0x6: 5.5 nominal data bit periods
- 0x7: 6.5 nominal data bit periods

58.7.3 CEC Tx data register (CEC_TXDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXD[7:0]**: Tx data

TXD is a write-only register containing the data byte to be transmitted.

58.7.4 CEC Rx data register (CEC_RXDR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXD[7:0]**: Rx data

RXD is read-only and contains the last data byte which has been received from the CEC line.

58.7.5 CEC Interrupt and Status Register (CEC_ISR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TX ACKE	TX ERR	TX UDR	TX END	TXBR	ARB LST	RX ACKE	LBPE	SBPE	BRE	RX OVR	RX END	RXBR
			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKE:** Tx-Missing Acknowledge Error

In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.

TXACKE is cleared by software write at 1.

Bit 11 **TXERR:** Tx-Error

In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.

TXERR is cleared by software write at 1.

Bit 10 **TXUDR:** Tx-Buffer Underrun

In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.

TXUDR is cleared by software write at 1

Bit 9 **TXEND:** End of Transmission

TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.

TXEND is cleared by software write at 1.

Bit 8 **TXBR:** Tx-Byte Request

TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within 6 nominal data-bit periods before transmission underrun error occurs (TXUDR).

TXBR is cleared by software write at 1.

Bit 7 **ARBLST:** Arbitration Lost

ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.

ARBLST is cleared by software write at 1.

Bit 6 **RXACKE:** Rx-Missing Acknowledge

In receive mode, RXACKE is set by hardware to inform application that no acknowledge was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception.

RXACKE is cleared by software write at 1.

Bit 5 **LBPE:** Rx-Long Bit Period Error

LBPE is set by hardware in case a Data-Bit waveform is detected with Long Bit Period Error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an Error-Bit on the CEC line if LBPEGEN=1. In case of broadcast, Error-Bit is generated even in case of LBPEGEN=0.

LBPE is cleared by software write at 1.

Bit 4 **SBPE:** Rx-Short Bit Period Error

SBPE is set by hardware in case a Data-Bit waveform is detected with Short Bit Period Error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an Error-Bit on the CEC line.

SBPE is cleared by software write at 1.

Bit 3 **BRE***:* Rx-Bit Rising Error

BRE is set by hardware in case a Data-Bit waveform is detected with Bit Rising Error. BRE is set either at the time the misplaced rising edge occurs, or at the end of the maximum BRE tolerance allowed by RXTOL, in case rising edge is still longing. BRE stops message reception if BRESTP=1. BRE generates an Error-Bit on the CEC line if BREGEN=1.

BRE is cleared by software write at 1.

Bit 2 **RXOVR***:* Rx-Overrun

RXOVR is set by hardware if RXBR is not yet cleared at the time a new byte is received on the CEC line and stored into RXD. RXOVR assertion stops message reception so that no acknowledge is sent. In case of broadcast, a negative acknowledge is sent.

RXOVR is cleared by software write at 1.

Bit 1 **RXEND***:* End Of Reception

RXEND is set by hardware to inform application that the last byte of a CEC message is received from the CEC line and stored into the RXD buffer. RXEND is set at the same time of RXBR.

RXEND is cleared by software write at 1.

Bit 0 **RXBR***:* Rx-Byte Received

The RXBR bit is set by hardware to inform application that a new byte has been received from the CEC line and stored into the RXD buffer.

RXBR is cleared by software write at 1.

58.7.6 CEC interrupt enable register (CEC_IER)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACKIE	TXERRIE	TX UDRIE	TXENDIE	TXBRIE	ARBLSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKIE***:* Tx-Missing Acknowledge Error Interrupt Enable

The TXACKEIE bit is set and cleared by software.

0: TXACKE interrupt disabled

1: TXACKE interrupt enabled

Bit 11 **TXERRIE***:* Tx-Error Interrupt Enable

The TXERRIE bit is set and cleared by software.

0: TXERR interrupt disabled

1: TXERR interrupt enabled

Bit 10 **TXUDRIE***:* Tx-Underrun Interrupt Enable

The TXUDRIE bit is set and cleared by software.

0: TXUDR interrupt disabled

1: TXUDR interrupt enabled

Bit 9 **TXENDIE***:* Tx-End Of Message Interrupt Enable
The TXENDIE bit is set and cleared by software.
0: TXEND interrupt disabled
1: TXEND interrupt enabled

Bit 8 **TXBRIE***:* Tx-Byte Request Interrupt Enable
The TXBRIE bit is set and cleared by software.
0: TXBR interrupt disabled
1: TXBR interrupt enabled

Bit 7 **ARBLSTIE***:* Arbitration Lost Interrupt Enable
The ARBLSTIE bit is set and cleared by software.
0: ARBLST interrupt disabled
1: ARBLST interrupt enabled

Bit 6 **RXACKIE***:* Rx-Missing Acknowledge Error Interrupt Enable
The RXACKIE bit is set and cleared by software.
0: RXACKE interrupt disabled
1: RXACKE interrupt enabled

Bit 5 **LBPEIE***:* Long Bit Period Error Interrupt Enable
The LBPEIE bit is set and cleared by software.
0: LBPE interrupt disabled
1: LBPE interrupt enabled

Bit 4 **SBPEIE***:* Short Bit Period Error Interrupt Enable
The SBPEIE bit is set and cleared by software.
0: SBPE interrupt disabled
1: SBPE interrupt enabled

Bit 3 **BREIE***:* Bit Rising Error Interrupt Enable
The BREIE bit is set and cleared by software.
0: BRE interrupt disabled
1: BRE interrupt enabled

Bit 2 **RXOVRIE***:* Rx-Buffer Overrun Interrupt Enable
The RXOVRIE bit is set and cleared by software.
0: RXOVR interrupt disabled
1: RXOVR interrupt enabled

Bit 1 **RXENDIE***:* End Of Reception Interrupt Enable
The RXENDIE bit is set and cleared by software.
0: RXEND interrupt disabled
1: RXEND interrupt enabled

Bit 0 **RXBRIE***:* Rx-Byte Received Interrupt Enable
The RXBRIE bit is set and cleared by software.
0: RXBR interrupt disabled
1: RXBR interrupt enabled

Caution: (*) It is mandatory to write CEC_IER only when CECEN=0

58.7.7 HDMI-CEC register map

The following table summarizes the HDMI-CEC registers.

Table 563. HDMI-CEC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CEC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEOM	TXSOM	CECEN
	Reset value																														0	0	0
0x04	**CEC_CFGR**	LSTN	OAR[14:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFTOPT	BRDNOGEN	LBPEGEN	BREGEN	BRESTP	RXTOL	SFT[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x08	**CEC_TXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C	**CEC_RXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x10	**CEC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKE	TXERR	TXUDR	TXEND	TXBR	ARBLST	RXACKE	LBPE	SBPE	BRE	RXOVR	RXEND	RXBR
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**CEC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3.2 on page 124* for the register boundary addresses.

59 Debug infrastructure

59.1 Introduction

The debug infrastructure allows software designers to debug and trace their embedded software.

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The trace and debug system has been designed to support a variety of typical use cases:

- **Low cost trace**

  Limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt detection and program counter sampling. Single-wire trace can be maintained even when the processor is switched off or clock-stopped.
- **Breakpoint debugging**

  The processor can be debugged using equipment connected to the JTAG/SWD debug port. This allows breakpoint and watchpoint setting, code stepping, memory access etc.
- **Tracing code execution via the trace port**

  Trace information is combined into a single trace stream and output to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.
- **Capturing trace continuously in a circular buffer**

  Instead of streaming it off-chip, the combined trace information can be stored on-chip in a circular buffer. The trace storage can be started and stopped by a debugger command, a software command, an external trigger signal, an internal event, etc.
- **Draining the buffer to the trace port**

  The stored trace can be dumped off-chip to the trace port analyzer. The buffer draining can be initiated by the debugger, software, external trigger, internal event etc.
- **Reading the buffer with the debugger**

  The debugger can read the contents of the trace buffer via the debug port. This is slower than the trace port, but allows basic trace functionality on the debugger without the cost of a trace port analyzer.
- **Analyzing stored trace in software**

  The trace buffer can be read by the processor core, or transferred into system memory by DMA. This powerful feature allows built-in test software to monitor code execution in real time, analyze and identify faults, autonomously handle exceptions, etc.
- **Uploading stored trace**

  The stored trace can also be uploaded to a host machine using one of the MCU’s many communications interfaces (USB, USART, SPI, I2C, Ethernet, CAN etc). This is especially useful if the trace port is not accessible, for example remote monitoring and failure analysis of a deployed product.

59.2 Debug infrastructure features

A comprehensive set of trace and debug features is provided to support software development and system integration:

- Breakpoint debugging
- Code execution tracing
- Software instrumentation
- Cross-triggering
- JTAG debug port
- Serial-wire debug port
- Trigger input and output
- Serial-wire trace port
- Trace port
- Arm® CoreSight™ debug and trace components

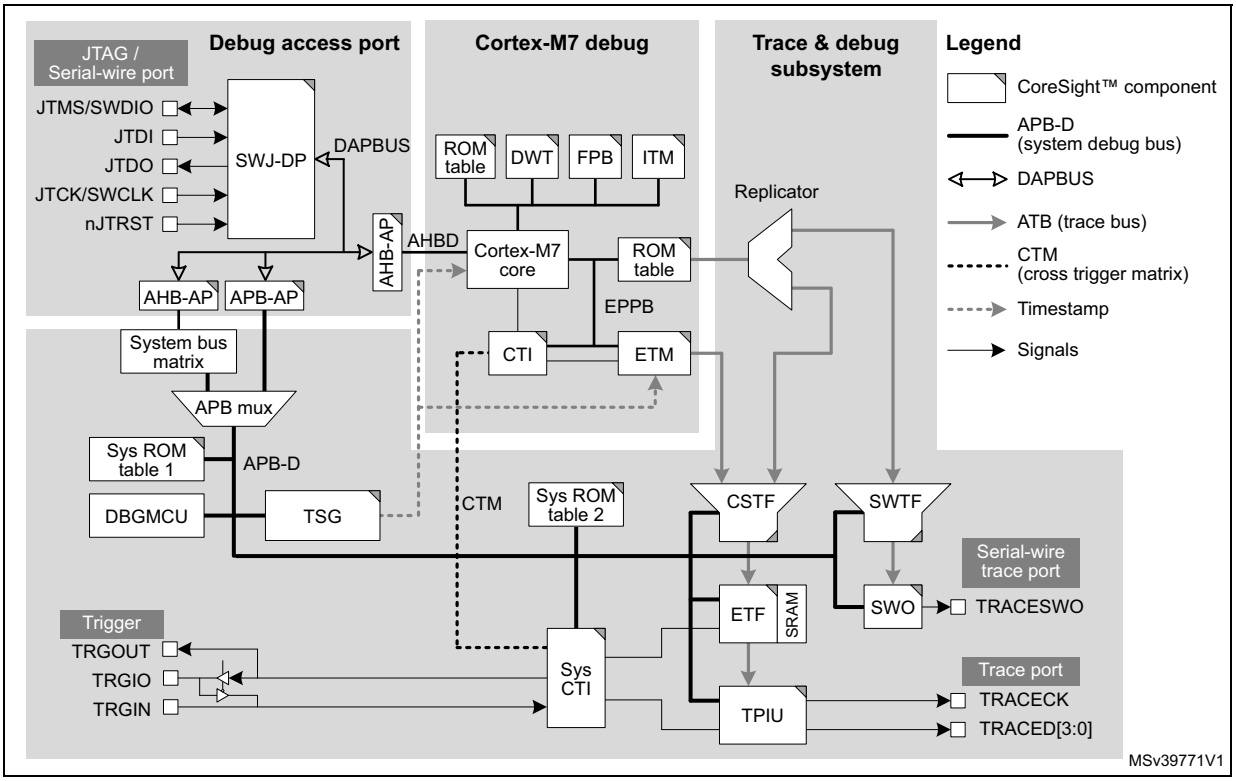
The CoreSight components are described at high level in this document. Detailed information is available in the Arm® documents referenced in *Section 59.7*.

59.3 Debug infrastructure functional description

59.3.1 Debug infrastructure block diagram

The block diagram shows the logical partitioning of the debug infrastructure.

Figure 815. Block diagram of debug infrastructure



59.3.2 Debug infrastructure pins and internal signals

Table 564. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial wire clock	PA14
JTDI	I	JTAG test data input	-	-	PA15
JTDO	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

Table 565. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to datasheet
TRACED1	O	Trace synchronous data out 1	
TRACED2	O	Trace synchronous data out 2	
TRACED3	O	Trace synchronous data out 3	
TRACECK	O	Trace clock	

Table 566. Serial-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that single wire trace is only available when using the serial wire debug interface, not when using JTAG

Table 567. Trigger pins

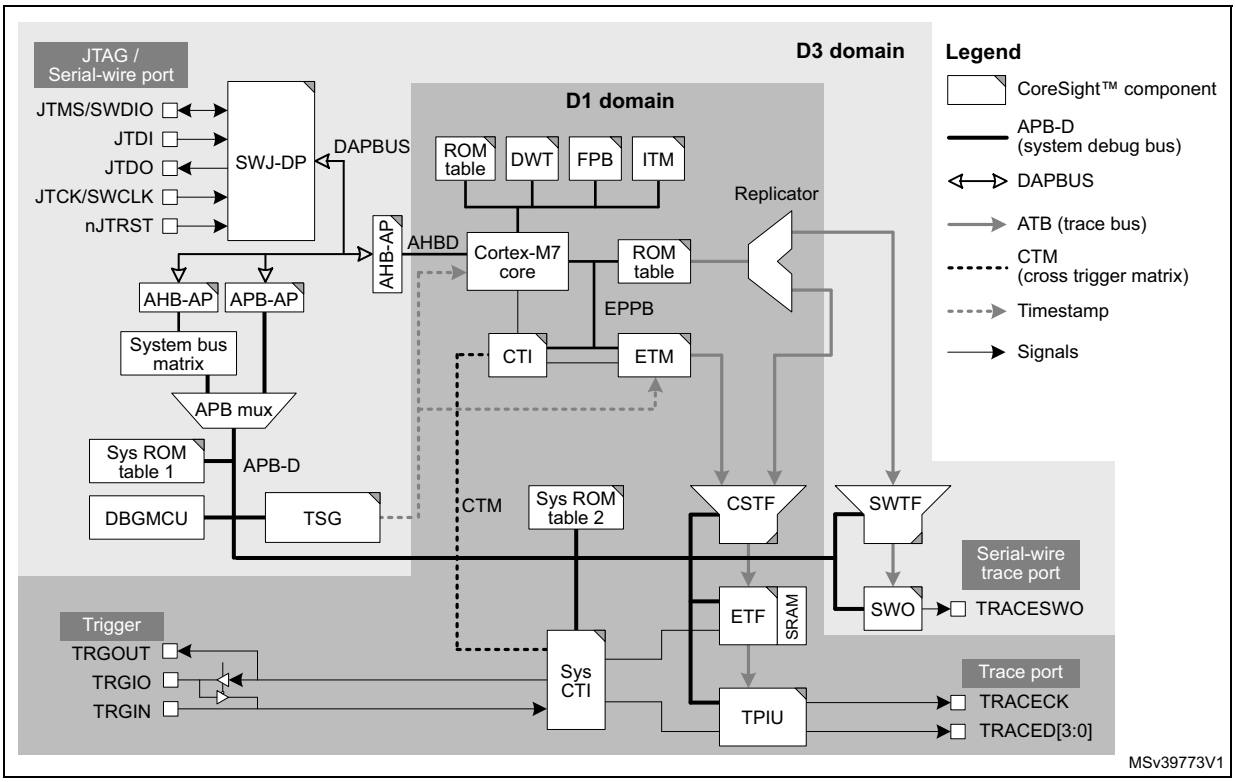
Pin name	Type	Description	Pin assignment
TRGIN	I	External trigger input	Refer to datasheet
TRGOUT	O	External trigger output	
TRGIO	IO	External trigger bi-directional(1)	

1. TRGIO can be configured as an input or an output by the TRGOEN bit in the DBGMCU. If configured as an input, it is connected to TRGIN. If an output, it is connected to TRGOUT. This is because TRGIN and TRGOUT are not available on certain packages.

59.3.3 Debug infrastructure powering, clocking and reset

Power domains

Figure 816. Power domains of debug infrastructure

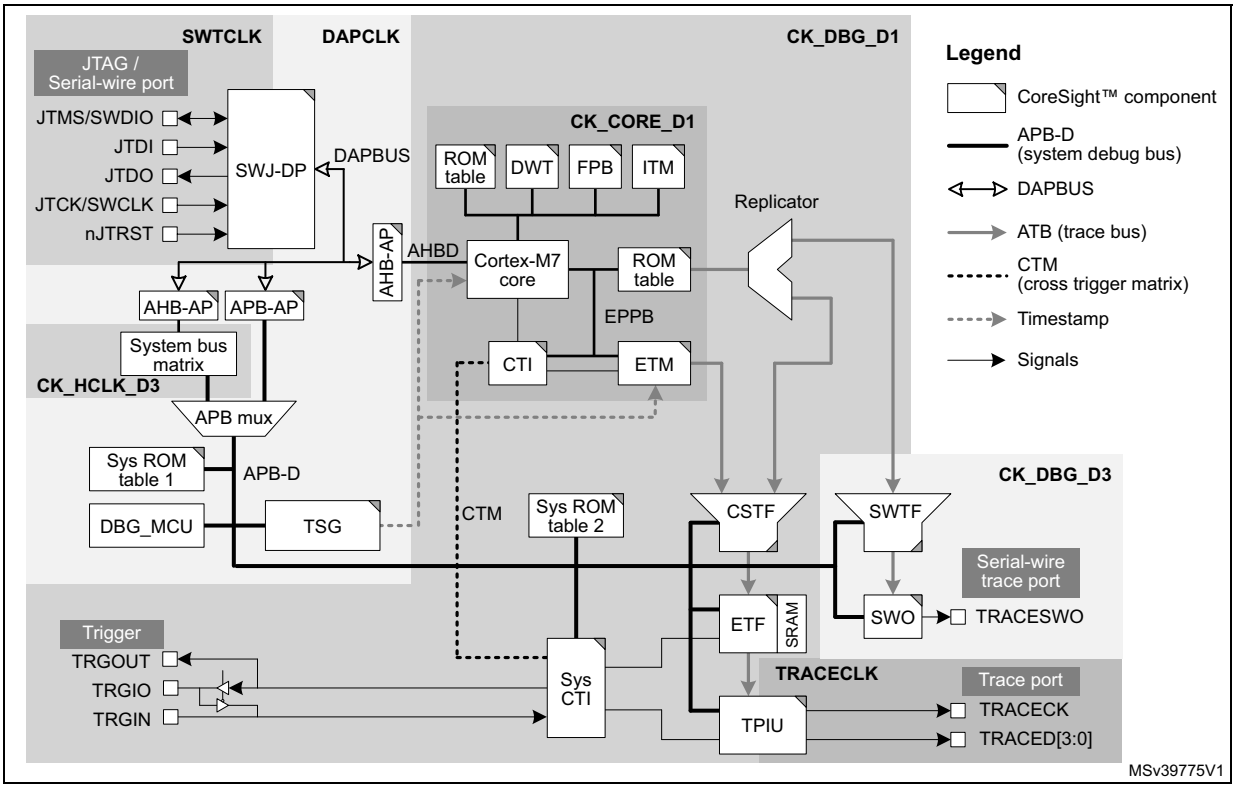


The debug components are distributed across the power domains D1 and D3. The D3 power domain is considered to be always on when the debugger is connected. It therefore contains the SWJ-DP, so that the debugger does not lose the connection with the SoC when the other power domain is switched off. In addition, it contains the timestamp generator, the DBGMCU and the serial wire trace features.

The D1 power domain contains the Cortex-M7 core and the associated debug and trace components. It also contains the system trace components located on the APB-D. This power domain therefore needs to be on whenever a debug access to the Cortex-M7 is required, or whenever a trace functionality is active on the processor.

Clock domains

Figure 817. Clock domains of debug infrastructure



The debugger supplies the clock for the debug port, SWTCLK, via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DAPCLK domain, which covers the rest of the SWJ-DP and the access ports. The DBGMCU, timestamp generator and System ROM table 1 are also in the DAPCLK domain.

CK_DBG_D3 clocks the SWO and serial wire trace funnel.

Both DAPCLK and CK_DBG_D3 are gated versions of the D3 domain system clock (CK_HCLK_D3).

CK_DBG_D1 clocks the trace components in the D1 power domain: System ROM table 2, CoreSight trace funnel, ETF, system CTI and TPIU. It is a gated version of the D1 domain system clock (CK_HCLK_D1).

TRACECLK is the trace port output clock. It is a gated version of the system clock (CK_SYS), except when the PLL1 is the source for the system clock. In this case, TRACECLK is derived directly from the PLL1 VCO output, divided by three. This is required in order to support the high data throughput on the trace port when the processor operates at its maximum frequency.

All the debug clocks (except DAPCLK) can be enabled and disabled by register bits in the DBGMCU. The DAPCLK domain is enabled by the debugger using the CDBGPWRUPREQ

bit in the debug port CTRL/STAT register. The clock must be enabled before the debugger can access any of the debug features on the device. It should be disabled at power up and when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor (ETM ITM, DWG, FPB etc) are clocked with the corresponding core clock (CK_CORE_D1).

Debug with low-power modes

The device includes power-saving features allowing individual power domains to be switched off or stopped when not required. If a power domain is switched off or not clocked, all debug components in that domain are inaccessible to the debugger. To avoid this, power saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode, while the debugger does not lose the connection.

The emulation mode is programmed in the MCU Debug (DBGMCU) unit. For more information, refer to *Section 59.5.8*

Reset of debug infrastructure

The debug components, except for the debug port and access ports, are reset by their respective power domain resets. The debug port (SWJ-DP) is reset by a power-on reset of the D3 domain only.

59.4 Debug access port functional description

The debug access port (DAP) is a debug subsystem comprising serial-wire and JTAG debug port (SWJ-DP) and three access ports.

59.4.1 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

The port can be configured as:

- a 5-pin standard JTAG debug port (JTAG-DP)
- a 2-pin (clock + data) “serial-wire” debug port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected upon a system or power-on reset. The five IOs are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on the JTDI, JTMS/SWDIO, and nJTRST lines, as well as a pull-down resistor on the JTCK/SWCLK line.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

...,(50 or more ones),...,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,...,(50 or more ones),...

JTCK/SWCLK must be cycled for each data bit.

In SW-DP mode, the unused JTAG lines JTDI, JTDO and nJTRST can be used for other functions.

All SWJ port IOs can be reconfigured to other functions by software, in which case debugging is no longer possible.

Serial wire debug port

The Serial wire debug protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three phases:

1. packet request (8 bits) transmitted by the host
2. acknowledge response (3 bits) transmitted by the target
3. data transfer (33 bits) transmitted by the host (in the case of a write) or target (in the case of a read)

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 568. Packet request

Field bits	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP register access - see *Table 572* for a list of DP registers 1: AP register access - see *Section 59.4.2*
2	RnW	0: Write request 1: Read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 572* and *Table 573*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host. Must be read as “1” by target.

Table 569. ACK response

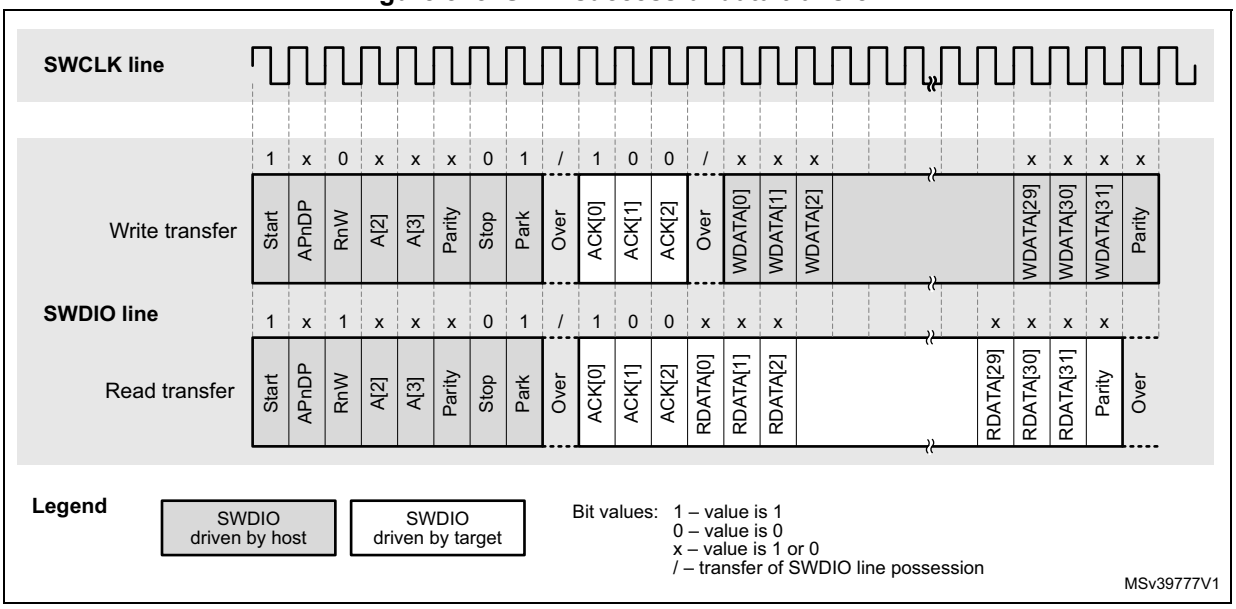
Field bits	Name	Description
2:0	ACK	000b: FAULT 010b: WAIT 100b: OK

Table 570. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or Read data
32	Parity	Single bit parity of 32 data bits

Figure 818 shows successful write and read transfers.

Figure 818. SWD successful data transfer



In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is cancelled, unless overrun detection is enabled, in which case the data will be ignored by the target (in the case of a write), or not driven (in the case of a read).

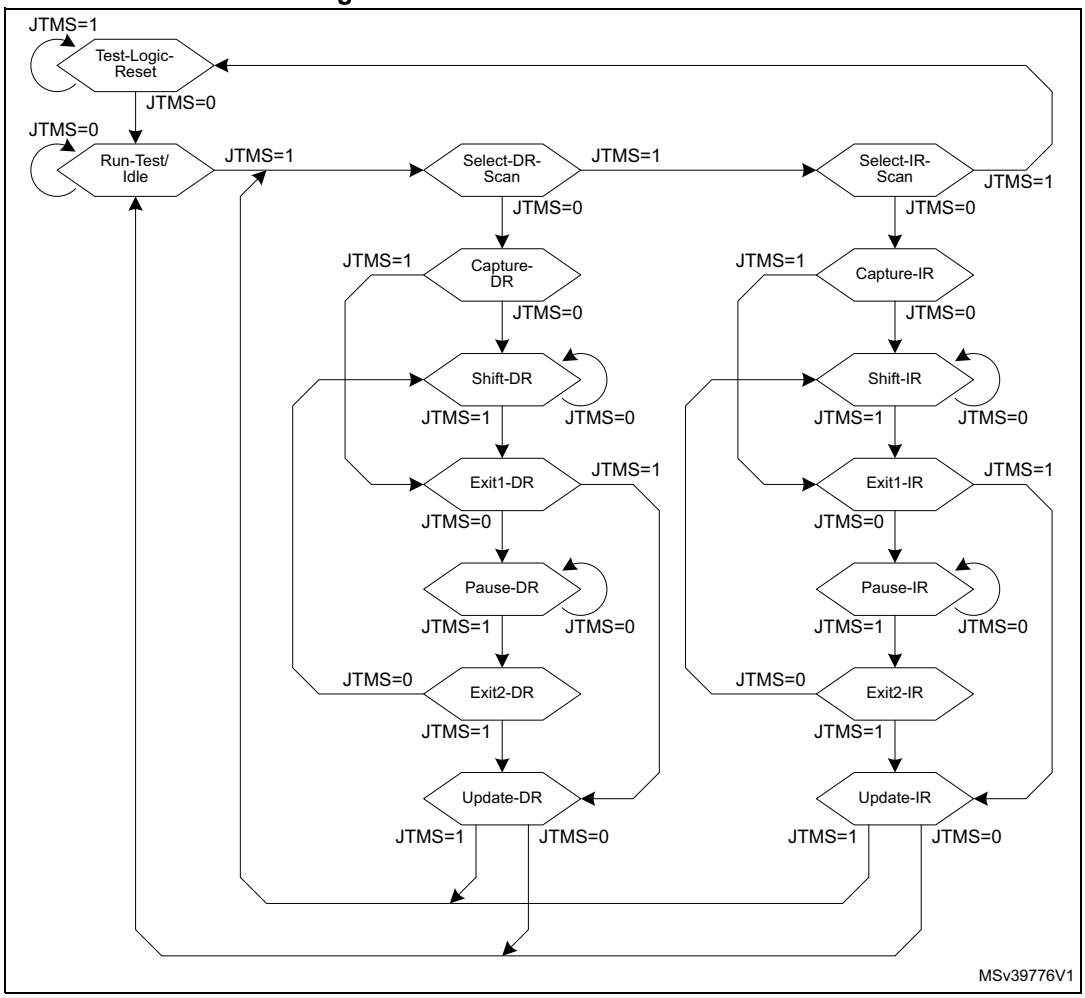
A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists of 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial Wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

JTAG debug port

Figure 819. JTAG TAP state machine



The JTAG-DP implements a TAP state machine (TAPSM) based on IEEE 1149.1-1990. The state machine is shown in *Figure 819*. It controls two scan chains, one associated with an instruction register (IR) and one with a number of data registers (DR).

When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.

While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:

- The LSB of the IR scan chain is output on JTDO.
- Bit [n] of the IR scan chain is transferred to bit [n-1].
- The value on JTDI is transferred to the MSB of the IR scan chain.

When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the instruction register.

When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.

The value held in the instruction register determines which data register, and associated DR scan chain, is selected.

This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.

When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected data register.

When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in *Table 571*.

Table 571. JTAG-DP data registers

Instruction register	Data register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	**Abort register** – Bits 31:1 = reserved – Bit 0 = APABORT: write 1 to generate an AP abort
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	**Debug port access register** Initiates the debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010b = OK/FAULT 001b = WAIT OTHER = reserved

Table 571. JTAG-DP data registers (continued)

Instruction register	Data register	Scan chain length	Description
1011	APACC	35	**Access port access register** Initiates an access port and allows access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register. Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010b = OK/FAULT 001b = WAIT OTHER = reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	**ID Code** 0x05BA 0477: Arm® JTAG debug port ID code
1111	BYPASS	1	**Bypass** A single JTCK cycle delay is inserted between JTDI and JTDO

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification *[1]*.

Debug port registers

The SW-DP and JTAG-DP both access the debug port (DP) registers. These are listed in *Table 572*.

The debugger can access the DP registers as follows:

1. Program the SELECT register DPBANKSEL field in the DP to select the register bank to be accessed (see *Table 572*)
2. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the R/W bit to select a read or a write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and R/W fields are part of the Packet Request word sent to the SW-DP with the APnDP bit reset (see *Table 568*). The write data is sent in the data phase.

Table 572. Debug port registers

Address	A(3:2) field value	R/W	Description
0x0	00	R	DP_DPIDR register. It contains the IDCODE for the debug port.
		W	DP_ABORT register(1). It aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRL/STAT register.
0x4	01	R/W	If DPBANKSEL[3:0] = 0x0 (DP_SELECT register): CTRL/STAT register. It controls the DP and provides status information.
			If DPBANKSEL[3:0] = 0x1 (DP_SELECT register): DP_DLCR register(2). It controls the operating mode of the SWD Data Link.
			If DPBANKSEL[3:0] = 0x2 (DP_SELECT register): DP_TARGETID register. It provides target identification information.
			If DPBANKSEL[3:0] = 0x3 (DP_SELECT register): DLPIDR register(2). It provides the SWD protocol version.
0x8	10	R	RESEND register(2). It returns the value that was returned by the last AP read or DP_RDBUFF read, used in the event of a corrupted read transfer.
		W	DP_SELECT register. It selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	DP_RDBUFF register Via JTAG-DP, it is used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation). Via SW-DP, it contains the result of the preceding AP read access, allowing a new AP access to be avoided.
		W	DP_TARGETSEL register(2). On a write to DP_TARGETSEL immediately following a line reset sequence, the target is selected if the following conditions are both met: – Bits [31:28] match bits [31:28] in the DP_DLPIDR register. – Bits [27:0] match bits [27:0] in the DP_TARGETID register. Writing any other value deselects the target. Debug tools must write 0xFFFFFFFF to deselect all targets. This is an invalid DP_TARGETID value. All other invalid DP_TARGETID values are reserved.

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

Debug port identification register (DP_DPIDR)

Address offset: 0x00

Reset value: 0x6BA0 2477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: Revision code
0x6

Bits 27:20 **PARTNO[7:0]**: Part number for the debug port
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: Minimal debug port (MINDP) implementation
0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm®

Bit 0 Reserved, must be kept at reset value.

Debug port abort register (DP_ABORT)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	STKCMPCLR	DAPABORT
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: Overrun error clear bit
0: No effect
1: Clear CTRL/STAT register's STICKYORUN bit

Bit 3 **WDERRCLR**: Write data error clear bit
0: No effect
1: Clear CTRL/STAT register's WDATAERR bit

Bit 2 **STKERRCLR**: Sticky error clear bit
0: No effect
1: Clear CTRL/STAT register's STICKYERR bit

Bit 1 **STKCMPCLR**: Sticky compare clear bit
0: No effect
1: Clear CTRL/STAT register's STICKYCMP bit

Bit 0 **DAPABORT**: Abort current AP transaction
The transaction is aborted if an excessive number of WAIT responses are returned, indicating that the transaction has stalled.

0: No effect
1: Abort transaction

Debug port control/status register (DP_CTRL/STAT)

Address offset: 0x4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSYSPWRUPACK	CSYSPWRUPREQ	CDBGPWRUPACK	CDBGPWRUPREQ	CDBGRSTACK	CDBGRSTREQ	Res.	Res.	TRNCNT[11:4]							
r	rw	r	rw	r	rw			rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRNCNT[3:0]				MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	w	w	w	w	w

Bit 31 **CSYSPWRUPACK**: System domain power-up status bit - not used in this device

Bit 30 **CSYSPWRUPREQ**: System domain power-up control bit - not used in this device

Bit 29 **CDBGPWRUPACK**: Debug domain power-up status bit
This bit is read-only. It returns the status of the debug domain power-up acknowledge signal from the power controller.

0: domain powered down
1: domain powered up

Bit 28 **CDBGPWRUPREQ:** Debug domain power-up/down control bit
This bit controls the debug domain power-up/down request signal to the power controller.

0: power-down requested
1: power-up requested

Bit 27 **CDBGRSTACK**: Debug domain reset status bit - not used in this device

Bit 26 **CDBGRSTREQ**: Debug domain reset control bit - not used in this device

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:12 **TRNCNT[11:0]**: Transaction counter

To program a sequence of transactions to incremental addresses via an AP, TRNCNT bits are loaded with the number of transactions to perform. It is decremented at the successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]**: Pushed-compare and pushed-verify masking bits

The field indicates the bytes to be masked in pushed-compare and pushed-verify operations (DP_CTRL/STAT register's field TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.

0b1XXX: include byte lane 3 in comparisons
0bX1XX: include byte lane 2 in comparisons
0bXX1X: include byte lane 1 in comparisons
0bXXX1: include byte lane 0 in comparisons

Bit 7 **WDATAERR**: Write data error in SW-DP

The bit indicates that

- there is a parity or a framing error on the data phase of a write operation, or
- a write operation that had been accepted by the DP has then be discarded without being submitted to the AP

This bit is read-only. It is reset by writing 1 to the WDERRCLR bit of the DP_ABORT register.

0: No error
1: Error has occurred

This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response in SW-DP

This bit indicates the response to the last AP read access. It is read-only.

0: Read not OK
1: Read OK

This bit is Reserved in JTAG-DP.

Bit 5 **STICKYERR**: Transaction error (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that an error occurred during an AP transaction.

0: No error
1: Error has occurred

In the SW-DP, this bit is reset by writing 1 to the STKERRCLR bit of the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to 1.

Bit 4 **STICKYCMP**: Compare match (read-only in SW-DP, R/W in JTAG-DP)
This bit indicates that a match occurred in a pushed operation.

0: Match if TRNMODE = 0x1; no match if TRNMODE = 0x2
1: No match if TRNMODE = 0x1; match if TRNMODE = 0x2

In the SW-DP, this bit is reset by writing 1 to the STKCMPCLR bit in the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to it.

Bits 3:2 **TRNMODE[1:0]**: Transfer mode for AP write operations
For read operations, this field must be set to 0x0.

0x0: Normal operation - AP transactions are passed directly to the AP.
0x1: Pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read operation is compared with the stored data. If they do not match, the STICKYCMP bit is set.
0x2: Pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data. If they match, the STICKYCMP bit is set.
0x3: Reserved

In pushed operations, only the data bytes indicated by the MASKLANE field are included in the comparison.

Bit 1 **STICKYORUN:** Overrun (read-only in SW-DP, R/W in JTAG-DP)
This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.

0: No overrun
1: Overrun occurred

In the SW-DP, this bit is reset by writing 1 to the ABORT register's ORUNERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0 **ORUNDETECT:** Overrun detection mode enable
0: Overrun detection disabled
1: Overrun detection enabled. In the event of an overrun, the STICKYORUN bit will be set and subsequent transactions will be blocked until the STICKYORUN bit is cleared.

Debug port data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: Tristate period for SWDIO
0x0: 1 data bit period
0x1: 2 data bit periods
0x2: 3 data bit periods
0x3: 4 data bit periods

Bits 7:0 Reserved, must be kept at reset value.

Debug port target identification register (DP_TARGETID)

Address offset: 0x4

Reset value: 0x1048 0401

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: Target revision
0x1: revision 1

Bits 27:12 **TPARTNO[15:0]**: Target part number
0x0450: STM32H7

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

Debug port data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: Target instance number
These bits define the instance number for this device in a multi-drop system.
0x0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial Wire Debug protocol version
0x1: Version 2

Debug port resend register (DP_RESEND)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: Last AP read or DP RDBUFF read value
These bits contain the value that was returned by the last AP read or DP RDBUFF read. Used in the event of a corrupted read transfer.

Debug port access port select register (DP_SELECT)

Address offset: 0x8

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:28 **APSEL[3:0]**: Access port select bits
These bits select the access port for the next transaction.

0x0: AP0 - Cortex-M7 debug access port (AHB-AP)
0x1: AP1 - D3 access port (AHB-AP)
0x2: AP2 - System debug access port (APB-AP)
0x3 to 0xF: Reserved

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select bits
These bits select the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select bits
These bits select the register at address 0x4 of the debug port.

0x0: CTRL/STAT register
0x1: DLCR register
0x2: TARGETID register
0x3: DLPIDR register
0x4 to 0xF: Reserved

Debug port read buffer register (DP_RDBUFF)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: Last AP read value
The field contains the value returned by the last AP read access. There are two ways to retrieve the value returned by an AP read access:

– perform a second read access to the same address, which will initiate a new transaction on the corresponding bus, or
– read the value returned by the last AP read access from the DP_RDBUFF register, in which case no new AP transaction occurs

Debug port target identification register (DP_TARGETSEL)

Address offset: 0xC

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				TPARTNO[15:4]											
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	

Bits 31:28 **TINSTANCE[3:0]**: Target instance number

The field defines the instance number for the target device in a multi-drop system. It must be programmed with the same value as TINSTANCE field of DP_DLPIDR register, in order to select this device.

Bits 27:12 **TPARTNO[15:0]**: Target part number

The field defines the part number for the target device. It must be programmed with the same value as TPARTNO field of DP_TARGETID register, in order to select this device.

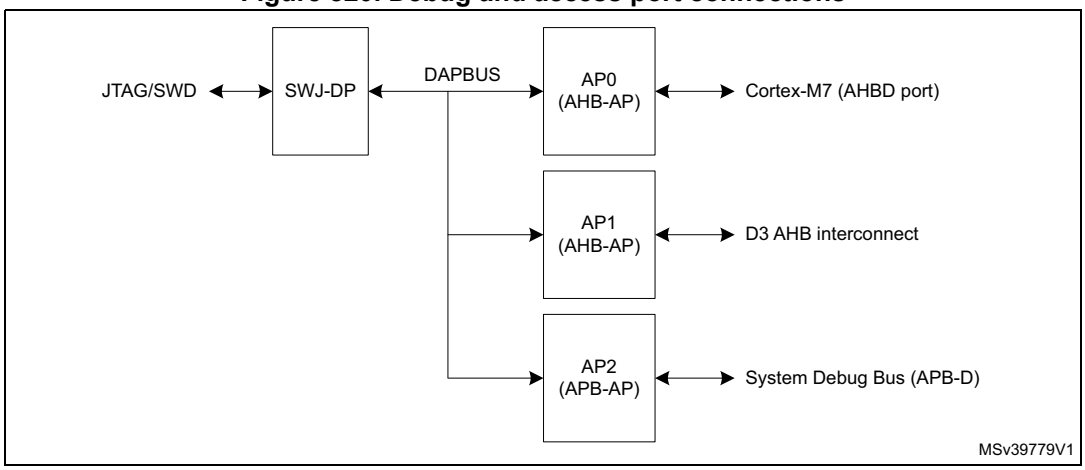
Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code

The field defines the JEDEC code for the target device. It must be programmed with the same value as TDESIGNER field of DP_TARGETID register, in order to select this device.

Bit 0 Reserved, must be kept at reset value.

59.4.2 Access ports

Figure 820. Debug and access port connections



There are the following access ports (AP) attached to the DP:

1. AP0: Cortex-M7 access port (AHB-AP). Allows access to the debug and trace features integrated in the Cortex-M7 processor core via an AHB-Lite bus connected to the AHBD port of the processor.
2. AP1: D3 access port (AHB-AP). Allows access to the bus matrix in the D3 domain. This gives visibility of the D3 domain memory and peripherals when the D1 and D2 domains are switched off. No CoreSight components are accessible via this port.
3. AP2: System access port (APB-AP). Allows access to the debug and trace features on the system APB debug bus, that is, all components not included in the processor core.

All access ports are of MEM-AP type, that is, the debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 573*.

The address of the AP registers is composed of:

- bits [7:4]: contents of the DP_SELECT register's APBANKSEL field
- bits [3:2]: contents of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 571*) or of the SW-DP Packet Request (see *Table 568*), depending on the debug interface used
- bits [1:0]: Always set to 0

The contents of the SELECT register APSEL field in the DP define which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the DP_SELECT register's APSEL field to choose one of the APs, and the APBANKSEL field to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or a write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit set (see *Table 568*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the MEM-AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to banked data register BDn triggers an access to address TAR[31:4] + n (this allows accessing up to four consecutive addresses without changing the address in the TAR register).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification *[1]*. The usage of MEM-AP to connect the debug port to the debug components (in the example, a processor, an ETM and a ROM table), is in the section 7.1.2.

MEM-AP registers

Table 573. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	AP_CSW	Control/status word register
0x04	0x0	1	AP_TAR	Transfer address register Target address for the bus transaction.
0x08	-	-	-	Reserved
0x0C	0x0	3	AP_DRW	Data read/write register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0]
0x10	0x1	0	AP_BD0	Banked data 0 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x0
0x14	0x1	1	AP_BD1	Banked Data 1 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x4
0x18	0x1	2	AP_BD2	Banked data 2 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x8
0x1C	0x1	3	AP_BD3	Banked data 3 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0xC
0x20-0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	-	-	-	Reserved
0xF8	0xF	2	AP_BASE	Debug base address register (RO) Base address of the ROM table
0xFC	0xF	3	AP_IDR	Identification register (RO)

Access port control/status word register (AP_CSW)

Address offset: 0x0

Reset value: 0x0000 0002 (APB-AP), 0x4000 0002 (AHB-AP)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	PROT[4:0]					SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw	rw	rw	rw	r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: Secure transfer request bit

In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attribute HPROT[6] of the bus transfer.

0: If SPIDEN is high, secure transfer. If SPIDEN is low, non-secure transfer.

1: Non-secure transfer.

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **PROT[4:0]** Bus transfer protection bits

In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attributes HPROT[4:0] of the bus transfer.

0bXXXX0: Instruction fetch

0bXXXX1: Data access

0bXXX0X: User mode

0bXXX1X: Privileged mode

0bXX0XX: Non-bufferable

0bXX1XX: Bufferable

0bX0XXX: Non-cacheable

0bX1XXX: Cacheable

0b0XXXX: Non-exclusive

0b1XXXX: Exclusive

Bit 23 **SPISTATUS**: Status of SPIDEN option bit

This bit determines whether the debugger can access secure memory. This field is reserved in the APB-AP.

0: Secure AHB transfers are blocked

1: Secure AHB transfers are allowed

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: Barrier support enabled bit

These bits define if memory barrier operation is supported.

0x0: Not supported

Bit 7 **TRINPROG**: Transfer in progress

This bit indicates that a bus transfer is in progress on the AP.

0: No transfer in progress.

1: Bus transfer in progress.

Bit 6 **DEVICEEN**: Device Enable bit
This bit defines whether the AP can be accessed or not.

1: AP access enabled.

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode bits
These bits define whether the TAR address is automatically incremented after a transaction.

0x0: no auto-increment
0x1: Address is incremented by the size in bytes of the transaction (SIZE field).
0x2: Packed transfers enabled (Only in AHB-APs - reserved in APB-AP). A 32-bit AP access will give rise to 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transactions corresponding to the programmed transaction size. The data will be packed or unpacked accordingly.
0x3: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction (only for AHB-APs)
0x0: Byte (8-bit)
0x1: Half-word (16-bit)
0x2: Word (32-bit)
0x3-0x7: Reserved

For APB-AP, this field is read-only and fixed at 0x2 (32-bit).

Access port base address register (AP_BASE)

Address offset: 0xF8

Reset value: 0xE00F E003 (AP0), 0x0000 0002 (AP1), 0xE00E 0003 (AP2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of ROM table for the AP
The 12 LSBs are zero since the ROM table must be aligned on a 4 Kbyte boundary.
AP0 (Cortex-M7 AHB-AP): 0xE00FE
AP1 (D3 AHB-AP): 0x00000 (No ROM table present)
AP2 (System APB-AP): 0xE00E0

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base address register format
1: Arm® debug interface v5.

Bit 0 **ENTRYPRESENT**: Debug component present status bit
This bit indicates that debug components are present on the access port bus.

0: Debug components are not present (AP1)
1: Debug components are present (AP0, AP2)

Access port identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x6477 0001 (AP0 and AP1), 0x4477 0002 (AP2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: Arm core revision
0x2: r0p3
0x4: r0p5
0x6: r0p7

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm®

Bit 16 **MEMAP**: Memory access port
1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: AP type identification
0x01: AHB-AP (AP0 and AP1)
0x02: APB-AP (AP2)
0x11: Reserved

59.5 Trace and debug subsystem functional description

The trace and debug subsystem features the following CoreSight components:

- System ROM tables
- Global timestamp generator (TSG)
- System cross-trigger interface (CTI)
- Cross-trigger matrix (CTM)
- Trace port interface unit (TPIU)
- Trace bus funnel (CSTF)
- Embedded trace FIFO (ETF)
- Serial wire output (SWO)
- Serial wire output trace funnel (SWTF)

These components are accessible by the debugger via the system APB-AP and its associated APB-D debug bus. They are also accessible by the Cortex-M7 processor.

The MCU debug unit (DBGMCU) is also accessed via the APB-D. This non-CoreSight component contains registers for configuring the behavior of the device in debug mode.

Trace bus replicator branches the trace bus from the CPU's ITM CoreSight component to ETF and SWO, through trace bus funnels.

59.5.1 System ROM tables

There are two ROM tables on the APB-D bus. The ROM table is a CoreSight component that contains the base addresses of all the CoreSight components on the APB-D bus. These tables allow a debugger to discover the topology of the CoreSight components automatically.

The first table points to the second table, and to the CoreSight components located in D3 power domain: SWO, SWTF, TSG. The DBGMCU is not referenced by the table as it is not a standard CoreSight component. The table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00E0000 to 0xE00E0FFC when accessed by the debugger, and from 0x5C000000 to 0x5C000FFC when accessed from the system bus.

Table 574. System ROM table 1

Address offset in ROM table	Component name	Component base address (debugger)	Component base address (system bus)	Component address offset	Size	Entry
0x000	System ROM table 2	0xE00F0000	0x5C010000	0x10000	4 Kbyte	0x00010003
0x004	SWO	0xE00E2000	0x5C003000	0x02000	4 Kbyte	0x00002003
0x008	Timestamp generator	0xE00E3000	0x5C005000	0x03000	4 Kbyte	0x00003003
0x00C	SWO funnel	0xE00E4000	0x5C004000	0x04000	4 Kbyte	0x00004003
0x010	Top of table	-	-	-	-	0x00000000
0x014 to 0xFC8	Reserved	-	-	-	-	0x00000000
0xFCC to 0xFFC	ROM table registers	-	-	-	-	See *System ROM registers*

The second table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00F0000 to 0xE00F0FFC when accessed by the debugger, and from 0x5C010000 to 0x5C010FFC when accessed from the system bus.

Table 575. System ROM table 2

Address offset in ROM table	Component name	Component base address (debugger)	Component base address (system bus)	Component address offset	Size	Entry
0x000	System CTI	0xE00F1000	0x5C011000	0x1000	4 Kbyte	0x00001003
0x004	Trace funnel	0xE00F3000	0x5C013000	0x3000	4 Kbyte	0x00003003
0x008	ETF	0xE00F4000	0x5C014000	0x4000	4 Kbyte	0x00004003
0x00C	TPIU	0xE00F5000	0x5C015000	0x5000	4 Kbyte	0x00005003
0x010	Top of table	-	-	-	-	0x00000000
0x014 to 0xFC8	Reserved	-	-	-	-	0x00000000
0xFCC to 0xFFC	ROM table registers	-	-	-	-	See *System ROM registers*

The top of each ROM table contains a number of read-only registers, including the standard CoreSight component and peripheral identity registers, see section *System ROM registers*.

Each debug component occupies one or more 4 Kbyte blocks of address space. This block of address space is referred to as the debug register file for the component.

The component address offset field of a ROM Table entry points to the start of the last 4 Kbyte block of the address space of the component. This block always contains the component and peripheral ID registers for the component, starting at offset 0xFD0 from the start of the block. The 4 Kbyte count field PIDR4 [7:4], specifies the number of 4 Kbyte blocks for the component. Therefore, the process for finding the start of the address space for a component is:

1. Read the ROM-table entry for the component and extract its Address_Offset[18:0] from bits [31:12] of the ROM-table entry.
2. Use the address offset, together with the base address of the ROM table, ROM_Base_Address, to calculate the base address of the component:

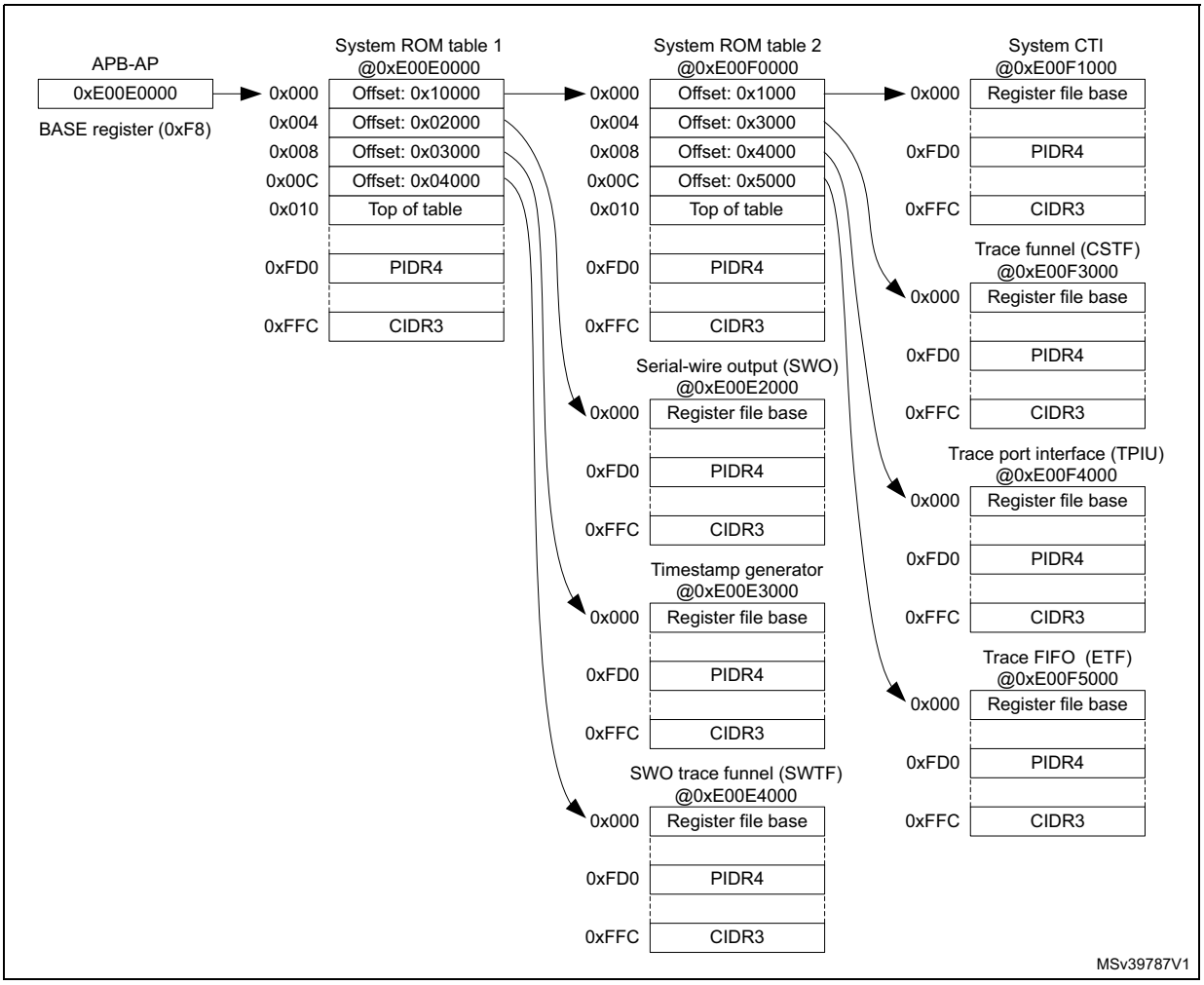
   Component_Base_Address = ROM_Base_Address + Address_Offset

   The Component_Base_Address is the start address of the 4 Kbyte block of the address space for the component.
3. Read the peripheral ID4 register for the component. The address of this register is:

   Peripheral_ID4_address = Component_Base_Address + 0xFD0
4. Extract the 4 Kbyte count field [7:4] from the value of the Peripheral ID4 Register.
5. Use the 4 Kbyte count field value to calculate the start address of the address space for the component. If the field value is 0b0000, which corresponds to a count value of 1, the address space for the component starts at Component_Base_Address obtained at stage 2.

The topology for the CoreSight components on the APB-D is shown in *Figure 821*.

Figure 821. APB-D CoreSight component topology



For more information on the use of the ROM table, refer to the Arm® Debug Interface Architecture Specification *[1]*.

System ROM registers

SYSROM memory type register (SYSROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory
0: No system memory is present on this bus

SYSROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

SYSROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Device part number field, bits [7:0]
0x50: STM32H7 device

SYSROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Device part number field, bits [11:8]
0x4: STM32H7 device

SYSROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Device revision number
0x1: Rev 1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x2: STMicroelectronics JEDEC code

SYSROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SYSROM CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SYSROM CoreSight component identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SYSROM CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SYSROM CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

System ROM register map and reset values

Table 576. System ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	SYSROM_ MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																0
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3;0			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	SYSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	0	1	0	0	0	0
0xFE4	SYSROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	SYSROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	SYSROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SYSROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SYSROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	SYSROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SYSROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.5.2 Global timestamp generator (TSG)

The global timestamp generator contains a 64-bit counter that provides a common timing reference for all of the trace sources in the system, namely the ETM and ITM in the processor core. These components insert timestamps in the trace streams that allow the trace analyzer to recover the chronological order of trace packets, which can be lost when multiple trace sources are multiplexed into one stream at the funnels.

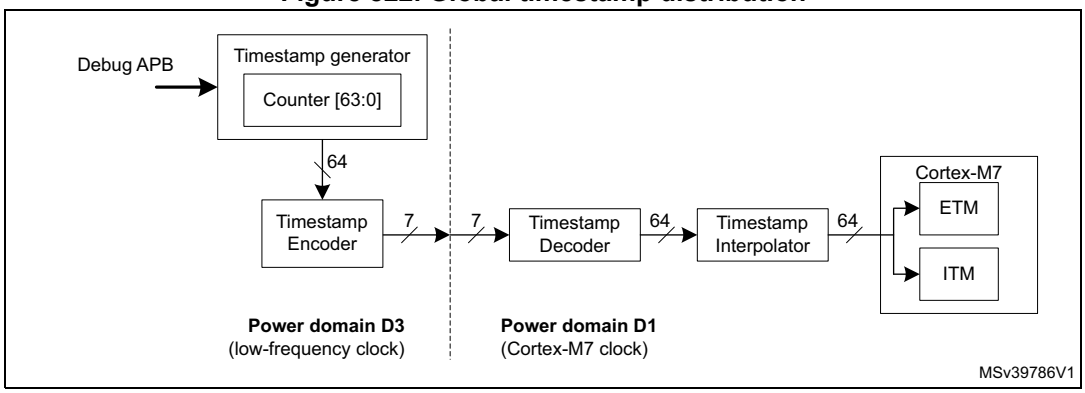
The TSG registers are accessible over the APB-D. This allows the debugger or debug software to:

- start and stop the timestamp incrementing
- read the current timestamp value
- change the current timestamp value
  - The timestamp counter must be halted while it is changed. When the timestamp value is changed, the timestamp generator resynchronizes all the trace sources.
- change the reported timestamp increment

For more information on the global timestamp generator CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

The timestamp generator is located in the D3 power domain, and the timestamp is distributed to the Cortex-M7. To simplify the distribution over power domain boundaries, the 64-bit timestamp is encoded in seven bits, then decoded in the destination power domain, and interpolated to increase its resolution if the processor clock is significantly faster than the generator clock. The timestamp distribution is shown in *Figure 822*.

Figure 822. Global timestamp distribution



TSG registers

TSG counter control register (TSG_CNTCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDBG	EN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HDBG**: Halt on debug
0: Normal operation
1: Halt counter when system-wide debug state is detected - not implemented

Bit 0 **EN:** Enable
0: Counter disabled
1: Counter enabled and incrementing

TSG counter status register (TSG_CNTSR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DBGH**: Debug halt
0: Normal operation
1: Counter halted due to system-wide debug state

Bit 0 Reserved, must be kept at reset value.

TSG current counter value lower register (TSG_CNTCVL)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVL[31:0]**: TSG current counter value field, bits[31:0]

To change the current timestamp value, write the lower 32 bits of the new value to this register before writing the upper 32 bits to CNTCVU. The timestamp value is not changed until the CNTVCVU register is written to. Note: The TSG_CNTCR register's EN bit must be cleared before writing to this register.

TSG current counter value upper register (TSG_CNTCVU)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVU[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVU[31:0]**: TSG current counter value field, bits[63:32]

To change the current timestamp value, write the lower 32 bits of the new value to CNTCVL before writing the upper 32 bits to this register. The 64-bit timestamp value is updated with the value from both writes when this register is written to. Note: The TSG_CNTCR register's EN bit must be cleared before writing to this register.

TSG base frequency ID register (TSG_CNTFID0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREQ[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FREQ[31:0]**: Increment frequency of TSG counter in Hz
This field must be programmed with the trace generator clock frequency whenever it changes.

TSG CoreSight peripheral identity register 4 (TSG_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TSG CoreSight Peripheral identity register 0 (TSG_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x01: TSG part number

TSG CoreSight peripheral identity register 1 (TSG_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x1: TSG part number

TSG CoreSight peripheral identity register 2 (TSG_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

TSG CoreSight peripheral identity register 3 (TSG_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

TSG CoreSight component identity register 0 (TSG_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

TSG CoreSight component identity register 1 (TSG_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xF: CoreSight Soc-400 component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

TSG CoreSight component identity register 2 (TSG_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

TSG CoreSight component identity register 3 (TSG_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

TSG register map and reset values

Table 577. TSG register map and reset values

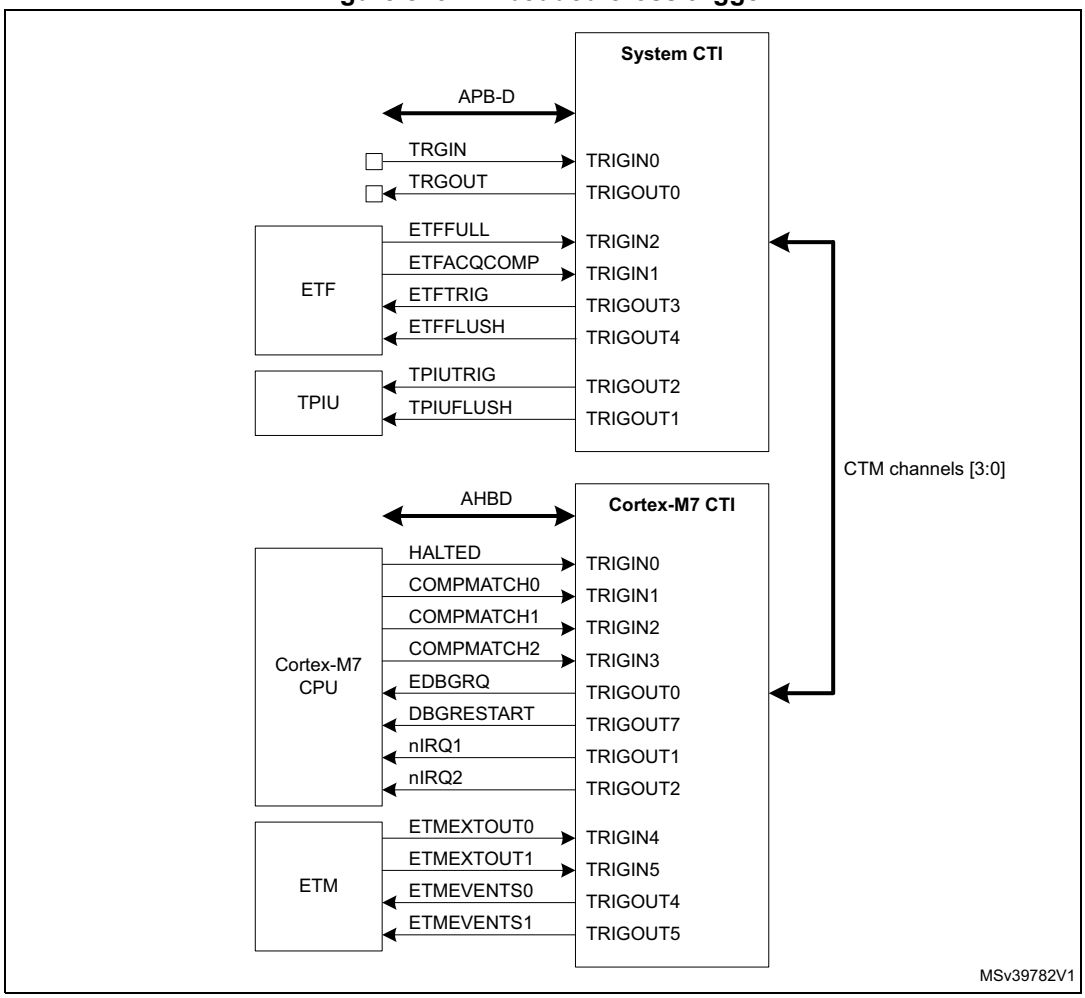
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TSG_CNTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDBG	EN
	Reset value																															0	0
0x004	TSG_CNTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
	Reset value																															0	
0x008	TSG_CNTCVL	CNTCVL_L_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TSG_CNTCVU	CNTCVU_U_32[3:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TSG_CNTFID0	FREQ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024 to 0xFCC	Reserved	Reserved																															
0xFD0	TSG_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	TSG_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	1	0	0	0	0	1
0xFE4	TSG_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	TSG_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	TSG_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TSG_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TSG_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TSG_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TSG_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.5.3 Cross trigger interfaces (CTI) and matrix (CTM)

The cross trigger interfaces (CTI) and cross trigger matrix (CTM) together form the CoreSight embedded cross trigger feature. There are two CTI components, one at system

level and one dedicated to the Cortex-M7. The two CTIs are connected to each other via the CTM. The system-level CTI is accessible to the debugger via the system access port and associated APB-D. The Cortex-M7 CTI is physically integrated in the Cortex-M7 core, and is accessible via the Cortex-M7 access port and associated AHBD.

Figure 823. Embedded cross trigger



The CTIs allow events from various sources to trigger debug and/or trace activity. For example, a transition detected on an external trigger input can start code trace.

Each CTI has up to 8 trigger inputs and 8 trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in *Table 578* to *Table 581*.

Table 578. System CTI inputs

#	Source signal	Source component	Comments
0	DBTRIGI	GPIO	External trigger input - allows an external signal to generate a debug event
1	ETFACQCOMP	ETF	ETF capture finished - allows a debug event to be generated when the trace FIFO is empty

Table 578. System CTI inputs (continued)

#	Source signal	Source component	Comments
2	ETFFULL	ETF	ETF full flag - allows a debug event to be generated when the trace FIFO is full
3	-	-	Not used
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

Table 579. System CTI outputs

#	Output signal	Destination component	Comments
0	DBTRIGO	GPIO	External IO trigger output - allows monitoring of events on the external DBTRIGO pin
1	TPIUFLUSH	TPIU	Trace port flush trigger - causes the TPIU FIFO to be flushed
2	TPIUTRIG	TPIU	Trace Port enable trigger - starts trace output on the external trace port
3	ETFTRIG	ETF	ETF enable trigger - starts filling the Trace FIFO
4	ETFFLUSH	ETF	ETF flush trigger - causes the Trace FIFO to be flushed
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

Table 580. Cortex-M7 CTI inputs

#	Source signal	Source component	Comments
0	HALTED	Cortex-M7 CPU	CPU halted - indicates CPU is in debug mode
1	COMPMATCH0	Cortex-M7 DWT	DWT comparator 0 match
2	COMPMATCH1	Cortex-M7 DWT	DWT comparator 1 match
3	COMPMATCH2	Cortex-M7 DWT	DWT comparator 2 match
4	ETMEXTOUT0	Cortex-M7 ETM	ETM external trigger out
5	ETMEXTOUT1	Cortex-M7 ETM	ETM external trigger out
6	-	-	Not used
7	-	-	Not used

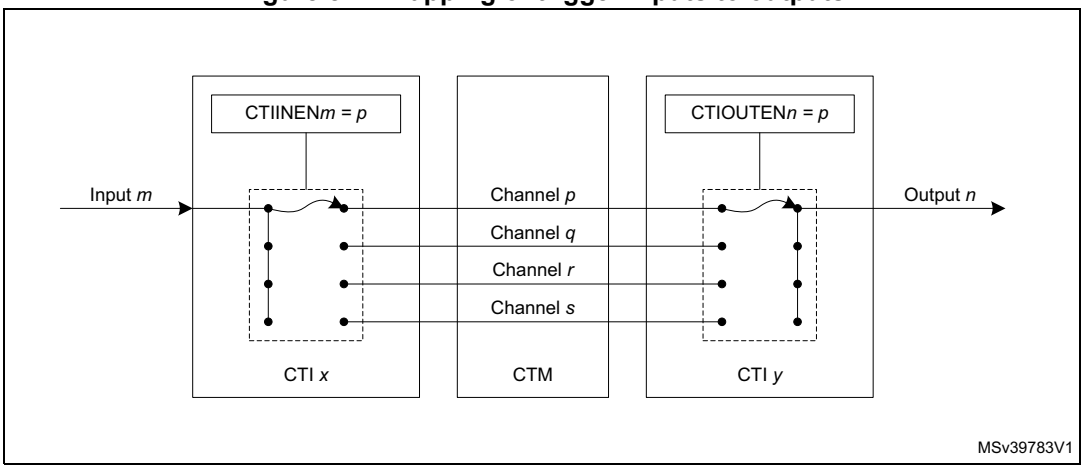
Table 581. Cortex-M7 CTI outputs

#	Output signal	Destination component	Comments
0	EDBGRQ	Cortex-M7 CPU	CPU halt request - puts CPU in debug mode
1	nIRQ1	Cortex-M7 NVIC	Interrupt request
2	nIRQ2	Cortex-M7 NVIC	Interrupt request
3	-	-	Not used
4	ETMEVENTS0	Cortex-M7 ETM	ETM trig request - enables CPU execution trace
5	ETMEVENTS1	Cortex-M7 ETM	ETM trig request - enables CPU execution trace
6	-	-	Not used
7	DBGRESTART	Cortex-M7 CPU	CPU restart request - CPU exits debug mode

There are four event channels in the cross trigger matrix, which allows up to four parallel bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number *m* on CTI *x* to output number *n* on CTI *y*, the input must be connected to an event channel *p* using the CTIINEN*m* register of CTI *x*. The same channel *p* must be connected to the output using the CTIOUTEN*n* register of CTI *y*. Note: this applies even if the input and output belong to the same CTI.

An input can be connected to more than one channel (up to four), so an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

Figure 824. Mapping of trigger inputs to outputs



For more information on the cross-trigger interface CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

CTI registers

The register file base address for each CTI is defined by the ROM table for the bus to which it is connected. The registers are the same for each CTI.

CTI control register (CTI_CONTROL)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: Global enable.

0: Cross-triggering disabled

1: Cross-triggering enabled

CTI trigger acknowledge register (CTI_INTACK)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: Trigger acknowledge

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSET)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: Set channel event

Read:
0bXXX0: Channel 0 event inactive
0bXXX0: Channel 0 event active
0bXX0X: Channel 1 event inactive
0bXX1X: Channel 1 event active
0bX0XX: Channel 2 event inactive
0bX1XX: Channel 2 event active
0b0XXX: Channel 3 event inactive
0b1XXX: Channel 3 event active

Write:
0bXXX0: No effect
0bXXX0: Set event on Channel 0
0bXX0X: No effect
0bXX1X: Set event on Channel 1
0bX0XX: No effect
0bX1XX: Set event on Channel 2
0b0XXX: No effect
0b1XXX: Set event on Channel 3

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: Clear channel event
0bXXX0: No effect
0bXXX0: Clear event on Channel 0
0bXX0X: No effect
0bXX1X: Clear event on Channel 1
0bX0XX: No effect
0bX1XX: Clear event on Channel 2
0b0XXX: No effect
0b1XXX: Clear event on Channel 3

CTI application pulse register (CTI_APPPULSE)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: Pulse channel event
This register clears itself immediately.

0bXXX0: No effect
0bXXX0: Generate pulse on Channel 0
0bXX0X: No effect
0bXX1X: Generate pulse on Channel 1
0bX0XX: No effect
0bX1XX: Generate pulse on Channel 2
0b0XXX: No effect
0b1XXX: Generate pulse on Channel 3

CTI trigger IN x enable register (CTI_INENx)

Address offset: 0x020 + 4 * x, where x = 0 to 7

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: Cross-trigger event enable

Enables or disables a cross-trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0bXXX0: Trigger n does not generate events on Channel 0
0bXXX0: Trigger n generates events on Channel 0
0bXX0X: Trigger n does not generate events on Channel 1
0bXX1X: Trigger n generates events on Channel 1
0bX0XX: Trigger n does not generate events on Channel 2
0bX1XX: Trigger n generates events on Channel 2
0b0XXX: Trigger n does not generate events on Channel 3
0b1XXX: Trigger n generates events on Channel 3

CTI trigger OUT x enable register (CTI_OUTENx)

Address offset: 0x0A0 + 4 * x, where x = 0 to 7

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: Enable trigger upon event

For each channel, the field defines whether an event on that channel will generate a trigger on CTITRIGOUTx (x = 0 to 7).

0bXXX0: Channel 0 events do not generate triggers on Trigger output n
0bXXX0: Channel 0 events generate triggers on Trigger output n
0bXX0X: Channel 1 events do not generate triggers on Trigger output n
0bXX1X: Channel 1 events generate triggers on Trigger output n
0bX0XX: Channel 2 events do not generate triggers on Trigger output n
0bX1XX: Channel 2 events generate triggers on Trigger output n
0b0XXX: Channel 3 events do not generate triggers on Trigger output n
0b1XXX: Channel 3 events generate triggers on Trigger output n

CTI trigger IN status register (CTI_TRGISTS)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: Trigger input status

There is one bit of the register for each CTITRIGIN input. When a bit is set to 1 it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger OUT status register (CTI_TRGOSTS)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1 it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel IN status register (CTI_CHINSTS)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: Channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel OUT status register (CTI_CHOUTSTS)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: Channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATE)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: Channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

0bXXX0: Channel 0 events do not propagate
0bXXX0: Channel 0 events propagate
0bXX0X: Channel 1 events do not propagate
0bXX1X: Channel 1 events propagate
0bX0XX: Channel 2 events do not propagate
0bX1XX: Channel 2 events propagate
0b0XXX: Channel 3 events do not propagate
0b1XXX: Channel 3 events propagate

CTI claim tag set register (CTI_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

CTI claim tag clear register (CTI_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

CTI lock access register (CTI_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CTI register write access enable
Enables write access to some CTI registers by processor core (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access
Other values: Disable write access

CTI lock status register (CTI_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the CTI_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.

0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists
1: Lock control mechanism is implemented

CTI authentication status register (CTI_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

CTI device configuration register (CTI_DEVID)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available
0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available
0x8: 8 trigger inputs and 8 trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers
0x0: None

CTI device type identifier register (CTI_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Indicates that this component is a cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x06: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CTI CoreSight component identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

CTI register map and reset values

Table 582. CTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x010	CTI_INTACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	CTI_APPSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
	Reset value																													0	0	0	0
0x01C	CTI_APPPULSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
	Reset value																													0	0	0	0
0x020 to 0x03C	CTI_INEN0 to CTI_INEN7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x040 to 0x09C	Reserved	Reserved																															
0x0A0 to 0x0BC	CTI_OUTEN0 to CTI_OUTEN7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0C0 to 0x12C	Reserved	Reserved																															
0x130	CTI_TRIGISTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRIGOSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	CTI_CHINSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHISTATUS[3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOSTATUS[3:0]			
	Reset value																													0	0	0	0
0x140	CTI_GATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144 to 0xF9C	Reserved	Reserved																															
0xFA0	CTI_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	CTI_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFAC	Reserved	Reserved																															

Table 582. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFB0	CTI_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	CTI_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	CTI_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	1	0	1	0
0xFC8	CTI_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXMUXNUM [4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	CTI_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	1	1	0
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	1	0	0	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.5.4 Trace funnel (CSTF)

The trace funnel is a CoreSight component that combines the ATB buses from two trace sources into one single ATB. The CSTF has two ATB slave ports, and one ATB master port. An arbiter selects the slave ports according to a programmable priority.

The slave ports are connected as follows:

- S0: Cortex-M7 ETM
- S1: Cortex-M7 ITM

The CSTF registers allow the slave ports to be individually enabled, and their priority settings to be configured. The priorities can be modified only when trace is disabled. The arbitration works as follows:

- The arbiter selects the slave port with the highest assigned priority that has data valid
- Up to *min_hold_time* transfers are passed from the selected slave to the master port, where *min_hold_time* is programmable in the CONTROL register.
- A new arbitration is then performed

High priority should be assigned to slave ports connected to sources with a small amount of buffering, or where data loss can not be tolerated. Low priority should be assigned to less critical sources or those with large buffers.

For more information on the ATB Funnel CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

Trace funnel registers

CSTF control register (CSTF_CTRL)

Address offset: 0x000

Reset value: 0x0000 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	ENS1	ENS0
				rw	rw	rw	rw							rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **MIN_HOLD_TIME[3:0]**: Number of transactions between arbitrations.
0x0: 1 transaction
:
0xE: 15 transactions
0xF: Reserved

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ENS1**: Slave port S1 enable
0: Disable port
1: Enable port

Bit 0 **ENS0**: Slave port S0 enable
0: Disable port
1: Enable port

CSTF priority register (CSTF_PRIORITY)

Address offset: 0x004

Reset value: 0x0000 0688

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT1[2:0]			PRIPORT0[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 **PRIPORT1[2:0]**: Slave port S1 priority
0: Highest priority
:
7: Lowest priority

Bits 2:0 **PRIPORT0[2:0]**: Slave port S0 priority
0: Highest priority
:
7: Lowest priority

CSTF claim tag set register (CSTF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

CSTF claim tag clear register (CSTF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

CSTF lock access register (CSTF_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CSTF register write access enable
The field enables write access to some CSTF registers by processor cores (debuggers do not need to unlock the component).

0xC5ACCE55: Enable write access
Other values: Disable write access

CSTF lock status register (CSTF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the CSTF_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.

0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists
1: Lock control mechanism is implemented

CSTF authentication status register (CSTF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

CSTF CoreSight device identity register (CSTF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SCHEME[3:0]**: Priority scheme
0x2: Static priority

Bits 3:0 **PORTCNT[3:0]**: Number of input ports connected
0x4: Four input ports

CSTF CoreSight device type identity register (CSTF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DEVTYPEID[7:0]**: Device type identifier
0x12: Trace funnel

CSTF CoreSight peripheral identity register 4 (CSTF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CSTF CoreSight peripheral identity register 0 (CSTF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x08: CSTF part number

CSTF CoreSight peripheral identity register 1 (CSTF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: CSTF part number

CSTF CoreSight peripheral identity register 2 (CSTF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

CSTF CoreSight peripheral identity register 3 (CSTF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CSTF CoreSight component identity register 0 (CSTF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CSTF CoreSight component identity register 1 (CSTF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CSTF CoreSight component identity register 2 (CSTF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CSTF CoreSight component identity register 3 (CSTF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Trace funnel register map and reset values

Table 583. CSTF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CSTF_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	ENS1	ENS0
	Reset value																					0	0	1	1							0	0
0x004	CSTF_PRIORITY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT1[2:0]			PRIPORT0[2:0]		
	Reset value																											0	0	1	0	0	0
0x008 to 0xF9C	Reserved	Reserved																															
0xFA0	CSTF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	CSTF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFA8to 0xFAC	Reserved	Reserved																															
0xFB0	CSTF_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	CSTF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	CSTF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	1	0	1	0
0xFC8	CSTF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
	Reset value																									0	0	1	0	0	1	0	0
0xFCC	CSTF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
	Reset value																									0	0	0	1	0	0	1	0

Table 583. CSTF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	CSTF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	CSTF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	1	0	0	0
0xFE4	CSTF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	CSTF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	CSTF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CSTF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CSTF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CSTF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CSTF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.5.5 Embedded trace FIFO (ETF)

The ETF is an 8 Kbyte memory that captures trace data from two trace sources, namely the ETM and ITM of the CPU core. The ETF is a design configuration of the CoreSight™ trace memory controller component.

The ETF can be used in three modes (selected in the mode register):

1. Hardware FIFO mode

   The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled. When the Trace buffer is not empty, trace data is drained out through the ATB master interface to the TPIU.

   In this mode, the role of the FIFO is to smooth the flow of trace information arriving at the trace port. Since the trace data can be very bursty in nature, the peak data rate can easily exceed the port capability, resulting in an overflow. The ETF allows a steady data rate at the trace port, which can then be sized according to the average rate rather than

the peak. The trace is stored off-chip in real time by the trace port analyzer tool, and so the trace log can be very big.

2. Software FIFO mode

   The trace memory is used as a FIFO that can be read through the RRD Register while trace is being captured. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled.

   This mode allows the trace to be transferred by DMA into the system memory, or to a high speed interface (SPI, USB etc), or even monitored by software. Note that unlike the hardware FIFO mode, this mode is invasive, since it uses system resources which are shared by the processor.

3. Circular buffer mode

   The trace memory is used as a circular buffer. Trace data is captured into the Trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition occurs.

   In this mode, the ETF stores the trace data on-chip, so the trace log size is limited to that of the ETF SRAM, 8 Kbytes in this case. Being a circular buffer, if the FIFO becomes full, incoming trace data overwrites the oldest stored data and the oldest stored data is lost. Therefore the contents of the trace buffer represent the most recent activity of the processor, up to the point when the buffer was stopped, rather than all the activity since the trace was started.

   There are three possible methods to read out the buffer contents once the trace stops:

   – via the Trace port - with the TPIU enabled, the contents of the buffer are output over the Trace port. This can be done by setting the DRAINBUF bit in the ETF_FFCR register.
   – via the Debug port - the debugger can read the buffer via the RRD register that is accessible over the system APB-D.
   – by software - the processor can read the buffer via the RRD register, since the APB-D is accessible from the system bus.

The ETF can transition between the following states:

- **Disabled**

  This state is entered after a reset, or when trace capture is disabled. The ETF must only be programmed in this state.

- **Running**

  Trace capture is performed in this state. It is entered by enabling trace capture while in Disabled state.

- **Stopped**

  Trace capture is stopped in this state, but the contents of the buffer can be read out or drained. This state is entered after a stop event (trigger or flush).

- **Disabling**

  This is a transitional state while disabling trace capture.

- **Stopping**

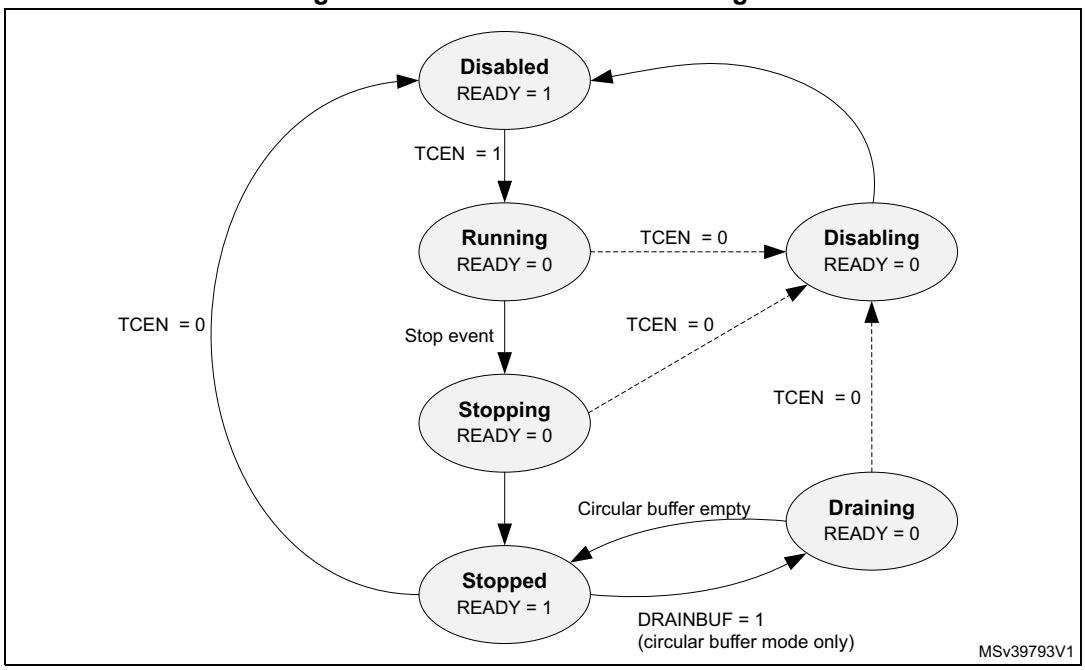
  This is a transitional state while stopping trace capture.

- **Draining**

  This state is entered while draining the buffer in Stopped state.

The state transition diagram is shown in *Figure 825*.

Figure 825. ETF state transition diagram



For more information on the trace memory controller CoreSight™ component, refer to the Arm® CoreSight™ trace memory controller technical reference manual *[3]*.

ETF registers

ETF RAM size register (ETF_RSZ)

Address offset: 0x004

Reset value: 0x0000 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RSZ[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSZ[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **RSZ[30:0]**: RAM size

The value of the field indicates the number of 32-bit words

0x800: 2048 words = 8 Kbyte

ETF status register (ETF_STS)

Address offset: 0x00C

Reset value: 0x0000 001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EMPTY**: Trace FIFO empty

This bit is valid only when the TCEN bit of the ETF_CTL register is high. This bit reads as zero when TCEN is low.

0: Trace FIFO contains data

1: Trace FIFO is empty.

Note: Empty trace FIFO does not mean that the ETF pipeline is empty. The latter is indicated by the FTEMPTY bit.

Bit 3 **FTEMPTY**: Formatter empty

This bit is set when trace capture has stopped, and all internal pipelines and buffers have drained. Unlike READY, it is not affected by buffer drains. The ACQCOMP output reflects the value of this bit.

Bit 2 **READY**: ETF ready

This bit is set when trace capture has stopped and all internal pipelines and buffers have drained (Stopped or Disabled state)

Bit 1 **TRIGD**: Triggered

The Triggered bit is set when trace capture is in progress and the TMC has detected a Trigger Event. This bit is cleared when leaving Disabled state.

This bit is operational only in the Circular buffer mode. In all other modes, this bit is always low.

This bit does not indicate that a trigger has been embedded in the formatted output trace data from the TMC. Trigger indication on the output trace stream is determined by the programming of the Formatter and Flush Control Register, ETF_FFCR.

Bit 0 **FULL**: Trace buffer full

In circular buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until the TCEN bit of the ETF_CTL register is cleared and set.

In software and hardware FIFO modes, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in the ETF_BUFWM Register, that is, Fill level >= MEM_SIZE - BUFWM.

This bit is cleared when leaving Disabled state. The FULL output reflects the value of this register bit.

ETF RAM read data register (ETF_RRD)

Address offset: 0x010

Reset value: Unknown

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RRD[31:0]**: RAM Read Data.

Circular buffer mode:

When in Stopped state and the buffer is not empty, reading this register returns the next word of data from the trace buffer. When all of the trace buffer has been read, the Empty bit in the ETF_STS Register is set, and subsequent reads return 0xFFFFFFF. Reading this register when not in Stopped state returns 0xFFFFFFFF.

Software FIFO mode:

Reading this register returns data from the FIFO. If this register is read when the FIFO is empty, the data returned is 0xFFFFFFFF.

Hardware FIFO mode:

Reading this register returns 0xFFFFFFFF.

ETF RAM read pointer register (ETF_RRP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RRP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RRP[12:0]**: RAM Read Pointer

The RAM Read Pointer Register contains the value of the read pointer that is used to read entries from the trace memory over the APB interface via the ETF_RRD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been written. When the pointer reaches its maximum value, it wraps around.

This register can only be written in Disabled state. It can be read in Disabled state, in Stopped state in circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.

ETF RAM write pointer register (ETF_RWP)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RWP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RWP[12:0]**: RAM write pointer

The RAM write pointer register contains the value of the write pointer that is used to write entries into the trace memory over the APB interface via the ETF_RWD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been read. When the pointer reaches its maximum value, it wraps around.

This register can only be written in Disabled state. It can be read in Disabled state, in Stopped state in circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.

ETF trigger counter register (ETF_TRG)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **TRG[10:0]**: Trigger counter

In Circular buffer mode, specifies the number of 32-bit words to capture in the trace RAM following the detection of either a rising edge on the TRIGIN input or a trigger packet in the incoming trace stream, ATID =7'h7D. On capturing the specified number of data words, a trigger event occurs. The effect of a trigger event on the ETF behavior is controlled by the FFCR Register.

The number of 32-bit words written into the trace RAM following the trigger is the value stored in this register, plus one. This register is ignored when the ETF is in Software FIFO mode or Hardware FIFO mode. When the trigger counter starts counting, any additional triggers, either on TRIGIN or in the incoming trace stream, are ignored until the counter reaches zero. When the trigger counter has reached zero, it remains at zero until it is re-programmed with a write to this register.

This register is cleared when READY goes high, so that the state of the counter when trace capture has stopped does not affect a subsequent trace capture session. Writing to this register when not in Disabled state results in unpredictable behavior.

A read access to this register is permitted at any time when in Disabled state, or in Circular buffer mode. A read access returns the current value of the trigger counter.

ETF control register (ETF_CTL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TCEN**: Trace capture enable

When writing:

0: Disable trace capture (moves from Running, Stopping or Stopped state into Disabling or Disabled state)

1: Enable trace capture (moves from Disabled state into Running state)

When reading, this bit is low when in Disabling or Disabled states, and high otherwise.

ETF RAM write data register (ETF_RWD)

Address offset: 0x024

Reset value: Unknown

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **RWD[31:0]**: RAM write data

When in Disabled state, a write to this register stores data at the location pointed to by the RWP. Writes to this register when not in Disabled state are ignored. When a full memory width (64-bit) of data has been written, the data is written to memory and the RAM Write Pointer is incremented to the next memory word.

This register is used for test purposes.

ETF mode register (ETF_MODE)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **MODE[1:0]**: Operation mode

00b: Circular buffer mode

In this mode, the trace memory is used as a circular buffer. Trace data is captured into the Trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition has occurred.

01b: Software FIFO mode

In this mode, the trace memory is used as a FIFO that can be read through the RRD Register while trace is being captured. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled.

10b: Hardware FIFO mode

In this mode, the trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled. When the Trace buffer is non-empty, trace data is drained out through the ATB master interface to the TPIU.

11b: Reserved

ETF latched buffer fill level register (ETF_LBUFLVL)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LBUFLEVEL[11:0]**: Latched buffer fill level

Reading this register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level.

When entering Disabled state, this register retains its last value. While in Disabled state, reads from this register do not affect its value. When exiting Disabled state, the LBUFLEVEL Register is cleared.

This register is used for performance analysis of the trace system.

ETF current buffer fill level register (ETF_CBUFLVL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CBUFLEVEL[11:0]**: Current buffer fill level

Reading this register returns the current fill level of the trace memory in 32-bit words.

This register is cleared when TCEN is low.

ETF buffer level watermark register (ETF_BUFWM)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res	Res	BUFWM[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **BUFWM[10:0]**: Buffer level watermark

The value programmed into this register indicates the required threshold vacancy level in 32-bit words in the trace memory. When the space in the FIFO is less than or equal to this value, that is, Fill level >= MEM_SIZE - BUFWM, the FULL output is pulled high and the FULL bit in the STS Register is set.

This register is used only in Software FIFO and Hardware FIFO modes. In Circular buffer mode, this functionality can be obtained by programming the RWP to the required vacancy trigger level, so that when the pointer wraps around, the FULL bit is set indicating that the vacancy level has fallen below the required level.

The maximum value that can be written into this register is MEM_SIZE - 1. In this case, the FULL bit output is asserted after the first 32-bit word is written to trace memory.

Writing to this register other than when in disabled state results in unpredictable behavior.

ETF formatter and flush status register (ETF_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FTSTOPPED**: Formatter stopped

This bit behaves in the same way as the FTEMPTY bit in the ETF_STS register.

Bit 0 **FLINPROG**: Flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the ETF_FFCR register, or requested by the ATB master port.

0: No flush in progress
1: Flush in progress

ETF formatter and flush control register (ETF_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRIGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	rw	rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **DRAINBUF**: Drain buffer

This bit is used to enable draining of the trace data through the ATB master interface after the formatter has stopped. This is useful in Circular buffer mode to capture trace data into trace memory and then to drain the captured trace through the ATB master interface.
Writing a one to this bit when in Stopped state starts the drain of the contents of the trace buffer through the ATB Master interface. This bit always reads as zero. The READY bit in the ETF_STS register goes low while the drain is in progress.
This bit is functional only when the ETF is in Circular buffer mode and formatting is enabled, that is, the ENFT bit in the ETF_FFCR register is set. Setting this bit when the ETF is in any other mode, or when not in Stopped state, results in Unpredictable behavior.
When trace capture is complete in Circular buffer mode, all of the captured trace must be retrieved from the trace memory through the same mechanism, either read all trace data out through RRD reads, or drain all trace data by setting the DRAINBUF bit. Setting the DRAINBUF bit after some of the captured trace has been read out through RRD results in unpredictable behavior.

Bit 13 **STPONTRGEV**: Stop on trigger event

0: No effect
1: Stop trace capture when a trigger event occurs

Enabling the ETF in Software FIFO mode or Hardware FIFO mode with this bit set results in unpredictable behavior.

Bit 12 **STOPONFL**: Stop on flush
0: No effect
1: Stop trace capture when flush is completed

If a flush is initiated by the ATB master interface, its completion does not lead to a formatter stop regardless of the value programmed in this bit.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGONFL**: Trigger on flush
0: No effect
1: Indicate a trigger in the trace stream when flush is completed

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.
If a flush is initiated by the ATB master interface, its completion does not lead to a trigger indication regardless of the value programmed in this bit.

Bit 9 **TRGONTRGEV**: Trigger on trigger event
0: No effect
1: Indicate a trigger in the trace stream when trigger event occurs

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.
This bit is not supported in Software FIFO mode or Hardware FIFO mode.

Bit 8 **TRGONTRGIN**: Trigger on trigger in
0: No effect
1: Indicate a trigger in the trace stream when a rising edge is detected on the TRIGIN input.

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FLUSHMAN**: Manual flush
0: No effect
1: Flush the trace FIFO and pipeline

This bit is cleared automatically when the flush completes. If the TCEN bit in the ETF_CTL register is 0, writes to this bit are ignored.

Bit 5 **FONTRGEV**: Flush on trigger event
0: No effect
1: Flush the trace FIFO and pipeline if a trigger event occurs

This bit is not supported in Software FIFO mode or Hardware FIFO mode. If STPONTRGEV is set, this bit is ignored.

Bit 4 **FONFLIN**: Flush on flush in
0: No effect
1: Flush the trace FIFO and pipeline if when a rising edge is detected on the FLUSHIN input

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENTI**: Enable trigger insertion

Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 8'h00 with ATID 7'h7D in the trace stream. Trigger indication on the trace stream is additionally controlled by the register bits TRIGONFL, TRGONTRGEV, and TRGONTRGIN in the FFCR Register. This bit can only be changed when READY is high, and TCEN is low. This bit takes effect only when the ENFT register bit in this register is set. If ENTI bit is set to high when ENFT is low, it results in formatting being enabled.

Bit 0 **ENFT**: Enable formatting.

0: Formatting is disabled. Incoming trace data is assumed to be from a single trace source.
1: Formatting is enabled.

If multiple ATIDs are received by the ETF when trace capture is enabled and the formatter is disabled, it results in interleaving of trace data. Disabling of formatting is supported only in Circular buffer mode. If the ETF is enabled in a mode other than Circular buffer mode with ENFT low, it results in formatting being enabled. If ENTI bit is set to high when ENFT is low, it results in formatting being enabled.
This bit is ignored when in Disabled state.

ETF periodic synchronization counter register (ETF_PSCR)

Address offset: 0x308

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PSCOUNT[4:0]**: Synchronization counter reload value

Determines the reload value of the Synchronization Counter. The reload value takes effect the next time the counter reaches zero. Reads from this register return the reload value programmed into this register. This register is set to 0xA on reset, corresponding to a synchronization period of 1024 bytes.

0x0: Synchronization disabled
0x1-0x6: Reserved
0x7-0x1B: Synchronization period is $2^{PSCOUNT}$ bytes
0x1C-0x1F: Reserved

ETF claim tag set register (ETF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

ETF claim tag clear register (ETF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

ETF lock access register (ETF_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETF register access enable

Enables write access to some ETF registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access
Other values: Disable write access

ETF lock status register (ETF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK TYPE	LOCK GRANT	LOCK EXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the ETF_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock

This bit always returns zero when read by an external debugger.

0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism

The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists
1: Lock control mechanism is implemented

ETF authentication status register (ETF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

ETF device configuration register (ETF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 01C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	MEMWIDTH[2:0]			CONFIGTYP[1:0]		CLK SCHEM	ATBINPORTCNT[4:0]				
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **MEMWIDTH[2:0]**: Memory interface data bus width
0x3: 64 bits (corresponds to 32-bit ATB data)

Bits 7:6 **CONFIGTYP[1:0]**: Configuration type of component (ETB, ETR or ETF)
0x2: ETF

Bit 5 **CLKSCHEM**: RAM clocking scheme (synchronous or asynchronous)
0: Synchronous

Bits 4:0 **ATBINPORTCNT**[4:0]: Number/type of ATB input port multiplexing
0x0: None

ETF device type identifier register (ETF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification

0x3: Captures trace data from the ATB slave interface into RAM that can be drained through the ATB master interface

Bits 3:0 **MAJORTYPE[3:0]**: Major classification

0x2: Component is a trace link because it has an ATB master interface through which trace data can be drained out in Hardware FIFO mode.

ETF CoreSight peripheral identity register 4 (ETF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size

0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x4: Arm® JEDEC code

ETF CoreSight peripheral identity register 0 (ETF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x61: ETF part number

ETF CoreSight peripheral identity register 1 (ETF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETF part number

ETF CoreSight peripheral identity register 2 (ETF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ETF CoreSight peripheral identity register 3 (ETF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETF CoreSight component identity register 0 (ETF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETF CoreSight component identity register 1 (ETF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETF CoreSight component identity register 2 (ETF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETF CoreSight component identity register 3 (ETF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

ETF register map and reset values

Table 584. ETF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETF_RSZ	Res.	RSZ[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0x008	Reserved	Reserved																															
0x00C	ETF_STS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
	Reset value																												1	1	1	0	0
0x010	ETF_RRD	RRD[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x014	ETF_RRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRP[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ETF_RWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWP[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ETF_TRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x020	ETF_CTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
	Reset value																																0
0x024	ETF_RWD	RWD[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x028	ETF_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
	Reset value																															0	0
0x02C	ETF_LBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x030	ETF_CBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x034	ETF_BUFWM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUFWM[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x2FC	Reserved	Reserved																															
0x300	ETF_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
	Reset value																															1	0

Table 584. ETF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x304	ETF_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	Reset value																		0	0	0		0	0	0		0	0	0			0	0
0x308	ETF_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
	Reset value																												0	1	0	1	0
0x30C to 0xF9C	Reserved	Reserved																															
0xFA0	ETF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	ETF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFAC	Reserved	Reserved																															
0xFB0	ETF_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	ETF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	ETF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFBC to 0xFC4	Reserved	Reserved																															
0xFC8	ETF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMWDTH[2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT[4:0]				
	Reset value																						0	0	1	1	1	0	0	0	0	0	0
0xFCC	ETF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value																									0	0	1	1	0	0	1	0

Table 584. ETF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	ETF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	ETF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	1	0	0	0	0	1
0xFE4	ETF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	ETF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	1	1	1
0xFEC	ETF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.5.6 Trace port interface unit (TPIU)

The TPIU is a CoreSight™ component that formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data. The trace port is a synchronous parallel port, comprising a clock output, TRACECK, and four data outputs, TRACED(7:0). The trace port width is programmable in the range 1 to 8. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes. However it restricts the bandwidth of the trace port and hence the quantity of trace information that can be output in real time. The TRACECK output must be enabled by setting the TRACECLKEN bit in the DBGMCU_CR register before trace is sent to the TPIU. Furthermore, the TRACECK frequency can be programmed in the RCC.

For more information on the Trace port interface CoreSight™ component, refer to the Arm® CoreSight™ SoC-400 technical reference manual *[2]*.

TPIU registers

TPIU supported port size register (TPIU_SUPPSIZE)

Address offset: 0x000

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Indicates supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported.

0x0000 000F: Port sizes 1 to 4 supported

TPIU current port size register (TPIU_CURPSIZE)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PORTSIZE[31:0]**: Indicates current trace port size

Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register should only be modified when the formatter is stopped.

TPIU supported trigger modes register (TPIU_SUPTRGM)

Address offset: 0x100

Reset value: 0x0000 011F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRGD
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
							r				r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **TRGRUN**: Trigger running
0: Trigger has not occurred or counter is at 0
1: Trigger has occurred and counter is not at 0

Bit 16 **TRIGD**: Triggered
0: Trigger has not occurred
1: Trigger has occurred and counter has reached 0

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TCOUNT8**: 8-bit counter register
1: Implemented

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: Multiplying the trigger counter by 65536 support
1: Supported

Bit 3 **MULT256**: Multiplying the trigger counter by 256 support
1: Supported

Bit 2 **MULT16**: Multiplying the trigger counter by 16 support
1: Supported

Bit 1 **MULT4**: Multiplying the trigger counter by 4 support
1: Supported

Bit 0 **MULT2**: Multiplying the trigger counter by 2 support
1: Supported

TPIU trigger counter value register (TPIU_TRGCNT)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGCOUNT[7:0]**: Enable trigger delay indication
Enables delaying the indication of triggers to any external connected trace capture or storage devices. This counter is only eight bits wide and is intended to be used only with the counter multipliers in the Trigger multiplier register, 0x108. When a trigger is started, this value, in combination with the multiplier, is the number of words before the trigger is indicated. When the trigger counter reaches 0, the value written here is reloaded. Writing to this register causes the trigger counter value to reset but does not reset any values on the multiplier. Reading this register returns the preset value, not the current count.

TPIU trigger multiplier register (TPIU_TRGMULT)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: Multiply the trigger counter by 65536
0: Disabled
1: Enabled

Bit 3 **MULT256**: Multiply the trigger counter by 256
0: Disabled
1: Enabled

Bit 2 **MULT16**: Multiply the trigger counter by 16
0: Disabled
1: Enabled

Bit 1 **MULT4**: Multiply the trigger counter by 4
0: Disabled
1: Enabled

Bit 0 **MULT2**: Multiply the trigger counter by 2
0: Disabled
1: Enabled

TPIU supported test patterns/modes register (TPIU_SUPTPM)

Address offset: 0x200

Reset value: 0x0003 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: Support of continuous mode
1: Supported

Bit 16 **PTIMEEN**: Support of timed mode
1: Supported

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: Support of FF/00 pattern
Indicates whether the FF/00 pattern is supported as output over the trace port.
1: Supported

Bit 2 **PATA5**: Support of AA/55 pattern
Indicates whether the AA/55 pattern is supported as output over the trace port.
1: Supported

Bit 1 **PATW0**: Support of walking 0's pattern
Indicates whether the walking 0's pattern is supported as output over the trace port.
1: Supported

Bit 0 **PATW1**: Support of walking 1's pattern
Indicates whether the walking 1's pattern is supported as output over the trace port.
1: Supported

TPIU current test pattern/mode register (TPIU_CURTPM)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: Continuous mode enable
0: Disabled
1: Enabled

Bit 16 **PTIMEEN**: Timed mode enable
0: Disabled
1: Enabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: FF/00 pattern enable
Indicates whether the FF/00 pattern is enabled as output over the trace port
0: Disabled
1: Enabled

Bit 2 **PATA5**: AA/55 pattern is enable
Indicates whether the AA/55 pattern is enabled as output over the trace port
0: Disabled
1: Enabled

Bit 1 **PATW0**: Walking 0's pattern enable
Indicates whether the walking 0's pattern is enabled as output over the trace port
0: Disabled
1: Enabled

Bit 0 **PATW1**: Walking 1's pattern enable
Indicates whether the walking 1's pattern is enabled as output over the trace port
0: Disabled
1: Enabled

TPIU test pattern repeat counter register (TPIU_TPRCR)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATTCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PATTCOUNT[7:0]**: Number of TRACECLKIN cycles
The field provides a 8-bit counter value to indicate the number of TRACECLKIN cycles for which a pattern runs before it switches to the next pattern.

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TCPRESENT**: TRACECTL output pin availability

Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: Formatter stopped

The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.

0: Formatter has not stopped
1: Formatter has stopped

Bit 0 **FLINPROG**: Flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.

0: No flush in progress
1: Flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STOPTRIG	STOPFL	Res.	TRIGFL	TRIGEVT	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	EN FTC
		rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **STOPTRIG**: Stop on trigger event

0: No effect
1: Stop formatter when a trigger event occurs

Bit 12 **STOPFL**: Stop on flush

0: No effect
1: Stop formatter when flush is completed

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGFL**: Trigger on flush

0: No effect
1: Indicate a trigger in the trace stream when flush is completed

Bit 9 **TRIGEVT**: Trigger on trigger event

0: No effect
1: Indicate a trigger in the trace stream when trigger event occurs

Bit 8 **TRIGIN**: Trigger on trigger in

0: No effect

1: Indicate a trigger in the trace stream when the TRIGIN input from the system CTI is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: Generate a manual flush

0: No effect

1: Flush the trace

This bit is cleared automatically when the flush completes.

Bit 5 **FONTRIG**: Flush on trigger event

A trigger event occurs when the trigger counter reaches 0, or, if the trigger counter is 0, when the TRIGIN input from the system CTI is high.

0: No effect

1: Flush the trace if a trigger event occurs

Bit 4 **FONFLIN**: Flush on flush in

0: No effect

1: Flush the trace if the FLUSHIN input from the system CTI is asserted

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** Enable continuous formatting

0: Continuous formatting is disabled

1: Continuous formatting is enabled

Bit 0 **ENFTC**: Enable the embedding of triggers in formatted trace

0: Formatting is disabled

1: Formatting is enabled

TPIU formatter synchronization counter register (TPIU_FSCR)

Address offset: 0x400

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CYCCOUNT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CYCCOUNT[11:0]**: Enables effective use of TPAs

Enables effective use of different-sized TPAs without wasting large amounts of storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

TPIU lock access register (TPIU_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: TPIU register access enable

Enables write access to some TPIU registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access
Other values: Disable write access

TPIU lock status register (TPIU_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the TPIU_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.

0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists
1: Lock control mechanism is implemented

TPIU authentication status register (TPIU_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

TPIU device configuration register (TPIU_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWO UART NRZ	SWO MAN	TCLK DATA	FIFO SIZE[2:0]			CLK RELAT	MUXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Support of SWO UART or NRZ
Indicates whether serial wire output, UART or NRZ, is supported.

0: Not supported

Bit 10 **SWOMAN**: Support of SWO Manchester format
Indicates whether serial wire output, Manchester encoded format, is supported.

0: Not supported

Bit 9 **TCLKDATA**: Support of trace clock plus data
0: Not supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relation
Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)

1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x0: None

TPIU device type identifier register (TPIU_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x12: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

TPIU CoreSight component identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

TPIU register map and reset values

Table 585. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SUPPSIZE	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x004	TPIU_CURPSIZE	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008 to 0x0FC	Reserved	Reserved																															
0x100	TPIU_SUPTRGM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRIGD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value															0	0								1				1	1	1	1	1
0x104	TPIU_TRGCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x108	TPIU_TRGMULT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value																												0	0	0	0	0
0x10C to 0x1FC	Reserved	Reserved																															
0x200	TPIU_SUPTPM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value															1	1													1	1	1	1
0x204	TPIU_CURTPM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value															0	0													0	0	0	0

Table 585. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x208	TPIU_TPRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATTCOUNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x20C to 0x2FC	Reserved	Reserved																															
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																														0	1	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPTRIG	STOPFL	Res.	TRiGFL	TRiGEVt	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	ENFTC
	Reset value																			0	0		0	0	0		0	0	0			0	0
0x308	TPIU_FSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CYCCOUNT[11:0]											
	Reset value																					0	0	0	0	0	1	0	0	0	0	0	0
0x30C to 0xF9C	Reserved	Reserved																															
0xFA0	TPIU_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFB0	TPIU_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	TPIU_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	TPIU_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFBC to 0xFC4	Reserved	Reserved																															
0xFC8	TPIU_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
	Reset value																					0	0	0	0	1	0	1	0	0	0	0	0

Table 585. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	TPIU_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYP[3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	TPIU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	0	1	0
0xFE4	TPIU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	1	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.5.7 Serial wire output (SWO) and SWO trace funnel (SWTF)

The SWO is a CoreSight component that formats the trace stream from the processor ITM and outputs it on the single wire TRACESWO output. The SWO trace funnel (SWTF) must be programmed to enable the trace bus from the Cortex-M7 ITM before trace is enabled. The SWTF registers are listed in *Table 587*.

Compared to the TPIU, the SWO contains:

- no formatter
- no pattern generator
- an 8-bit ATB input
- no synchronous trace output, that is, no TRACEDATA or TRACECLK pins
- no support for flush, because this is not required
- no support for triggering

The SWO output supports Manchester encoded and UART NRZ formats.

For more information about the serial wire output CoreSight™ component, refer to the Arm® CoreSight™ Components Technical Reference Manual *[4]*.

SWO registers

SWO current output divisor register (SWO_CODR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: SWO baud rate scaling

The baud rate is the trace clock frequency divided by (PRESCALER - 1). The baud rate changes instantly, so it is recommended to stop the trace source and wait until the port is idle before writing to this register.

SWO selected pin protocol register (SWO_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							PPROT[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **PPROT[1:0]**: Pin protocol

0x0: Reserved
0x1: Manchester
0x2: NRZ
0x3: Reserved

SWO formatter and flush status register (SWO_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: Change of settings without stopping formatter
1: Change of settings is allowed with formatter running

Bit 2 **TCPRESENT**: TRACECTL pin present on SWO
0: TRACECTL pin not present

Bit 1 **FTSTOPPED**: Formatter stopped
0: Formatter running
The bit always returns 0 as the SWO formatter cannot be stopped in this device.

Bit 0 **FLINPROG**: Flush in progress
0: Flush is not in progress
The bit always returns 0 as SWO flushing is not supported in this device.

SWO claim tag set register (SWO_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

SWO claim tag clear register (SWO_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

SWO lock access register (SWO_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: SWO register write access enable
Enables write access to some SWO registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access
Other values: Disable write access

SWO lock status register (SWO_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the SWO_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.

0: Write access is permitted
1: Write access is blocked - only read access is permitted

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists
1: Lock control mechanism is implemented

SWO authentication status register (SWO_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

SWO device configuration register (SWO_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0EA0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWO UART NRZ	SWO MAN	TCLK DATA	FIFO SIZE[2:0]			CLK RELAT	MUXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: SWO UART or NRZ support
Indicates whether serial wire output, UART or NRZ, is supported.

1: Supported

Bit 10 **SWOMAN**: SWO Manchester format support
Indicates whether serial wire output, Manchester encoded format, is supported.
1: Supported

Bit 9 **TCLKDATA**: Trace clock plus data support
Indicates whether trace clock plus data is supported

1: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT**: ATB clock to TRACECLKIN relation
Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)

1: Asynchronous

Bits 4:0 **MUXNUM[4:0]**: Number/type of ATB input port multiplexing
0x0: None

SWO device type identifier register (SWO_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

SWO CoreSight peripheral identity register 4 (SWO_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

SWO CoreSight peripheral identity register 0 (SWO_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x14: SWO part number

SWO CoreSight peripheral identity register 1 (SWO_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: SWO part number

SWO CoreSight peripheral identity register 2 (SWO_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

SWO CoreSight peripheral identity register 3 (SWO_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SWO CoreSight component identity register 0 (SWO_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SWO CoreSight component identity register 1 (SWO_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SWO CoreSight component identity register 2 (SWO_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SWO CoreSight component identity register 3 (SWO_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

SWO register map and reset values

Table 586. SWO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SWO_CODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 to 0x0EC	Reserved	Reserved																															
0x0F0	SWO_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPROT[1:0]	
	Reset value																															0	1

Table 586. SWO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F4 to 0x2FC	Reserved	Reserved																															
0x300	SWO_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304 to 0xF9C	Reserved	Reserved																															
0xFA0	SWO_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3;0]			
	Reset value																													1	1	1	1
0xFA4	SWO_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3;0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFAC	Reserved	Reserved																															
0xFB0	SWO_LAR	ACCESS_W[31;0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	SWO_LSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	SWO_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1;0]		NSNID[1:0]		NSID[1;0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFBC to 0xFC4	Reserved	Reserved																															
0xFC8	SWO_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT		MUXNUM[4:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0
0xFCC	SWO_DEVTYPE	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0xFD0	SWO_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															

Table 586. SWO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	SWO_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	1	0	0
0xFE4	SWO_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	SWO_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	SWO_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SWO_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SWO_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	SWO_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SWO_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

SWTF registers

SWTF control register (SWTF_CTRL)

Address offset: 0x000

Reset value: 0x0000 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENS0
				rw	rw	rw	rw								rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **MIN_HOLD_TIME[3:0]**: Number of transactions between arbitrations.
0x0: 1 transaction
:
0xE: 15 transactions
0xF: Reserved

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **ENS0**: Slave port S0 enable
0: Disable port
1: Enable port

SWTF priority register (SWTF_PRIORITY)

Address offset: 0x004

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT0[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PRIPORT0[2:0]**: Slave port S0 priority
0: Highest priority
:
7: Lowest priority

SWTF claim tag set register (SWTF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

SWTF claim tag clear register (SWTF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

SWTF lock access register (SWTF_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: SWTF register write access enable
Enables write access to some SWTF registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access
Other values: Disable write access

SWTF lock status register (SWTF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the SWTF_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.

0: Write access is permitted
1: Write access is blocked - only read access is permitted

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists
1: Lock control mechanism is implemented

SWTF authentication status register (SWTF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

SWTF CoreSight device identity register (SWTF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SCHEME[3:0]**: Priority scheme
0x2: Static priority

Bits 3:0 **PORTCNT[3:0]**: Number of input ports connected
0x2: Two input ports

SWTF CoreSight device type identity register (SWTF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DEVTYPEID[7:0]**: Device type identifier
0x12: Trace funnel

SWTF CoreSight peripheral identity register 4 (SWTF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

SWTF CoreSight peripheral identity register 0 (SWTF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x08: SWTF part number

SWTF CoreSight peripheral identity register 1 (SWTF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: SWTF part number

SWTF CoreSight peripheral identity register 2 (SWTF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

SWTF CoreSight peripheral identity register 3 (SWTF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SWTF CoreSight component identity register 0 (SWTF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SWTF CoreSight component identity register 1 (SWTF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SWTF CoreSight component identity register 2 (SWTF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SWTF CoreSight component identity register 3 (SWTF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

SWTF register map and reset values

Table 587. SWTF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SWTF_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENS0
	Reset value																					0	0	1	1								0
0x004	SWTF_PRIORITY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT0[2:0]		
	Reset value																														0	0	0
0x008 to 0xF8C	Reserved	Reserved																															
0xFA0	SWTF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	SWTF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFAC	Reserved	Reserved																															
0xFB0	SWTF_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	SWTF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1

Table 587. SWTF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFB8	SWTF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFBC to 0xFC4	Reserved	Reserved																															
0xFC8	SWTF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
	Reset value																									0	0	1	0	0	0	1	0
0xFCC	SWTF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
	Reset value																									0	0	0	1	0	0	1	0
0xFD0	SWTF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	SWTF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	1	0	0	0
0xFE4	SWTF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	SWTF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	SWTF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SWTF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SWTF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	SWTF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SWTF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.5.8 Microcontroller debug unit (DBGMCU)

The DBGMCU component contains a number of registers that control the power and clock behavior in debug mode. Specifically it allows the debugger, or debug software, to:

- maintain the clock and power to the processor cores when in low-power modes (sleep, stop or standby)
- maintain the clock and power to the system debug and trace components when in low power modes
- stop the clock to certain peripherals (CAN, SMBUS timeout, Watchdogs, Timers, RTC) when either processor core is stopped in debug mode. For timers having complementary outputs, the outputs are disabled (as if the MOE bit was reset) for safety purposes when the counter is stopped (TIM1/8/15/16/17 = 1 in DBGMCU_APB2FZ1).

The DBGMCU registers are not reset by a system reset, only by a power on reset. They are accessible to the debugger via the APB-D bus at base address 0xE00E1000. They are also accessible by both processor cores at base address 0x5C001000

Note: the DBGMCU is not a standard CoreSight component. Therefore, it does not appear in the system ROM table.

DBGMCU registers

DBGMCU identity code register (DBGMCU_IDC)

Address offset: 0x000

Reset value: 0x1000 6480

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Revision
0x1001 = Revision Z
0x1003 = Revision Y
0x2001 = Revision X

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device ID
0x450: STM32H743/753 and STM32H750

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	Res.	Res.	D3DBGCKEN	D1DBGCKEN	TRACECLKEN	Res.	Res.	Res.	Res.
			rw						rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTBY_D3	DBGSTOP_D3	Res.	Res.	Res.	Res.	DBGSTBY_D1	DBGSTOP_D1	DBGSLEEP_D1
							rw	rw					rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TRGOEN**: External trigger output enable

This bit controls the direction of the bi-directional trigger pin, TRGIO.

0: Input - TRGIO is connected to TRGIN
1: Output - TRGIO is connected to TRGOUT

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **D3DBGCKEN**: D3 debug clock enable

This bit allows the debug components in the D3 domain (excluding the DAPCLK domain) to be switched off if they are not needed.

0: Disabled - D3 domain debug components are disabled and their clocks gated
1: Enabled - D3 domain debug components are clocked whenever the corresponding domain clock (CK_HCLK_D3) is active

Bit 21 **D1DBGCKEN**: D1 debug clock enable

This bit allows the debug components in the D1 domain (excluding those in the processor core) to be switched off if they are not needed.

0: Disabled - D1 domain debug components are disabled and their clocks gated
1: Enabled - D1 domain debug components are clocked whenever the corresponding domain clock (CK_HCLK_D1) is active

Bit 20 **TRACECLKEN**: Trace port clock enable

This bit enables the trace port clock, TRACECLK.

0: Disabled - TRACECLK is disabled
1: Enabled - TRACECLK is active

Bits 19:9 Reserved, must be kept at reset value.

Bit 8 **DBGSTBY_D3**: Allow debug in D3 Standby mode

0: Normal operation - all clocks are disabled and the domain powered down automatically in Standby mode(1).
1: Automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Standby mode, and the domain supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 7 **DBGSTOP_D3**: Allow debug in D3 Stop mode

0: Normal operation - domain clocks are disabled automatically in Stop mode(2)

1: Automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Stop mode. On exit from Stop mode, the clock settings is set to the Stop mode exit state.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **DBGSTBY_D1**: Allow D1 domain debug in Standby mode

0: Normal operation - all clocks is disabled and the domain powered down automatically in Standby mode.

1: Automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Standby mode, and the domain supply is maintained, allowing full debug capability. On exit from Standby mode, a domain reset is performed.

Bit 1 **DBGSTOP_D1**: Allow D1 domain debug in Stop mode

0: Normal operation - all clocks are disabled automatically in Stop mode

1: Automatic clock stop disabled - all active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings is set to the Stop mode exit state.

Bit 0 **DBGSLEEP_D1**: Allow D1 domain debug in Sleep mode

0: Normal operation - processor clock is stopped automatically in Sleep mode

1: Automatic clock stop disabled - processor clock continues to run, allowing full debug capability

1. If CDBGPWRUPREQ in the debug port Control/Stat register is asserted, D3 will remain powered up and the DAPCLK active in Standby mode, even if DBGSTBY_D3 is reset. However the remaining D3 domain clocks will be switched off.
2. If CDBGPWRUPREQ in the debug port Control/Stat register is asserted, D3 will remain powered up and the DAPCLK active in Stop mode, even if DBGSTOP_D3 is reset. However the remaining D3 domain clocks will be switched off.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZ1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1	Res.	Res.	Res.	Res.	Res.	Res.
									rw						

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **WWDG1**: WWDG1 stop in debug

0: Normal operation - WWDG1 continues to operate while the core is in debug mode

1: Stop in debug - WWDG1 is frozen while the core is in debug mode

Bits 5:0 Reserved, must be kept at reset value.

DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZ1)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3	I2C2	I2C1	Res.	Res.	Res.	Res.	Res.
								rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **I2C3**: I2C3 SMBUS timeout stop in debug

0: Normal operation - I2C3 SMBUS timeout continues to operate while the core is in debug mode

1: Stop in debug - I2C3 SMBUS timeout is frozen while Cortex-M7 is in debug mode

Bit 22 **I2C2**: I2C2 SMBUS timeout stop in debug

0: Normal operation - I2C2 SMBUS timeout continues to operate while the core is in debug mode

1: Stop in debug - I2C2 SMBUS timeout is frozen while Cortex-M7 is in debug mode

Bit 21 **I2C1**: I2C1 SMBUS timeout stop in debug

0: Normal operation - I2C1 SMBUS timeout continues to operate while the core is in debug mode

1: Stop in debug - I2C1 SMBUS timeout is frozen while the core is in debug mode

Bits 20:11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1**: LPTIM1 stop in debug

0: Normal operation - LPTIM1 continues to operate while the core is in debug mode

1: Stop in debug - LPTIM1 is frozen while Cortex-M7 is in debug mode

Bit 8 **TIM14**: TIM14 stop in debug

0: Normal operation - TIM14 continues to operate while the core is in debug mode

1: Stop in debug - TIM14 is frozen while Cortex-M7 is in debug mode

Bit 7 **TIM13**: TIM13 stop in debug

0: Normal operation - TIM13 continues to operate while the core is in debug mode

1: Stop in debug - TIM13 is frozen while Cortex-M7 is in debug mode

Bit 6 **TIM12**: TIM12 stop in debug

0: Normal operation - TIM12 continues to operate while the core is in debug mode

1: Stop in debug - TIM12 is frozen while Cortex-M7 is in debug mode

Bit 5 **TIM7**: TIM7 stop in debug

0: Normal operation - TIM7 continues to operate while the core is in debug mode

1: Stop in debug - TIM7 is frozen while Cortex-M7 is in debug mode

Bit 4 **TIM6**: TIM6 stop in debug

0: Normal operation - TIM6 continues to operate while the core is in debug mode

1: Stop in debug - TIM6 is frozen while Cortex-M7 is in debug mode

Bit 3 **TIM5**: TIM5 stop in debug

0: Normal operation - TIM5 continues to operate while the core is in debug mode
1: Stop in debug - TIM5 is frozen while Cortex-M7 is in debug mode

Bit 2 **TIM4**: TIM4 stop in debug

0: Normal operation - TIM4 continues to operate while the core is in debug mode
1: Stop in debug - TIM4 is frozen while Cortex-M7 is in debug mode

Bit 1 **TIM3**: TIM3 stop in debug

0: Normal operation - TIM3 continues to operate while the core is in debug mode
1: Stop in debug - TIM3 is frozen while Cortex-M7 is in debug mode

Bit 0 **TIM2**: TIM2 stop in debug

0: Normal operation - TIM2 continues to operate while the core is in debug mode
1: Stop in debug - TIM2 is frozen while Cortex-M7 is in debug mode

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZ1)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	HRTIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17	TIM16	TIM15
		rw											rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM8	TIM1
														rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **HRTIM**: HRTIM stop in debug

0: Normal operation - HRTIM continues to operate while the core is in debug mode
1: Stop in debug - HRTIM is frozen while Cortex-M7 is in debug mode

Bits 28:18 Reserved, must be kept at reset value.

Bit 18 **TIM17**: TIM17 stop in debug

0: Normal operation - TIM17 continues to operate while the core is in debug mode
1: Stop in debug - TIM17 is frozen and TIM17 outputs are disabled while Cortex-M7 is in debug mode

Bit 17 **TIM16**: TIM16 stop in debug

0: Normal operation - TIM16 continues to operate while the core is in debug mode
1: Stop in debug - TIM16 is frozen and TIM16 outputs are disabled while Cortex-M7 is in debug mode

Bit 16 **TIM15**: TIM15 stop in debug

0: Normal operation - TIM15 continues to operate while the core is in debug mode
1: Stop in debug - TIM15 is frozen and TIM15 outputs are disabled while Cortex-M7 is in debug mode

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **TIM8**: TIM8 stop in debug

0: Normal operation - TIM8 continues to operate while the core is in debug mode
1: Stop in debug - TIM8 is frozen and TIM8 outputs are disabled while Cortex-M7 is in debug mode

Bit 0 **TIM1**: TIM1 stop in debug

0: Normal operation - TIM1 continues to operate while the core is in debug mode
1: Stop in debug - TIM1 is frozen and TIM1 outputs are disabled while Cortex-M7 is in debug mode.

DBGMCU APB4 peripheral freeze register (DBGMCU_APB4FZ1)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGLSD1	Res.	RTC
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.	I2C4	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw		rw							

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **WDGLSD1**: LS watchdog for D1 stop in debug

0: Normal operation - watchdog continues to count while the core is in debug mode
1: Stop in debug - watchdog is frozen while Cortex-M7 is in debug mode

Bit 17 Reserved, must be kept at reset value.

Bit 16 **RTC**: RTC stop in debug

0: Normal operation - RTC continues to operate while the core is in debug mode
1: Stop in debug - RTC is frozen while Cortex-M7 is in debug mode

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5**: LPTIM5 stop in debug

0: Normal operation - LPTIM5 continues to operate while the core is in debug mode
1: Stop in debug - LPTIM5 is frozen while Cortex-M7 is in debug mode

Bit 11 **LPTIM4**: LPTIM4 stop in debug

0: Normal operation - LPTIM4 continues to operate while the core is in debug mode
1: Stop in debug - LPTIM4 is frozen while Cortex-M7 is in debug mode

Bit 10 **LPTIM3**: LPTIM2 stop in debug

0: Normal operation - LPTIM2 continues to operate while the core is in debug mode
1: Stop in debug - LPTIM2 is frozen while Cortex-M7 is in debug mode

Bit 9 **LPTIM2**: LPTIM2 stop in debug

0: Normal operation - LPTIM2 continues to operate while the core is in debug mode
1: Stop in debug - LPTIM2 is frozen while Cortex-M7 is in debug mode

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4**: I2C4 SMBUS timeout stop in debug

0: Normal operation - I2C4 SMBUS timeout continues to operate while the core is in debug mode

1: Stop in debug - I2C4 SMBUS timeout is frozen while the core is in debug mode

Bits 6:0 Reserved, must be kept at reset value.

DBGMCU register map and reset values

Table 588. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_IDC	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0					0	1	0	0	0	1	0	1	0	0	0	0
0x004	DBGMCU_CR	Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	Res.	Res.	D3DBGCKEN	D1DBGCKEN	TRACECLKEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTBD3	DBGSTPD3	Res.	DBGSTBD2	DBGSTPD2	DBGSLPD2	DBGSTBD1	DBGSTPD1	DBGSLPD1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008 to 0x030	Reserved	Reserved																															
0x034	DBGMCU_APB3FZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																										0						
0x038	Reserved	Reserved																															
0x03C	DBGMCU_APB1LFZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3	I2C2	I2C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
	Reset value									0	0	0												0	0	0	0	0	0	0	0	0	0
0x040 to 0x48	Reserved	Reserved																															
0x04C	DBGMCU_APB2FZ1	Res.	Res.	HRTIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17	TIM16	TIM15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM8	TIM1
	Reset value			0											0	0	0														0	0
0x050	Reserved	Reserved																															
0x054	DBGMCU_APB4FZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGLSD1	Res.	RTC	Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.	I2C4	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0		0				0	0	0	0		0							

59.6 Cortex-M7 debug functional description

The Cortex-M7 subsystem features the following CoreSight™ components:

- ROM tables
- System control space (SCS)
- Breakpoint unit (FPB)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace macrocell (ETM)
- Cross trigger interface (CTI)

These components are accessible by the debugger via the Cortex-M7 AHB-AP and its associated AHBD bus.

59.6.1 Cortex-M7 ROM tables

The ROM table is a CoreSight™ component that contains the base addresses of all the CoreSight debug components accessible via the AHBD. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There are two ROM tables in the Cortex-M7 sub-system:

- Cortex-M7 CPU ROM table

  This table is pointed to by the BASE register in the Cortex-M7 AHB-AP. It contains the base address pointers for the ETM and CTI, as well as for the Cortex-M7 CPU ROM table.
- Cortex-M7 PPB (private peripheral bus) ROM table

  This table contains pointers to the Cortex-M7 System Control Space registers allowing the debugger to identify the CPU core, as well as to the remaining CoreSight components in the Cortex-M7 subsystem: FPB, DWT and ITM.

The CPU ROM table occupies a 4-Kbyte, 32-bit wide chunk of AHBD address space, from 0xE00FE000 to 0xE00FEFFC.

Table 589. Cortex-M7 CPU ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00FE000	Cortex-M7 PPB ROM table	0xE00FF000	0x00001000	4 Kbyte	0x00001003
0xE00FE004	Cortex-M7 ETM	0xE0041000	0xFFF43000	4 Kbyte	0xFFF43003
0xE00FE008	Cortex-M7 CTI	0xE0043000	0xFFF44000	4 Kbyte	0xFFF44003
0xE00FE00C	Reserved	-	-	-	0x1FF02002
0xE00FE010	Top of table	-	-	-	0x00000000
0xE00FE010 to 0xE00FEFC8	Reserved	-	-	-	0x00000000
0xE00FEFCC to 0xE00FEFFC	ROM table registers	-	-	-	See *Table 591*

The Cortex-M7 PPB ROM table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00FF000 to 0xE00FFFFC.

Table 590. Cortex-M7 PPB ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00FF000	SCS	0xE000E000	0xFFF0F000	4 Kbyte	0xFFF0F003
0xE00FF004	DWT	0xE0001000	0xFFF02000	4 Kbyte	0xFFF02003
0xE00FF008	FPB	0xE0002000	0xFFF03000	4 Kbyte	0xFFF03003
0xE00FF00C	ITM	0xE0000000	0xFFF01000	4 Kbyte	0xFFF01003
0xE00FF010	TPIU(1)	0xE0040000	0xFFF41000	4 Kbyte	0xFFF41002
0xE00FF014	ETM(1)	0xE0041000	0xFFF42000	4 Kbyte	0xFFF42002

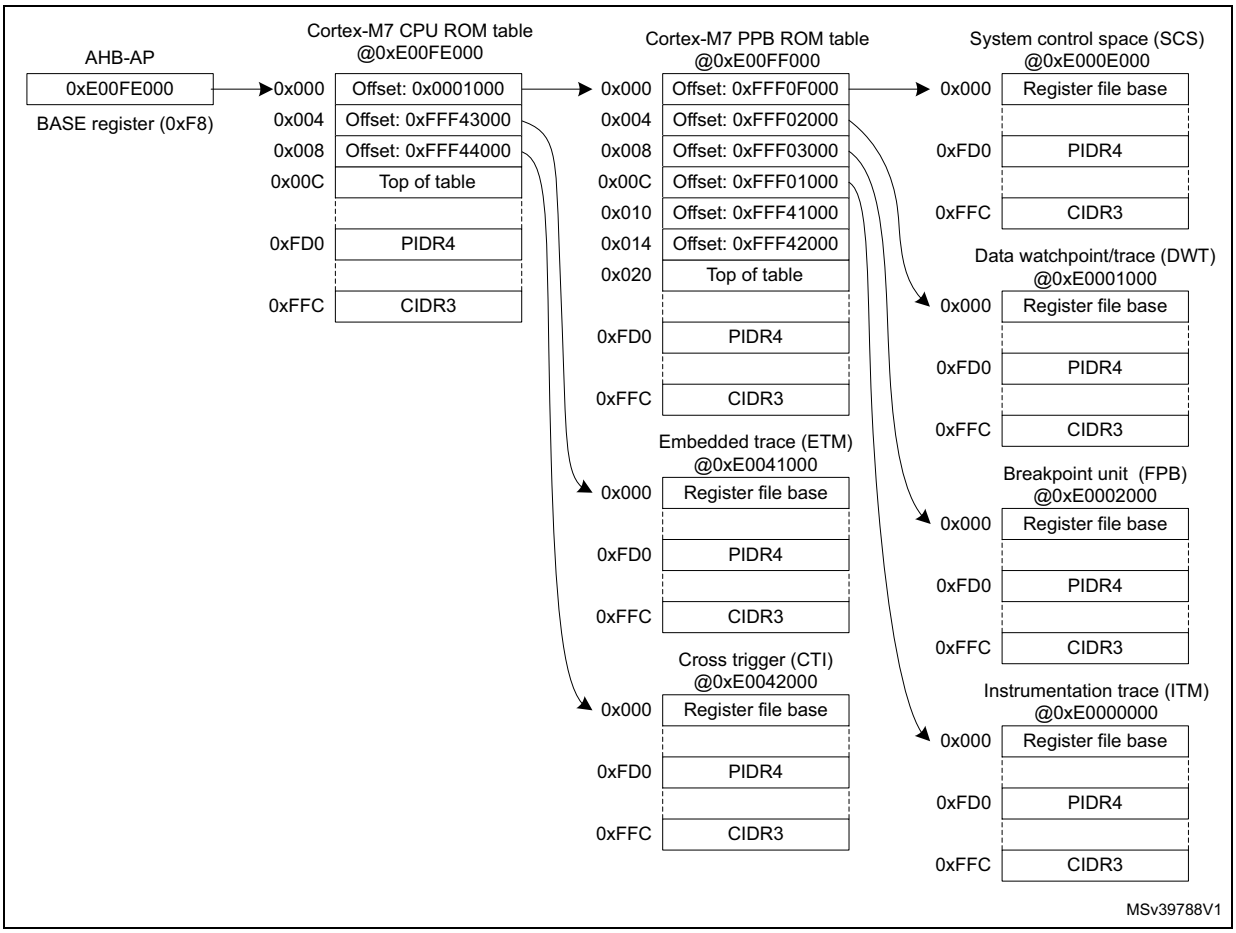
Table 590. Cortex-M7 PPB ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00FF018	Top of table	-	-	-	0x00000000
0xE00FF01C to 0xE00FFFC8	Reserved	-	-	-	0x00000000
0xE00FFFCC to 0xE00FFFFC	ROM table registers	-	-	-	See *Table 592*

1. The TPIU and ETM are included in this table by default, but bit 0 is reset to indicate that they are not present.

The Topology for the CoreSight™ components in the Cortex-M7 subsystem is shown in *Figure 826*.

Figure 826. Cortex-M7 CoreSight Topology



Cortex-M7 CPU ROM registers

CPU ROM memory type register (M7_CPUROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory presence
1: System memory is present on this bus

CPU ROM CoreSight peripheral identity register 4 (M7_CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC continuation code

CPU ROM CoreSight peripheral identity register 0 (M7_CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0xC8: Cortex-M7 Processor ROM table

CPU ROM CoreSight peripheral identity register 1 (M7_CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: Cortex-M7 Processor ROM table

CPU ROM CoreSight peripheral identity register 2 (M7_CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

CPU ROM CoreSight peripheral identity register 3 (M7_CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CPU ROM CoreSight component identity register 0 (M7_CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CPU ROM CoreSight component identity register 1 (M7_CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CPU ROM CoreSight component identity register 2 (M7_CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CPU ROM CoreSight component identity register 3 (M7_CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Cortex-M7 CPU ROM table register map and reset values

Table 591. Cortex-M7 CPU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	M7_CPUROM_MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	M7_CPUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	M7_CPUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	0	1	0	0	0
0xFE4	M7_CPUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	M7_CPUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	M7_CPUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	M7_CPUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	M7_CPUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0

Table 591. Cortex-M7 CPU ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	M7_CPUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	M7_CPUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Cortex-M7 PPB ROM registers

PPB ROM memory type register (M7_PPBROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory presence
1: System memory is present on this bus

PPB ROM CoreSight peripheral identity register 4 (M7_PPBROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC continuation code

PPB ROM CoreSight peripheral identity register 0 (M7_PPBROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0xC7: Cortex-M7 PPB ROM table

PPB ROM CoreSight peripheral identity register 1 (M7_PPBROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: Cortex-M7 PPB ROM table

PPB ROM CoreSight peripheral identity register 2 (M7_PPBROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

PPB ROM CoreSight peripheral identity register 3 (M7_PPBROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

PPB ROM CoreSight component identity register 0 (M7_PPBROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

PPB ROM CoreSight component identity register 1 (M7_PPBROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

PPB ROM CoreSight component identity register 2 (M7_PPBROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

PPB ROM CoreSight component identity register 3 (M7_PPBROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Cortex-M7 PPB ROM table register map and reset values

Table 592. Cortex-M7 PPB ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	M7_PPBROM_MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	M7_PPBROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															

Table 592. Cortex-M7 PPB ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	M7_PPBROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	0	0	1	1	1
0xFE4	M7_PPBROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	M7_PPBROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	M7_PPBROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	M7_PPBROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	M7_PPBROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	M7_PPBROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	M7_PPBROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.6.2 Cortex-M7 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (comparator 1 only)
- clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- clock cycles
- folded instructions
- load store unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares one of the following with the value held in its DWT_COMP register:

- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual *[5]*.

Cortex-M7 DWT registers

DWT control register (M7_DWT_CTRL)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0].				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: Number of comparators implemented (read-only)
0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read-only)
0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read-only)
0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read-only)
0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read-only)
0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA**: Enable for folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA**: Enable for sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA**: Enable for CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA**: Enable for exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: POSTCNT counter use enable
Enables use of POSTCNT counter as a timer for Periodic PC sample packet generation.

0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Position of synchronization packet counter tap on CYCCNT counter
This selection determines the synchronization packet rate.

0x0: Disabled - no synchronization packets
0x1: Tap at CYCCNT[24]
0x2: Tap at CYCCNT[26]
0x3: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Position of the POSTCNT tap on the CYCCNT counter
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter

Writes to this field are ignored if POSTCNT counter is enabled (that is, CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTPRESET[3:0]**: Reload value of the POSTCNT counter.

Bit 0 **CYCCNTENA**: CYCCNT counter enable

0: Disabled

1: Enabled

DWT cycle count register (M7_DWT_CYCCNT)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CYCCNT[31:0]**: Processor clock cycle counter

DWT CPI count register (M7_DWT_CPICNT)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter

Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNT, and counts any instruction fetch stalls.

DWT exception count register (M7_DWT_EXCCNT)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter
Counts the number of cycles spent in exception processing.

DWT sleep count register (M7_DWT_SLPCNT)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (M7_DWT_LSUCNT)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: Load store counter
Counts additional cycles required to execute load and store instructions.

DWT fold count register (M7_DWT_FOLDCNT)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter
Increments on each instruction that takes 0 cycles.

DWT program counter sample register (M7_DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **EIASAMPLE[31:0]**: Executed instruction address sample value
Samples the current value of the program counter.

DWT comparator register x (M7_DWT_COMPx)

Address offset: 0x020 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **COMP[31:0]**: Reference value for comparison.

DWT mask register x (M7_DWT_MASKx)

Address offset: 0x024 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **MASK[4:0]**: Comparator mask size

Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

DWT function register x (M7_DWT_FUNCTx)

Address offset: 0x028 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]			
							r					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAVADDR0[3:0]				DATAVSIZE[1:0]		LINK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match (read-only)

Indicates if a comparator match has occurred since the register was last read.

0: No match
1: Match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]**: Comparator number of a second comparator
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]**: Comparator number of a comparator
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]**: Size of required data comparison
For data value matching, specifies the size of the required data comparison.

0x0: Byte
0x1: Half word
0x2: Word
0x3: Reserved

Bit 9 **LNK1ENA**: Support of a second linked comparator (read-only)
Indicates whether use of a second linked comparator is supported (read-only).

1: Supported

Bit 8 **DATAVMATCH**: Cycle comparison enable
0: Perform address comparison
1: Perform data value comparison

Bit 7 **CYCMATCH**: Cycle count comparison enable on comparator 0
This field is reserved for other comparators.

0: No cycle count comparison
1: Compare DWT_COMP0 with the cycle counter, DWT_CYCCNT

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: Data trace address offset packet enable
Enables generation of data trace address offset packets (containing data address bits 0 to 15)
0: Disabled
1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: Action on comparator match
The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See *[5]*.

DWT CoreSight peripheral identity register 4 (M7_DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

DWT CoreSight peripheral identity register 0 (M7_DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r						

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x02: DWT part number

DWT CoreSight peripheral identity register 1 (M7_DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x0: DWT part number

DWT CoreSight peripheral identity register 2 (M7_DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

DWT CoreSight peripheral identity register 3 (M7_DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DWT CoreSight component identity register 0 (M7_DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

DWT CoreSight component identity register 1 (M7_DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

DWT CoreSight component identity register 2 (M7_DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

DWT CoreSight component identity register 3 (M7_DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Cortex-M7 DWT register map and reset values

The Cortex-M7 DWT registers are located at address range 0xE0001000 to 0xE0001FFC, on the AHBD.

Table 593. Cortex-M7 DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	M7_DWT_CTRL		NUMCOMP[3:0]			NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	M7_DWT_CYCCNT	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	M7_DWT_CPICNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x00C	M7_DWT_EXCCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 593. Cortex-M7 DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	M7_DWT_SLPCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	M7_DWT_LSUCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x018	M7_DWT_FOLDCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	M7_DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	M7_DWT_COMP0	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	M7_DWT_MASK0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x028	M7_DWT_FUNCT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x02C	Reserved	Reserved																															
0x030	M7_DWT_COMP1	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	M7_DWT_MASK1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x038	M7_DWT_FUNCT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x03C	Reserved	Reserved																															
0x040	M7_DWT_COMP2	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	M7_DWT_MASK2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x048	M7_DWT_FUNCT2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x04C	Reserved	Reserved																															
0x050	M7_DWT_COMP3	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 593. Cortex-M7 DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	M7_DWT_MASK3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x058	M7_DWT_FUNCT3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x05C to 0xFCC	Reserved	Reserved																															
0xFD0	M7_DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]			JEP106CON[3:0]				
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFCC	Reserved	Reserved																															
0xFE0	M7_DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	1	0
0xFE4	M7_DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	M7_DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	1	1	1	0	1	1
0xFEC	M7_DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	M7_DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	M7_DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	1	1	1	0	0	0	0
0xFF8	M7_DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	M7_DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.6.3 Cortex-M7 instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. Software trace

   Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access, and the data written, into a packet that it writes to a FIFO. The ITM

outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. Hardware trace

   The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. Local timestamping

   The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

4. Global system timestamping

   Timestamps can also be generated using the system-wide 64-bit count value coming from the Timestamp Generator component.

Cortex-M7 ITM registers

ITM stimulus register x (M7_ITM_STIMx)

Address offset: 0x000 + x * 0x4 (x = 0 to 31)

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMULUS[31:0]**: Software event packet / FIFOREADY

Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FIFOREADY indicator:

0: Stimulus port buffer is full (or port is disabled)
1: Stimulus port can accept new write data

ITM trace enable register (M7_ITM_TER)

Address offset: 0xE00

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable
Each bit n (0:31) enables the stimulus port associated with the M7_ITM_STIMn register.

0: Port disabled
1: Port enabled

ITM trace privilege registers (M7_ITM_TPR)

Address offset: 0xE40

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: Enable unprivileged access to ITM stimulus ports
Each bit controls eight stimulus ports:

0bXXX0: Unprivileged access permitted on ports 0 to 7
0bXXX1: Only privileged access permitted on ports 0 to 7
0bXX0X: Unprivileged access permitted on ports 8 to 15
0bXX1X: Only privileged access permitted on ports 8 to 15
0bX0XX: Unprivileged access permitted on ports 16 to 23
0bX1XX: Only privileged access permitted on ports 16 to 23
0b0XXX: Unprivileged access permitted on ports 24 to 31
0b1XXX: Only privileged access permitted on ports 24 to 31

ITM trace control register (M7_ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	GTSFREQ[1:0]		TSPRESCALE [1:0]		Res.	Res.	Res.	SWOE NA	TXENA	SYNC ENA	TSENA	ITM ENA
				rw	rw	rw					r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: ITM busy
Indicates whether the ITM is currently processing events (read-only):

0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field. Note: different IDs must be used for each trace source in the system.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:10 **GTSFREQ[1:0]**: Global timestamp frequency
Defines how often the ITM generates a global timestamp, based on the global timestamp clock frequency, or disables generation of global timestamps. The possible values are:

0x0: Disable generation of global timestamps
0x1: Generate timestamp request whenever the ITM detects a change in global timestamp counter bits [63:7]; this is approximately every 128 cycles
0x2: Generate timestamp request whenever the ITM detects a change in global timestamp counter bits [63:13]; this is approximately every 8192 cycles
0x3: Generate a timestamp after every packet, if the output FIFO is empty

Bits 9:8 TSPRESCALE[1:0]: Local timestamp prescale
Prescale used with the trace packet reference clock The possible values are:

0x0: No prescaling
0x1: Divide by 4
0x2: Divide by 16
0x3: Divide by 64

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SWOENA**: Asynchronous clocking enable for the timestamp counter (read-only)
0: Timestamp counter uses processor clock

Bit 3 **TXENA**: Hardware event packet forwarding enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.

0: Disabled
1: Enabled

Bit 2 **SYNCENA**: Synchronization packet transmission enable
If a debugger sets this bit it must also configure the DWT_CTRL register SYNCTAP field in the DWT for the correct synchronization speed.

0: Disabled
1: Enabled

Bit 1 **TSENA**: Local timestamp generation enable
0: Disabled
1: Enabled

Bit 0 **ITMENA**: ITM enable
0: Disabled
1: Enabled

ITM CoreSight peripheral identity register 4 (M7_ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ITM CoreSight peripheral identity register 0 (M7_ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x01: ITM part number

ITM CoreSight peripheral identity register 1 (M7_ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x1: ITM part number

ITM CoreSight peripheral identity register 2 (M7_ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ITM CoreSight peripheral identity register 3 (M7_ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ITM CoreSight component identity register 0 (M7_ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ITM CoreSight component identity register 1 (M7_ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ITM CoreSight component identity register 2 (M7_ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ITM CoreSight component identity register 3 (M7_ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Cortex-M7 ITM register map and reset values

The ITM registers are located at address range 0xE0000000 to 0xE0000FFC, on the AHBD.

Table 594. Cortex-M7 ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	M7_ITM_STIM0-31	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x080 to 0xDFC	Reserved	Reserved																															
0xE00	M7_ITM_TER	STIMENA[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xE04 to 0xE3C	Reserved	Reserved																															
0x0E40	M7_ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
	Reset value																													0	0	0	0
0xE44 to 0xE7C	Reserved	Reserved																															
0xE80	M7_ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	GTSFREQ[1:0]		TSPRESCALE[1:0]		Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0					0	0	0	0				0	0	0	0	0
0xE84 to 0xFCC	Reserved	Reserved																															

Table 594. Cortex-M7 ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	M7_ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	M7_ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	M7_ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	M7_ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	1	1	1	0	1	1
0xFEC	M7_ITM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	M7_ITM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	M7_ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	M7_ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	M7_ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.6.4 Cortex-M7 breakpoint unit (FPB)

The FPB allows hardware breakpoints to be set. It contains eight comparators which monitor the instruction fetch address and return a breakpoint instruction when a match is detected.The Cortex-M7 FPB does not support flash patch functionality.

Cortex-M7 FPB registers

FPB control register (M7_FPB_CTRL)

Address offset: 0x000

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r	r	r	r	r	r	r	r	r			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **NUM_CODE[6:4]**: Instruction address comparator number field, three MSBs
This read-only field holds the three MSBs of the number of instruction address comparators supported.

0x0: the MSBs of the number are all 0

Bits 11:8 **NUM_LIT[3:0]**: Number of literal address comparators supported (read-only).
0x0: No literal comparators supported.

Bits 7:4 **NUM_CODE[3:0]**: Instruction address comparator number field, four LSBs
This read-only field holds the four LSBs of the number of instruction address comparators supported.

0x8: 8 instruction comparators supported

Bit 1 **KEY**: Write protect key
A write to M7_FPB_CTRL register will be ignored if this bit is not set to 1.

Bits 0 **ENABLE**: FPB enable
0: Disable
1: Enable

FPB remap register (M7_FPB_REMAP)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	RMPSPT	REMAP[23:11]												
		r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REMAP[10:0]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **RMPSPT**: Flash patch remap support (read-only)
0: Remapping not supported

Bits 28:5 **REMAP[23:0]**: Reserved - not supported

Bits 4:0 Reserved, must be kept at reset value.

FPB comparator registers (M7_FPB_COMPx)

Address offset: 0x008 + x * 0x4 (for x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REPLACE[1:0]		Res.	COMP[26:14]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[13:0]														Res.	ENABLE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:30 **REPLACE[1:0]**: Behavior upon COMP versus instruction fetch address match
Defines the behavior when a match occurs between the COMP field and the instruction fetch address:

0x0: Reserved
0x1: Breakpoint on lower half-word, upper half-word is unaffected.
0x2: Breakpoint on upper half-word, lower half-word is unaffected.
0x3: Breakpoint on both upper and lower half-words.

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[26:0]**: Value to compare with code memory access address
Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF). If a match occurs, the action to take is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Comparator enable
The comparator is only enabled if both this bit and the FPB ENABLE bit in the M7_FPB_CTRL register are set.

0: Disabled
1: Enabled

FPB CoreSight peripheral identity register 4 (M7_FPB_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

FPB CoreSight peripheral identity register 0 (M7_FPB_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x0C: FPB part number

FPB CoreSight peripheral identity register 1 (M7_FPB_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x0: FPB part number

FPB CoreSight peripheral identity register 2 (M7_FPB_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x2: r0p3

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

FPB CoreSight peripheral identity register 3 (M7_FPB_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

FPB CoreSight component identity register 0 (M7_FPB_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

FPB CoreSight component identity register 1 (M7_FPB_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

FPB CoreSight component identity register 2 (M7_FPB_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

FPB CoreSight component identity register 3 (M7_FPB_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Cortex-M7 FPB register map and reset values

The Cortex-M7 FPB registers are located at address range 0xE0002000 to 0xE0002FFC.

Table 595. Cortex-M7 FPB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	M7_FPB_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	Reset value																		0	0	0	0	0	0	0	1	0	0	0			0	0

Table 595. Cortex-M7 FPB register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	M7_FPB_REMAP	Res.	Res.	RMPSP	REMAP[23:0]																								Res.	Res.	Res.	Res.	Res.
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x008 to 0x020	M7_FPB_COMP0 to M7_FPB_COMP7	REPLACE[1:0]		Res.	COMP[26:0]																								Res.	ENABLE			
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024 to 0xFCC	Reserved	Reserved																															
0xFD0	M7_FPB_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]			JEP106CON[3:0]				
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	M7_FPB_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	1	1	0	0
0xFE4	M7_FPB_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	M7_FPB_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	M7_FPB_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	M7_FPB_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	M7_FPB_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	M7_FPB_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	M7_FPB_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.6.5 Cortex-M7 embedded trace macrocell (ETM)

The Cortex-M7 ETM is a CoreSight™ component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M7 core to be traced. In the STM32H7, the ETM is configured for instruction trace only, so data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- The number of instructions executed in the same cycle
- Changes in program flow
- The current processor instruction state
- The addresses of memory locations accessed by load and store instructions
- The type, direction and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM™-M7 technical reference manual *[6]*.

Cortex-M7 ETM registers

ETM programming control register (M7_ETM_PRGCTL)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Trace program enable
0: Trace unit is disabled
1: Trace unit is enabled

ETM processor select control register (M7_ETM_PROCSEL)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROCSEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PROCSEL**: Processor select

This field has no effect since only the Cortex-M7 uses this ETM.

ETM status register (M7_ETM_STAT)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: Programmers model stable

Indicates whether the ETM registers are stable and can be read.

0: Registers are not stable

1: Registers are stable

Bit 0 **IDLE**: Trace unit inactive

0: ETM is not idle

1: ETM is idle

ETM trace configuration register (M7_ETM_CONFIG)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	TS	COND[2:0]			Res.	Res.	Res.	CCI	BB	INSTP0[1:0]		Res.
			rw	rw	rw	rw	rw				rw	rw	r	r	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DV**: Data value tracing (read-only)
0: Disabled

Bit 16 **DA**: Data address tracing (read-only)
0: Disabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RS**: Return stack enable
0: Disabled
1: Enabled

Bit 11 **TS**: Global timestamp tracing
0: Disabled
1: Enabled

Bits 10:8 **COND[2:0]**: Conditional instruction tracing
0x0: Conditional instruction tracing disabled
0x1: Conditional load instructions are traced
0x2: Conditional store instructions are traced
0x3: Conditional load and store instructions are traced
0x7: All conditional instructions are traced
Other: Reserved

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CCI**: Cycle counting in instruction trace
0: Disabled
1: Enabled

Bit 3 **BB**: Branch broadcast mode
0: Disabled
1: Enabled

Bits 2:1 **INSTP0[1:0]**: Determines which instructions are P0 instructions (read-only)
0x0: Only branches are P0 instructions

Bit 0 Reserved, must be kept at reset value.

ETM event control 0 register (M7_ETM_EVENTCTL0)

Only accepts writes when trace unit is disabled

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: Resource type for event 1
0: Single selected resource
1: Boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: Resource / Boolean combined resource pair, for event 1
When TYPE1 is 0, selects a single selected resource from 0-15 defined by bits[3:0]
When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

Bit 7 **TYPE0**: Resource type for event 0
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: Resource / Boolean combined resource pair for event 0
When TYPE0 is 0, selects a single selected resource from 0-15 defined by bits[3:0]
When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

ETM event control 1 register (M7_ETM_EVENTCTL1)

Only accepts writes when trace unit is disabled

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[3:0]			
			rw	rw								rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: Low power state behavior override
0: Low power state normal behavior
1: Entry to low power state does not affect resources and event trace generation

Bit 11 **ATB**: ATB trigger enable
0: Disabled
1: Enabled

Bits 10:4 Reserved, must be kept at reset value.

Bits 3:0 **INSTEN[3:0]**: Instruction trace event element enable
Each bit corresponds to an event:

0bXXX0: Event 0 does not cause an event element
0bXXX1: Event 0 causes an event element
0bXX0X: Event 1 does not cause an event element
0bXX1X: Event 1 causes an event element
0bX0XX: Event 2 does not cause an event element
0bX1XX: Event 2 causes an event element
0b0XXX: Event 3 does not cause an event element
0b1XXX: Event 3 causes an event element

ETM stall control register (M7_ETM_STALLCTL)

Only accepts writes when trace unit is disabled

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DSTALL	ISTALL	Res.	Res.	Res.	Res.	LEVEL[1:0]		Res.	Res.
						rw	rw					rw	rw		

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DSTALL**: Stall processor based on data trace buffer space
0: Do not stall processor
1: Stall processor

Bit 8 **ISTALL**: Stall processor based on instruction trace buffer space
0: Do not stall processor
1: Stall processor

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **LEVEL[1:0]**: Stalling threshold level

A low level minimizes the amount of processor stalling, with a higher risk of FIFO overflow. A high level minimizes the risk of FIFO overflow but increases the amount of processor stalling.

Bits 1:0 Reserved, must be kept at reset value.

ETM global timestamp control register (M7_ETM_TSCTL)

Only accepts writes when trace unit is disabled

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
								rw				rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TYPE**: Resource type for time stamp insertion

0: Single selected resource

1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]**: Resource / Boolean combined resource pair

When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]

When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

ETM synchronization period register (M7_ETM_SYNCP)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: Trace bytes between synchronization requests

Defines the number of bytes of trace information between trace synchronization requests.

0xA: 1024 bytes

ETM cycle count control register (M7_ETM_CCCTL)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: Threshold value for instruction trace cycle counting
The threshold represents the minimum interval between cycle count trace packets.
0x0: Reserved
Other: Threshold

ETM trace ID register (M7_ETM_TRACEID)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace ID
0x00: Reserved
0x01 to 0x6F: Valid ID
0x70 to 0x7F: Reserved

ETM ViewInst main control register (M7_ETM_VICTL)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S3	Res.	Res.	EXLEVEL_S0
												rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
				rw	rw	rw		rw				rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **EXLEVEL_S3**: Trace disable, exception level 3
Disables tracing in the specified exception level in Secure state for exception level 3.
0: Enable ViewInst in this exception level
1: Disable ViewInst in this exception level

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **EXLEVEL_S0**: Trace disable, exception level 0
Disables tracing in the specified exception level in Secure state for exception level 0.
0: Enable ViewInst in this exception level
1: Disable ViewInst in this exception level

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: Tracing of system error exception
Selects whether a system error exception must always be traced.

0: System error exception is traced only if the instruction or exception immediately before the system error exception is traced
1: System error exception is always traced regardless of the value of ViewInst

Bit 10 **TRCRESET**: Tracing of reset exception
Selects whether a reset exception must always be traced.

0: Reset exception is traced only if the instruction or exception immediately before the reset exception is traced
1: Reset exception is always traced regardless of the value of ViewInst

Bit 9 **SSSTATUS**: Current status of the start/stop logic
0: Stop state
1: Started state

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TYPE**: Resource type
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]**: Resource / Boolean combined resource pair
When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]
When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

ETM ViewInst start/stop control register (M7_ETM_VISSCTL)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **STOP[7:0]**: Selector of single address comparators to stop trace
Defines the single address comparators to stop trace with the ViewInst Start/Stop control. One bit is provided for each implemented single address comparator.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **START[7:0]**: Selector of single address comparators to start trace
Defines the single address comparators to start trace with the ViewInst Start/Stop control. One bit is provided for each implemented single address comparator.

ETM ViewInst start/stop processor comparator control register (M7_ETM_VIPCSSCTL)

Address offset: 0x08C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[3:0]			
												rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **STOP[3:0]**: Selector of processor comparator input to stop trace

Selects which processor comparator inputs are in use with ViewInst start-stop control, for the purpose of stopping trace. One bit is provided for each processor comparator input.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **START[3:0]**: Selector of processor comparator input to start trace

Selects which processor comparator inputs are in use with ViewInst start-stop control, for the purpose of starting trace. One bit is provided for each processor comparator input.

ETM counter reload value register (M7_ETM_CNTRLDV)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: Counter reload value

This value is loaded into the counter each time the reload event occurs.

ETM ID register 8 (M7_ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXSPEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXSPEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MAXSPEC[31:0]**: Maximum speculation depth

Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.

0x2: Maximum trace speculation depth is 2

ETM ID register 9 (M7_ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: Number of P0 right-hand keys used
0x0: No P0 keys used in instruction trace only configuration

ETM ID register 10 (M7_ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: Total number of P1 right-hand keys
Indicates the total number of P1 right-hand keys, including normal and special keys.
0x0: No P1 keys used in instruction trace only configuration

ETM ID register 11 (M7_ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: Total number of special P1 right-hand keys used
0x0: No special P1 keys used

ETM ID register 12 (M7_ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDKEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDKEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDKEY[31:0]**:
Indicates the total number of conditional instruction right-hand keys, including normal and special keys.
0x1: One conditional instruction right hand-key implemented

ETM ID register 13 (M7_ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDSPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDSPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDSPC[31:0]**: Number of special conditional instruction right-hand keys
0x0: No special conditional instruction right hand-keys implemented

ETM implementation specific register 0 (M7_ETM_IMSPEC0)

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: Support for implementation specific extensions
0x0: No implementation specific extensions are supported

ETM ID register 0 (M7_ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x0C00 1EE1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	TSSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSUPP[1]
		r	r	r	r	r	r								r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	Res.	Res.	Res.	Res.	Res.
r		r	r	r	r	r		r	r	r					

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: Meaning of the commit field in some packets
0: Commit mode 0

Bits 28:24 **TSSIZE[4:0]**: Global timestamp size
0x08: Maximum of 64-bit global timestamp implemented

Bits 23:17 Reserved, must be kept at reset value.

Bits 16:15 **QSUPP[1:0]**: Q element support
0x0: Q elements not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: Way of conditional result tracing
0x1: APSR condition flag values traced

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported in the trace
0x1: Two events supported for instruction only configuration

Bit 9 **RETSTACK**: Return stack support
1: Two entry return stack supported

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: Support for cycle counting in the instruction trace
1: Cycle counting in the instruction trace is implemented

Bit 6 **TRCCOND**: Support for conditional instruction tracing
1: Conditional instruction trace is implemented

Bit 5 **TRCBB**: Support for branch broadcast tracing
1: Branch broadcast trace is implemented

Bits 4:0 Reserved, must be kept at reset value.

ETM ID register 1 (M7_ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F401

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Trace unit designer entity
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: Major trace unit architecture version number
0x4: ETM v4

Bits 7:4 **TRCARCHMIN[3:0]**: Minor trace unit architecture version number
0x0: Minor version 0

Bits 3:0 **REVISION[3:0]**: Implementation revision number
0x1: Rev 1

ETM ID register 2 (M7_ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: Cycle counter size
Indicates the size of the cycle counter in bits minus 12.
0x0: Cycle counter is 12 bits

Bits 24:20 **DVSIZE[4:0]**: Data value size in bytes
0x0: Data value size is not supported in instruction only configuration

Bits 19:15 **DASIZE[4:0]**: Data address size in bytes
0x0: Data address size is not supported in instruction only configuration

Bits 14:10 **VMIDSIZE[4:0]**: Virtual machine ID size
0x0: Virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: Context ID size
0x0: Context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: Instruction address size
0x4: 32-bit maximum address size

ETM ID register 3 (M7_ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0509 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: Support of NOOVERFLOW
Indicates whether the NOOVERFLOW of trace stall control is implemented.
0: Not implemented

Bits 30:28 **NUMPROC[2:0]**: Number of processors available for tracing
0x0: Only one processor can be traced

Bit 27 **SYSSTALL**: System support for stall control of the processor
0: Not supported

Bit 26 **STALLCTL**: Stall control support
1: Trace stall control (TRCSTALLCTLR) is implemented

Bit 25 **SYNCPR**: Trace synchronization period support
0: TRCSYNCPR is read-only for instruction trace only configuration; the trace synchronization period is fixed

Bit 24 **TRCERR**: Support of TRCVICTLR.TRCERR
Indicates whether TRCVICTLR.TRCERR is implemented.
0x4: 32-bit maximum address size

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Support of privilege levels
Privilege levels are implemented; one bit for each level.
0x9: Privilege levels Thread and Handler are implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: Instruction trace cycle counting minimum threshold
0x4: Minimum threshold is 4 instruction trace cycle

ETM ID register 4 (M7_ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0001 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPADC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: Number of Virtual Machine ID comparators implemented
0x0: None

Bits 27:24 **NUMCIDC[3:0]**: Number of Context ID comparators implemented
0x0: None

Bits 23:20 **NUMSSCC[3:0]**: Number of single-shot comparator controls implemented
0x0: None

Bits 19:16 **NUMRSPAIR[3:0]**: Number of resource selection pairs implemented
0x1: None

Bits 15:12 **NUMPC[3:0]**: Number of processor comparator inputs implemented
0x4: Four

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPADC**: Support of data address comparisons
0: Not implemented

Bits 7:4 **NUMDVC[3:0]**: Number of data value comparators implemented
0x0: None

Bits 3:0 **NUMACPAIRS[3:0]**: Number of address comparator pairs implemented.
0x0: None

ETM ID register 5 (M7_ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0402

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: Support of reduced function counter
1: Implemented

Bits 30:28 **NUMCNTR[2:0]**: Number of counters implemented
0x1: One counter implemented

Bits 27:25 **NUMSEQSTATE[2:0]**: Number of sequencer states implemented
0x0: None

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: Support of low-power state override
1: Implemented

Bit 22 **ATBTRIG**: Support of ATB trigger
1: Implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: Number of bits of trace ID
0x07: Seven-bit trace ID implemented.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: Number of external input selectors implemented
0x2: Two external input selectors implemented

Bits 8:0 **NUMEXTIN[8:0]**: Number of external inputs implemented
0x2: Two external inputs implemented

ETM resource selection register 2 (M7_ETM_RSCTL2)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: Inversion of result of a combined pair of resources
0: Not inverted
1: Inverted

Bit 20 **INV**: Inversion of the selected resources
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Selects a group of resources

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: Selector of resources from desired group

Selects one or more resources from the desired group. One bit is provided per resource from the group.

ETM resource selection register 3 (M7_ETM_RSCTL3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: Inversion of the selected resources

0: Not inverted

1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Selects a group of resources

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: Selector of resources from desired group

Selects one or more resources from the desired group. One bit is provided per resource from the group.

ETM single-shot comparator control register 0 (M7_ETM_SSCC0)

Address offset: 0x280

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: Single-shot comparator resource reset enable

Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.

0: Disabled

1: Reset enabled; multiple matches can occur

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (M7_ETM_SSCS0)

Address offset: 0x2A0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA	INST
													r	r	r

Bit 31 **STATUS**: Single-shot status

This indicates whether any of the selected comparators have matched. If SSCC0.RST is set to 0, the STATUS bit must be written with 0 in order to enable single-shot comparator control.

0: No match occurred

1: Match has occurred at least once.

Bits 30:3 Reserved, must be kept at reset value.

Bit 2 **DV**: Data value comparator support

0: Single-shot data value comparisons not supported

Bit 1 **DA**: Data address comparator support

0: Single-shot data address comparisons not supported

Bit 0 **INST**: Instruction address comparator support

1: Single-shot instruction address comparisons supported

ETM single-shot processor comparator input control register (M7_ETM_SSPCIC0)

Address offset: 0x2C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PC[7:0]**: Comparator input selector for single-shot control

Selects one or more processor comparator inputs for single-shot control. One bit is provided for each processor comparator input.

ETM power-down control register (M7_ETM_PDC)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												r			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: Power up request

Request to maintain power to the ETM and access to the trace registers.

0: Power not requested
1: Power requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (M7_ETM_PDS)

Address offset: 0x314

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: Sticky power-down state

This bit is set to 1 when power to the ETM registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.

0: Trace register power uninterrupted since the last read of PDS register
1: Trace register power interrupted since the last read of PDS register

Bit 0 **POWER**: ETM powered up

1: ETM is powered up; all registers are accessible

ETM claim tag set register (M7_ETM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

ETM claim tag clear register (M7_ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

ETM lock access register (M7_ETM_LAR)

Address offset: 0xFB0

Reset value: N/A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETM register write access

Enables write access to some ETM registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access
Other values: Disable write access

ETM lock status register (M7_ETM_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the M7_ETM_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock

This bit always returns zero when read by an external debugger.

0: Write access is permitted

1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism

The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists

1: Lock control mechanism is implemented

ETM authentication status register (M7_ETM_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug

0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

ETM CoreSight device architecture register (M7_ETM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4770 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Component architect
0x23B: Arm®

Bit 20 **PRESENT**: Indicates the presence of this register
1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: Rev 0

Bits 15:0 **ARCHID[15:0]**: Architecture ID
0x4A13: ETMv4 component

ETM CoreSight device type identity register (M7_ETM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (M7_ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETM CoreSight peripheral identity register 0 (M7_ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0075

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, field, bits [7:0]
0x75: ETM part number

ETM CoreSight peripheral identity register 1 (M7_ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETM part number

ETM CoreSight peripheral identity register 2 (M7_ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p2

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight peripheral identity register 3 (M7_ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (M7_ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETM CoreSight component identity register 1 (M7_ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: Debug component with CoreSight-compatible registers

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETM CoreSight component identity register 2 (M7_ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETM CoreSight component identity register 3 (M7_ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Cortex-M7 ETM register map and reset values

The ETM registers are accessed by the debugger via the Cortex-M7 PPB, at address range 0xE0041000 to 0xE0041FFC.

Table 596. Cortex-M7 ETM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	M7_ETM_PRGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	M7_ETM_PROCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROCSEL[2:0]		
	Reset value																														0	0	0
0x00C	M7_ETM_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value																															0	0
0x010	M7_ETM_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA	Res.	Res.	Res.	RS	TS	COND[2:0]			VMID	CID	Res.	CCI	BB	INSTP0[1:0]		Res.
	Reset value															0	0				0	0	0	0	0	0	0		0	0	0	0	1
0x014 to 0x01C	Reserved	Reserved																															
0x020	M7_ETM_EVENTCTL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value																	0				0	0	0	0	0				0	0	0	0
0x024	M7_ETM_EVENTCTL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[3:0]			
	Reset value																				0	0								0	0	0	0
0x02C	M7_ETM_STALLCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSTALL	ISATLL	Res.	Res.	Res.	Res.	LEVEL[1:0]		Res.	Res.
	Reset value																							0	0					0	0		
0x030	M7_ETM_TSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
	Reset value																									0				0	0	0	0
0x034	M7_ETM_SYNCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
	Reset value																												0	1	0	1	0
0x038	M7_ETM_CCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	M7_ETM_TRACEID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										0	0	0	0	0	0	0

Table 596. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044 to 0x07C	Reserved	Reserved																															
0x080	M7_ETM_VICTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S3	Res.	Res.	EXLEVEL_S0	Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
	Reset value													0			0					0	0	0		0				0	0	0	0
0x084	Reserved	Reserved																															
0x088	M7_ETM_VISSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x08C	M7_ETM _VIPCSSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[3:0]			
	Reset value													0	0	0	0													0	0	0	0
0x090 to 0x13C	Reserved	Reserved																															
0x140	M7_ETM _CNTRLDV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144 to 0x17C	Reserved	Reserved																															
0x180	M7_ETM_IDR8	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x184	M7_ETM_IDR9	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	M7_ETM_IDR10	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	M7_ETM_IDR11	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	M7_ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	M7_ETM_IDR13	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198 to 0x1BC	Reserved	Reserved																															
0x1C0	M7_ETM_IMSPEC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
	Reset value																													0	0	0	0
0x1E0	M7_ETM_IDR0	Res.	Res.	COMMOPT	TSSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSUPP[1:0]		Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0	1	1	0	0								0	0		0	1	1	1	1		1	1	1					1

Table 596. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E4	M7_ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	0	0	0	0	0	0	1
0x1E8	M7_ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]					VMIDSIZE[4:0]			CIDSIZE[4:0]							IASIZE[4:0]			
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	M7_ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]		SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]													
	Reset value	0	0	0	0	0	1	0	1					1	0	0	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F0	M7_ETM_IDR4	NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]				NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0				0	0	0	0	0	0	0	0	0
0x1F4	M7_ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					0	1	0	0	0	0	0	0	0	0	1	0
0x1F8 to 0x204	Reserved	Reserved																															
0x208	M7_ETM_RSCTL2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											0	0		0	0	0									0	0	0	0	0	0	0	0
0x20C	M7_ETM_RSCTL3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												0		0	0	0									0	0	0	0	0	0	0	0
0x210 to 0x27C	Reserved	Reserved																															
0x280	M7_ETM_SSCC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0																								
0x284 to 0x29C	Reserved	Reserved																															
0x2A0	M7_ETM_SSCS0	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA	INST
	Reset value	0																													0	0	1

Table 596. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2A4 to 0x2BC	Reserved	Reserved																															
0x2C0	M7_ETM_SSPCIC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x2C4 to 0x30C	Reserved	Reserved																															
0x310	M7_ETM_PDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
	Reset value																													0			
0x314	M7_ETM_PDS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318 to 0xF9C	Reserved	Reserved																															
0xFA0	M7_ETM_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	M7_ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFA8to 0xFAC	Reserved	Reserved																															
0xFB0	M7_ETM_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	M7_ETM_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	M7_ETM_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	1	0	1	0
0xFBC	M7_ETM_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHID[15:0]															
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0to 0xFC8	Reserved	Reserved																															
0xFCC	M7_ETM_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Table 596. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	M7_ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	M7_ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	1	1	0	1	0	1
0xFE4	M7_ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	M7_ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	M7_ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	M7_ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	M7_ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	M7_ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	M7_ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

59.6.6 Cortex-M7 cross trigger interface (CTI)

See *Section 59.5.3*.

59.7 References for debug infrastructure

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B
4. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H
5. DDI 0403D (ID100710) - Arm®v7-M Architecture Reference Manual, Issue E.b
6. DDI 0494-2a (ID062813) - Arm® CoreSight™ ETM™-M7 r0p1 Technical Reference Manual, Issue D

60 Device electronic signature

The electronic signature is stored in the Flash memory area. It can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the STM32H743/53xx and STM32H750xB microcontrollers.

60.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

Base address: 0x1FF1 E800

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(31:0)																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(31:0):** 31:0 unique ID bits

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(63:48)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(63:32):** 63:32 unique ID bits

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(95:80)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(95:64):** 95:64 Unique ID bits.

60.2 Flash size

Base address: 0x1FF1 E880

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 F_ID(15:0): Flash memory size

This bitfield indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x0400 corresponds to 1024 Kbytes.

60.3 Package data register

Refer to SYSCFG package register (SYSCFG_PKGR) for package identification. The SYSCFG clock should be enabled first in the RCC_APB4ENR register.

61 Revision history

Table 597. Document revision history

Date	Revision	Changes
12-Dec-2016	1	Initial release.
26-May-2017	2	***Section 2*** renamed Memory and bus architecture. Updated *Section : Embedded bootloader*. **Section 3: Embedded Flash memory (FLASH)** Updated *Section : Flash sector erase*, *Section : Standard Flash bank erase*, *Section : Flash bank erase with automatic protection removal* and *Section : Flash mass erase*. Added *Section : Flash mass erase with automatic protection removal*. Updated *Table 16: FLASH register map and reset value*: – FLASH_BOOT7_CURR/PRG (add offset = 0x140/0x144) available only on RM0399. – Added FLASH_CRCEADD2R at add. offset = 0x158 – Added FLASH_CRCEADD2R at add offset = 0x15C **Section 6: Power control (PWR)** Updated *Section 6.3.1: PWR pins and internal signals*. Updated VBAT in *Section 6.4: Power supplies*. Updated *Figure 13: Power supply overview* and *Figure 15: Device startup with VCORE supplied from voltage regulator*. Removed VCORE in *Section 6.5: Power supply supervision*. Updated *Figure 24: VCORE voltage scaling versus system power modes* and *Figure 25: Power control modes detailed state diagram*. **Section 8: Reset and Clock Control (RCC)** Names of all clock source selection bits changed from XXSCR to XXSEL. RC48 renamed HSI48. VSWRST bit renamed BDRST in RCC_BDCR and bit description modified. CAMITFEN renamed DCMIEN in RCC_AHB2ENR; and CAMITFLPEN renamed DCMILPEN in RCC_AHB2LPENR. FLITFLPEN bit renamed FLASHLPEN in RCC_AHB3LPENR. HDMICECLPEN renamed CECLPEN in RCC_APB1LLPENR, Updated peripheral kernel clock names in the whole document. Updated *Figure 34: System reset circuit* to remove VDDA. Updated maximum frequency for ADC1, 2, 3 in *Table 51: Kernel clock distribution overview*. Removed lsi_ck as USBxOTG clock in *Section : Peripherals dedicated to control and data transfer*.

Table 597. Document revision history (continued)

Date	Revision	Changes
26-May-2017	2 (continued)	***Section 8: Reset and Clock Control (RCC)*** (continued) Renamed USART7RST/EN/LPEN bits into UART7RST/EN/LPEN, and USART8RST/EN/LPEN bits into UART8RST/EN/LPEN in Section 8.7.31: RCC APB1 Peripheral Reset Register *(RCC_APB1LRSTR)*, *Section 8.7.43: RCC APB1 Clock Register (RCC_APB1LENR)* and *Section 8.7.52: RCC APB1 Low Sleep Clock Register (RCC_APB1LLPENR)*. **Section 10: Hardware semaphore (HSEM)** Renamed pclk into hsem_hclk in the whole document. Updated COREID bit description in *Section 10.4.1: HSEM register (HSEM_R0 - HSEM_R31)*. **Section 11: General-purpose I/Os (GPIO)** *Table 84: GPIO register map and reset values*: – updated GPIOx_MODER reset values – changed index to A to K for GPIOx_AFRH. – added GPIOC..K_PUPDR **Section 12: System configuration controller (SYSCFG)** Updated SYSCFG_UR3 register. **Section 13: Block interconnect** In *Table 92: DMAMUX1, DMA1 and DMA2 connections*, renamed dac1_dma and dac2_dma into dac_ch1_dma and dac_ch2_dma, respectively. Updated several source and destination signals in *Table 88: Peripherals interconnect matrix details* and *Table 89: EXTI wakeup inputs*. **Section 17: DMA request multiplexer (DMAMUX)** Updated resources in *Table 110: DMAMUX1: assignment of multiplexer inputs to resources* to *Table 112: DMAMUX1: assignment of synchronization inputs to resources*. **Section 19: Nested Vectored Interrupt Controllers** Added LCD-TFT interrupts (ltdc_it and ltdc_err_it) in *Table 130: NVIC*. Extended interrupt and event controller (EXTI) Replaced DMA1 by BDMA for events 66 to 73 in *Table 133: EXTI Event input mapping*.

Table 597. Document revision history (continued)

Date	Revision	Changes
26-May-2017	2 (continued)	**Section 22: Flexible memory controller (FMC)** – Updated internal signals in *Figure 86: FMC block diagram* – HCLK renamed fmc_hclk – KCK_FMC renamed fmc_ker_ck Updated *Section 22.5: AXI interface* to add 32-bit accesses. Read FIFO depth changed to 6x64 bits. AXI bus width correct (64 bits instead of 32 bits). All waveforms made generic for what regards data bus and NBL bits. In *Section : SRAM/NOR-Flash chip-select timing registers 1..4 (FMC_BTR1..4)*, modified DATAST example and updated BURSTURN description. In *Section : SRAM/NOR-Flash write timing registers 1..4 (FMC_BWTR1..4)*: updated BURSTURN description. Updated *Section : SDRAM Control registers 1,2 (FMC_SDCR1,2)* to add bitfield width. Added missing FMC_SDCR2 bits in *Table 181: FMC register map*. **Section 23: Quad-SPI interface (QUADSPI)** Updated internal signals in *Figure 115: QUADSPI block diagram when dual-flash mode is disabled* and *Figure 116: QUADSPI block diagram when dual-flash mode is enabled*. Added *Section 23.3.2: QUADSPI pins and internal signals*. Modified error type for access by Cortex CPU in *Section 23.3.7: QUADSPI memory-mapped mode*. Added *Section 23.3.8: QUADSPI Free running clock mode* as well as FRCM bit in *Section 23.5.6: QUADSPI communication configuration register (QUADSPI_CCR)*. Updated *Section 23.5.1: QUADSPI control register (QUADSPI_CR)*. In *Table 185: QUADSPI register map and reset values*, changed DMAEN bit to reserved for QUADSPI_CR register. **Section 24: Delay block (DLYB)** Added internal signals in *Figure 123: DLYB block diagram* and added *Section 24.3.2: DLYB pins and internal signals*. **Section 25: Analog-to-digital converters (ADC)** Number of ADCs changed to 3 in *Section 25.1: Introduction*. Added internal signals in *Section 25.3.1: ADC block diagram*. Changed ADCx_IN[19:0] into ADCx_INP[19:0] and ADCx_INN[19:0] in *Figure 124: ADC block diagram* and *Table 190: ADC input/output pins*. Changed ADC_CLK into adc_ker_ck. Updated *Figure 126: ADC1 connectivity*, *Figure 127: ADC2 connectivity* and *Figure 128: ADC3 connectivity* and added note below figure. Updated *Figure 175: Dual ADC block diagram(1)*. Updated notes in *Section : Dual clock domain architecture*. Updated *Section 25.3.11: Channel selection (SQRx, JSQRx)* and *Section 25.3.12: Channel preselection register (ADCx_PCSEL)*.

Table 597. Document revision history (continued)

Date	Revision	Changes
26-May-2017	2 (continued)	***Section 25: Analog-to-digital converters (ADC)*** **(**continued**)** Examples given for 24 MHz instead of 72 MHz in *Section 25.3.13: Channel-wise programmable sampling time (SMPR1, SMPR2)* and *Section 25.3.17: Timing*. 6-bit resolution removed and updated *Table 195: TSAR timings depending on resolution* in *Section 25.3.23: Programmable resolution (RES) - fast conversion mode*. Updated notes in *Section : Combined regular/injected simultaneous mode* and *Section : Combined regular simultaneous + alternate trigger mode*. Added *Section : DFSDM mode in dual ADC simultaneous mode*. Changed ADC3_IN18 into ADC3 VINP[18] in *Section 25.3.33: Temperature sensor*. Changed ADC3_IN17 into ADC3 VINP[17] in *Section 25.3.34: VBAT supply monitoring*. Changed ADC3_IN19 into ADC3 VINP[19] in *Section 25.3.35: Monitoring the internal voltage reference*. Changed all bit access type to 'r' in *Section 25.5.15: ADC x regular Data Register (ADCx_DR) (x=1 to 3)*. Updated *Table 206: ADC register map and reset values (master and slave ADC common registers) offset =0x300)*. **Section 26: Digital-to-analog converter (DAC)** Updated/added internal signals in *Figure 194: DAC channel block diagram*. Added *Section 26.3.2: DAC pins and internal signals*.. APB1 clock and LSI clock replaced by dac_pclk and by lsi_ck in the whole document. **Section 28: Comparator (COMP)** Replaced COMP_IFCR by COMP_ICFR in *Section 28.7.1: Comparator status register (COMP_SR)*. **Section 29: Operational amplifiers (OPAMP)** Replaced all occurrences of DACx_int by dac_outx (x = 1, 2). Updated VP_SEL bit description in *Section 29.6.1: OPAMP1 control/status register (OPAMP1_CSR)* register. **Section 31: Digital camera interface (DCMI)** Section *DCMI pins* merged with *Section 31.4.4: DCMI physical interface*. External signals HSYNC, VSYNC and PIXCLK standardized to DCMI_HSYNC, DCMI_VSYNC and DCIM_PIXCLK in the whole document. Updated *Figure 224: DCMI block diagram* and *Figure 225: Top level block diagram*.

Table 597. Document revision history (continued)

Date	Revision	Changes
26-May-2017	2 (continued)	**Section 32: LCD-TFT Display Controller (LTDC)** *LCD-TFT pins and signal interface* renamed *LCD-TFT pins and external signal interface*. Internal signal names updated in *Figure 233: LTDC block diagram*. ck_axi_d1 renamed ltdc_aclk in the whole document. Updated *Section 32.3.4: LTDC reset and clocks*. Updated CFBP bitfield size in *Section 32.7.23: LTDC Layerx Color Frame Buffer Length Register (LTDC_LxCFBLR) (where x=1..2)*. **Section 33: JPEG codec (JPEG)** Added Internal signal names in *Figure 238: JPEG codec block diagram* and added *Section 33.3.2: JPEG internal signals*. Suppressed DMA feature. Updated JPEG_CR, JPEG_SR and JPRG_CFR registers in *Table 259: JPEG codec register map and reset values*. **Section 37: High-Resolution Timer (HRTIM)** Renamed SCOUT into HRTIM_SCOUT in the whole section. Renamed Tx into HRTIM_CHxy in all figures where it is referred to. Updated internal signals in *Figure 274: High-resolution timer block diagram*. Added hrtim_in_sync1 and hrtim_out_sync1 in *Table 283: HRTIM Input/output summary*. Updated *Section : Definition of terms*. Updated internal signal names in all the figures where they are mentioned and in HRTIM functional description Added note related to hrtim_ker_ck in *Table 283: HRTIM Input/output summary*. Modified ADC3TAPER bit description in *Section 37.5.56: HRTIM ADC Trigger 3 Register (HRTIM_ADC3R)*. Modified ADC4TCRST, ADC4TAPER, ADC4TAC2, ADC4EEV6, ADC4MPER and DC4MC1 bit descriptions in and *Section 37.5.57: HRTIM ADC Trigger 4 Register (HRTIM_ADC4R)*. **Section 43: Low-power timer (LPTIM)** Added internal signals in *Figure 515: Low-power timer block diagram (LPTIM1 and LPTIM2)*, *Figure 516: Low-power timer block diagram (LPTIM3)* and *Figure 517: Low-power timer block diagram (LPTIM4 and LPTIM5)*. Added *Section 43.4.2: LPTIM pins and internal signals*. *Table 335: LPTIM1 external trigger connection* to *Table 339: LPTIM5 external trigger connection* updated and moved under *Section 43.4.3: LPTIM input and trigger mapping*. *Table 340: LPTIM1 Input 1 connection* to *Table 344: LPTIM3 Input 1 connection* updated and moved under *Section 43.4.3: LPTIM input and trigger mapping*. Updated TRIGSEL bitfield description in *Section 43.6.4: LPTIM configuration register (LPTIM_CFGR)*.

Table 597. Document revision history (continued)

Date	Revision	Changes
26-May-2017	2 (continued)	***Section 43: Low-power timer (LPTIM)*** (continued) Added caution note for COUNTRST bit in *Section 43.6.5: LPTIM control register (LPTIM_CR)*. Updated IN1SEL and IN2SEL bitfield description and added caution note in *Section 43.6.9: LPTIM configuration register 2 (LPTIM_CFGR2)*. Added caution note in *Section 43.6.10: LPTIM3 configuration register 2 (LPTIM3_CFGR2)*. **Section 46: Real-time clock (RTC)** Added *Figure 527: RTC block overview*. Updated title of *Figure 528: Detailed RTC block diagram*. Added *Section 46.3.2: RTC pins and internal signals*. Updated *Table 353: RTC pins and internal signals*. Updated *Section 46.3.3: GPIOs controlled by the RTC*. Updated *Section 46.6.16: RTC tamper configuration register (RTC_TAMPCR)*. **Section 47: Inter-integrated circuit (I2C) interface** Updated OA1[7:1] and OA2[7:1] bit description in *Section 47.7.3: Own address 1 register (I2C_OAR1)* and *Section 47.7.4: Own address 2 register (I2C_OAR2)*. Replaced HSI16 by HSI or CSI or internal oscillator in *Section 47.4.14: Wakeup from Stop mode on address match*. **Section 48: Universal synchronous asynchronous receiver transmitter (USART)** *Section 48.5.14: USART synchronous mode*: removed Figure *RX data setup/hold time, added* reference to synchronous mater mode in *Figure 574: USART data clock timing diagram in synchronous master mode (M bits ='00')* and *Figure 575: USART data clock timing diagram in synchronous master mode (M bits = '01')* and figure contents updated to mention two M bits instead of one. Updated *Figure 586: Wakeup event verified (wakeup event = address match, FIFO disabled)* in *Section 48.5.21: USART low-power management*. **Section 51: Serial audio interface (SAI)** Updated SYNCIN bitfield description in *Section 51.5.1: Global configuration register (SAI_GCR)*.

Table 597. Document revision history (continued)

Date	Revision	Changes
26-May-2017	2 (continued)	**Section 50: Serial peripheral interface (SPI)** Updated note in *Section : Simplex communications*. Added note about the PCM long and short frame definition in *Section : PCM standard*. Added *Section 50.9.3: Bits and fields usable in I2S/PCM mode* description and *Table 391: Bit fields usable in PCM/I2S mode*. Updated WSINV in *Section 50.11.14: SPI/I2S configuration register (SPI_I2SCGFR)* and *Section 50.12: SPI register map and reset values*. **Section 52: SPDIF receiver interface (SPDIFRX)** Added internal signals in *Section Figure 658.: SPDIFRX block diagram* and added *Section 52.3.1: SPDIFRX pins and internal signals*. **Section 55: Secure digital input/output MultiMediaCard interface (SDMMC)** – Renamed internal signals in the whole section; updated *Figure 694: SDMMC block diagram*; added *Section 55.4.2:*SDMMC pins and internal signals – Added *Section 55.4.7: MDMA request generation* – Removed SDMMC_VER, SDMMC_ID, SDMMC_SID – Updated SDMMC_STAR bit 12 and 13 in *Table 449: SDMMC register map*. **Section 56: FD Controller Area Network (FDCAN)** ASC (asynchronous serial communication) removed from the whole section. Added F0OM bit (bit 31) and F1OM bit (bit 31) in *FDCAN Rx FIFO 0 Configuration Register (FDCAN_RXF0C)* and *FDCAN Rx FIFO 1 Configuration Register (FDCAN_RXF1C)*, respectively. ***Section 57: USB on-the-go high-speed (OTG_HS)*** Added *Section 57.4.2: USB OTG pin and internal signals*. **Section 60: Debug infrastructure** Updated *Section 60.5.8: Microcontroller debug unit (DBGMCU)*. Replaced in all DBGMCU register names: – D1APB1 by APB3 – D2APB1 by APB1 – D2APB2 by APB2 – D3APB4byAPB4 WDGLSD2 bit changed to reserved in *Section : DBGMCU APB4* peripheral freeze register CPU *(DBGMCU_APB4FZ1)*DBGMCU_APB4FZ1. Removed reserved registers in *Table 578: DBGMCU register map and reset values*.

Table 597. Document revision history (continued)

Date	Revision	Changes
11-Aug-2017	3	**Section 2.3: Embedded SRAM** Added *Section : Error code correction (ECC)*. Section 12: System configuration controller (SYSCFG) Changed 2.5 to 2.7 V in HSLV bit description of *SYSCFG compensation cell control/status register (SYSCFG_CCCSR)* and IO_HSLV of *SYSCFG user register 17 (SYSCFG_UR17)*. **Section 3: Embedded Flash memory (FLASH)** In *Table 16: FLASH register map and reset value*: – Updated FLASH_OPTSR_CUR at address offset 0x11C. – Updated FLASH_OPTSR_PRG at address offset 0x120. Changed 2.5 to 2.7 V in IO_HSLV bit description of *FLASH option status register (current value) (FLASH_OPTSR_CUR)* and *FLASH option status register (value to program) (FLASH_OPTSR_PRG)*. **Section 6: Power control (PWR)** Replace AIEC by EXTI and wait VDD11 by Wait VCORE in Figure 27: Dynamic voltage scaling behavior with D1, D2 and *system in Stop mode*, *Figure 28: Dynamic Voltage Scaling D1, D2, system Standby mode* and *Figure 29: Dynamic voltage scaling* behavior with D1 and D2 in DStandby mode and D3 in autonomous *mode*. Updated *Section 6.4.7: USB regulator* Updated *Section : Entering Stop mode*. **Section 7: Low-power D3 domain** Replaced VDD11 by VCORE. **Section 8: Reset and Clock Control (RCC)** Updated maximum frequency for ADC1, 2, 3 in *Table 51: Kernel clock distribution overview*. **Section 10: Hardware semaphore (HSEM)** Whole section updated to align with the product features. **Section 14: MDMA controller (MDMA)** Removed iterator in MDMA_CxISR and MDMA_CxIFCR bit names. **Section 19: Nested Vectored Interrupt Controllers** Changed usart4_gbl_it into uart4_gbl_it in *Table 130: NVIC*. **Section 21: Cyclic redundancy check calculation unit (CRC)** Updated first two features of *Section 21.2: CRC main features*.

Table 597. Document revision history (continued)

Date	Revision	Changes
11-Aug-2017	3 (continued)	**Section 25: Analog-to-digital converters (ADC)** *Section 25.2: ADC main features*: suppressed ADC supply requirements. Renamed VTS into VSENSE in the whole section. Updated *Figure 124: ADC block diagram*, *Figure 126: ADC1 connectivity*, *Figure 127: ADC2 connectivity* and *Figure 128: ADC3 connectivity*. VINN connected to ADC_INN instead of ADC_INN-1 in *Table 190: ADC input/output pins*. *Section : I/O analog switches voltage booster*: voltage booster enable bit renamed BOOTSE (instead of BOOTSEN) and corresponding register renamed SYSCFG_PMCR (instead of SYSCFG_CFGR1). Added note related to software trigger selection in *Section 25.3.16: Starting conversions (ADSTART, JADSTART)*. Updated *Figure 136: Triggers are shared between ADC master and ADC slave*, *Figure 191: Temperature sensor channel block diagram*, *Figure 192: VBAT channel block diagram* and *Figure 193: VREFINT channel block diagram*. Updated *Section : Analog watchdog*. Updated software notification by interrupt at end of conversion in *Section : Interleaved mode with independent injected*; updated Figure 178: Interleaved mode on 1 channel in continuous *conversion mode: dual ADC mode* and *Figure 179: Interleaved mode on 1 channel in single conversion mode: dual ADC mode*. Removed temperature sensor precision and replaced ambient by junction temperature in *Section 25.3.33: Temperature sensor*. Section 25.6.2: ADC x common control register (ADCx_CCR) (x=12 *or 3)*: – VBATEN bit description made generic – TSEN renamed VSENSEEN and description made generic – Updated *Table 203: DELAY bits versus ADC resolution*. ADCx_OR register suppressed since all bits are reserved. Renamed ADCx_LHTR1 into ADCx_HTR1. Changed ADCx_ISR bit access type to rc_w1. Updated iterators for all ADC common registers **Section 26: Digital-to-analog converter (DAC)** Updated VREF+ range in *Table 207: DAC input/output pins* Updated *Section 26.3.5: DAC conversion*. Added note related to ENx limitation when CENx is set in *Section 26.3.12: DAC channel buffer calibration*. Added case of accesses to DHRxxxD for wave generation in *Section 26.3.13: Dual DAC channel conversion (if available)*. Renamed DAC_M_ID into DAC_SIDR and M_ID bits into SID.

Table 597. Document revision history (continued)

Date	Revision	Changes
11-Aug-2017	3 (continued)	***Section 26: Digital-to-analog converter (DAC)* (continued)** Added *Section 26.5: DAC interrupts* *Section 26.6.1: DAC x control register (DACx_CR) (x=1 to 2)*: – TEN2 bit: replaced DACx_DHRy by DACx_DHR2 – TEN1 bit: replaced DACx_DHRy by DACx_DHR1 in. – TSELx bits: updated information on trigger selection/mapping Section 26.6.16: DAC x mode control register (DACx_MCR) (x=1 to *2)*: added note related to ENx bit for MODEx bit description. Section 26.6.19: DAC x Sample and Hold hold time register *(DACx_SHHR)(x=1 to 2)*: added notes related to THOLDx modification when ENx=0. Section 26.6.20: DAC x Sample and Hold refresh time register *(DACx_SHRR)(x=1 to 2)*: added note related to TREFRESHx modification when ENx=0. Suppressed DAC_OR register. **Section 29: Operational amplifiers (OPAMP)** Updated OPAMP1_VINM internal connection in *Table 225: Operational amplifier possible connections*. **Section 37: High-Resolution Timer (HRTIM)** Updated Note *1.* in *Table 289: External events mapping and associated features*. **Section 44: System window watchdog (WWDG)** Corrected math equations used to calculate the WWDG period. **Section 47: Inter-integrated circuit (I2C) interface** Updated NACKCF bit definition in *Section 47.7.8: Interrupt clear register (I2C_ICR)* register. **Section 51: Serial audio interface (SAI)** SAIXEN bit renamed SAIEN in SAI_xCR register. Updated *Figure 643: PDM typical connection and timing* to make the block diagram independent from the number of data and clock lines. Updated *Figure 644: Detailed PDM interface block diagram* to add “n” and “p” indexes (n being the number of data lines and p the number of microphone pairs). **Section 52: SPDIF receiver interface (SPDIFRX)** Updated *Figure 658: SPDIFRX block diagram*. **Section 59: HDMI-CEC controller (HDMI-CEC)** Updated – *Section 59.2: HDMI-CEC controller main features* – Updated *Figure 798: HDMI-CEC block diagram*

Table 597. Document revision history (continued)

Date	Revision	Changes
11-Aug-2017	3 (continued)	**Section 61: Device electronic signature** Updated Unique device ID register base address in *Section 61.1: Unique device ID register (96 bits)*. Updated Flash size base address in *Section 61.2: Flash size*. Replaced package data register description by reference to SYSCFG_PKGR in *Section 61.3: Package data register*.
15-Jun-2018	4	***Section 2: Memory and bus architecture*** Updated *Figure 1: System architecture for STM32H743/53xx and STM32H750xB devices*. Updated *Table 2: Bus-master-to-bus-slave interconnect*. *Section 2.1.5: CPU buses*: updated Cortex®-M7 and Cortex®-M7 DTCM, core with FPU replaced by CPU for I-bus, D_bus and S-bus. *Section 2.1.6: Bus master peripherals*: updated MDMA controller, DMA1/2 controllers and BDMA controller. Added *Table 6: Memory map and default device memory area attributes* in *Section 2.3.2: Memory map and register boundary addresses*. Updated *Table 7: Register boundary addresses*. Major update of ***Section 3: Embedded Flash memory (FLASH)***. ***Section 4: Secure memory management***: Modified title. ***Section 5: Power control (PWR)*** Changed RC48 to HSI48 in *Figure 18: Power supply overview*. ***Section 7: Reset and Clock Control (RCC)*** Removed USB_PHY2 in the whole section. Updated maximum allowed clock frequency for ADC1, 2, 3 *Table 55: Kernel clock distribution overview*. Updated *Figure 52: Kernel clock distribution for USB (2)*. In RCC_AHB1ENR/RCC_C1_AHB1ENR/RCC_C2_AHB1ENR: renamed USB2OTGEN into USB2OTGHSEN, USB1ULPIEN into USB1OTGHSULPIEN, USB1OTGEN into USB1OTGHSEN. In RCC_AHB1LPENR/RCC_C1_AHB1LPENR/RCC_C2_AHB1LPENR: renamed USB2OTGLPEN into USB2OTGHSLPEN, USB1ULPILPEN into USB1OTGHSULPILPEN, USB1OTGLPEN into USB1OTGHSLPEN. Bits changed to reserved: TMPSENSRST of RCC_APB4RSTR register, TMPSENSEN of RCC_APB4ENR and TMPSENSLPEN of RCC_APB4LPENR.

Table 597. Document revision history (continued)

Date	Revision	Changes
15-Jun-2018	4 (continued)	***Section 11: System configuration controller (SYSCFG)*** Renamed SYSCFG_CMPCR into SYSCFG_CCCSR. Added *SYSCFG configuration register (SYSCFG_CFGR)*. *SYSCFG user register 2 (SYSCFG_UR2)*: changed High/medium/Low level to Level3/2/1. *SYSCFG user register 11 (SYSCFG_UR11)*: swapped IWDG1M configurations. *SYSCFG user register 12 (SYSCFG_UR12)*: swapped IWDG2M configurations. ***Section 12: Block interconnect*** In *Table 92: Peripherals interconnect matrix details*: – Changed EXTx into adc_ext_trgx and JEXTx into adc_jext_trgx for ADC3 – Updated D2/USB1/2 SOF connection Changed PRC0 to 3 into d3_pendclear_in[0] to d3_pendclear_in[3] in *Table 94: EXTI pending requests clear inputs*. Added DSI signals in *Table 95: MDMA*. Updated EXTI connection in *Table 97: DMAMUX2 and BDMA connections*: dmamux1_req_inx numbering changed to start from dmamux1_req_in1 instead of dmamux1_req_in0 in *Table 96: DMAMUX1, DMA1 and DMA2 connections* and *Table 97: DMAMUX2 and BDMA connections* ***Section 15: Basic direct memory access controller (BDMA)*** Major section update.

Table 597. Document revision history (continued)

Date	Revision	Changes
15-Jun-2018	4 (continued)	***Section 16: DMA request multiplexer (DMAMUX)*** Updated *Section 16.2: DMAMUX main features*. Updated all tables from *Table 115: DMAMUX1 and DMAMUX2 instantiation* to *Table 121: DMAMUX2: assignment of synchronization inputs to resources*. Updated *Section : Synchronization mode and channel event generation*. Removed note from *Section : Synchronization overrun and interrupt*. Updated *Section 16.4.5: DMAMUX request generator*, *Section 16.6.1: DMAMUX1 request line multiplexer channel x configuration register (DMAMUX1_CxCR)*, *Section 16.6.2: DMAMUX2 request line multiplexer channel x configuration register (DMAMUX2_CxCR)*, *Section 16.6.7: DMAMUX1 request generator channel x configuration register (DMAMUX1_RGxCR)* and *Section 16.6.8: DMAMUX2 request generator channel x configuration register (DMAMUX2_RGxCR)*. Removed *Section 12.6.13: DMAMUX illegal access flag register (DMAMUX_IAFR)*, *Section 12.6.14: DMAMUX illegal access flag clear status register (DMAMUX_IAFCR)* and *Section 12.6.15: DMAMUX illegal access interrupt enable register (DMAMUX_IAIER)*. ***Section 17: Chrom-Art Accelerator™ controller (DMA2D)*** Added *Section 17.4.8: DMA2D output FIFO byte reordering*. Added *Section : Memory-to-memory with PFC, blending and fixed color FG* and *Section : Memory-to-memory with PFC, blending and fixed color BG*. Updated *Section 17.6.1: DMA2D control register (DMA2D_CR)*: – Updated MODE[2:0] to MODE[3:0] bit description. – Added bit 6 LOM Updated *Section 17.6.5: DMA2D foreground offset register (DMA2D_FGOR)* LO[15:0] bit description. Updated *Section 17.6.7: DMA2D background offset register (DMA2D_BGOR)* LO[15:0] bit description. Updated *Section 17.6.9: DMA2D foreground color register (DMA2D_FGCOLR)*. Updated *Section 17.6.11: DMA2D background color register (DMA2D_BGCOLR)*. Updated *Section 17.6.14: DMA2D output PFC control register (DMA2D_OPFCCR)* adding SB bit 6 description. Updated *Section 17.6.17: DMA2D output offset register (DMA2D_OOR)* LO[15:0] bit description. Added identification registers: DMA2D_VERR, DMA2D_IPIDR, DMA2D_SIDR. ***Section 18: Nested Vectored Interrupt Controllers*** Updated *Table 138: NVIC* and *Section 18.1.1: SysTick calibration value register*.

Table 597. Document revision history (continued)

Date	Revision	Changes
15-Jun-2018	4 (continued)	***Section 19: Extended interrupt and event controller (EXTI)*** Added note related to WKUP1 to 6 in *Table 141: EXTI Event input mapping* ***Section 21: Flexible memory controller (FMC)*** Updated *Figure 94: FMC memory banks (default mapping)* and *Table 147: FMC bank mapping options* to change Bank 4 to “not used by FMC). Updated *Section : General transaction rules*. ***Section 22: Quad-SPI interface (QUADSPI)*** Updated access types in *Section 22.5.1: QUADSPI control register (QUADSPI_CR)* and *Section 22.5.4: QUADSPI flag clear register (QUADSPI_FCR)*. ***Section 24: Analog-to-digital converters (ADC)*** f_{ADC} and T_{ADC} replaced by $f_{ADC_ker_ck}$ and $t_{ADC_ker_ck}$ in the whole IP user specification. Added $f_{ADC_ker_ck}$ in *Figure 132: ADC clock scheme*. F_{HCLK} replaced by F_{adc_hclk} in *Section : Clock ratio constraint between ADC clock and AHB clock*. Replaced EXTSEL by JEXTSEL and adc_ext_trg0/1 by adc_jext_trg0/1 in *Table 202: ADC1, ADC2 and ADC3 - External triggers for injected channels*. Updated note for JEOIE in *Section 24.5.2: ADC interrupt enable register (ADC_IER)* Updated conditions for programming ADC_JSQR register in *Section 24.3.10: Constraints when writing the ADC control bits*, *Section 24.3.11: Channel selection (SQRx, JSQRx)*, as well as in notes related to all *Section 24.5.16: ADC injected sequence register (ADC_JSQR)* bitfields (product errata sheets updated accordingly). Updated note related to offset correction in *Section : Single ADC operating modes support when oversampling*. Updated *Section 24.5.8: ADC channel preselection register (ADC_PCSEL)*. Removed calibration value in *Section 24.5.26: ADC calibration factors register (ADC_CALFACT)*.

Table 597. Document revision history (continued)

Date	Revision	Changes
15-Jun-2018	4 (continued)	***Section 25: Digital-to-analog converter (DAC)*** Changed data register indexing scheme in the whole document. Updated *Figure 201: Dual-channel DAC block diagram*. Added *Section 25.3: DAC implementation*. *Section : Sample and Hold mode:* – Replaced SAMx by TSAMPEx. – Replaced C_L by C_{SH} – Replaced TSAMx by TSAMPLEx – Replaced TREFx by TREFRESHx – Replaced $t_{sampling}$ by $t_{SAMP.}$ – Replaced tstab-BON and tstab-BOFF by 7 µs and 3 µs, respectively. – Replace Tofftrimmax by t_{TRIM} *DAC status register (DAC_SR)*: for DAC1RDY and DAC1RDYbits, replaced DAC_CH1 and DAC_CH2 by DAC channel 1 and DAC channel 2, respectively. ***Section 29: Digital filter for sigma delta modulators (DFSDM)*** Updated *Section 29.4.3: DFSDM reset and clocks* and *Section 29.4.4: Serial channel transceivers*. ***Section 30: Digital camera interface (DCMI)*** Added 10-12-14-bit progressive video format in *Section 30.5.1: Data formats*. ***Section 33: True random number generator (RNG)*** BYP removed in the register description and map register. ***Section 35: Hash processor (HASH)*** In the whole document: – Replaced SHA1, SHA224, SHA256 by SHA-1, SHA-224, SHA-256 – Replaced HASH_Hx by HASH_HRx Updated *Section 35.1: Introduction* Updated *Section 35.2: HASH main features* Updated *Section 35.3.3: About secure hash algorithms* ***Section 37: Advanced-control timers (TIM1/TIM8)*** Updated procedure in *Section : External clock source mode 1*, *Section 37.3.7: Input capture mode*, *Section 37.3.8: PWM input mode* and *Section 37.3.20: One-pulse mode*. Updated *Section 37.3.16: Using the break function*, including *Figure 386: Break and Break2 circuitry overview*. ***Section 38: General-purpose timers (TIM2/TIM3/TIM4/TIM5)*** Added *TIM4 timer input selection register (TIM4_TISEL)*.

Table 597. Document revision history (continued)

Date	Revision	Changes
15-Jun-2018	4 (continued)	***Section 40: General-purpose timers (TIM15/TIM16/TIM17)*** TI1SEL[3:0] description updated for 0001 configuration in *Section 40.6.19: TIM17 input selection register (TIM17_TISEL)*. ***Section 42: Low-power timer (LPTIM)*** Added *Section 42.4.16: Debug mode*. ***Section 43: System window watchdog (WWDG)*** Updated *Figure 530: Watchdog block diagram* and *Figure 531: Window watchdog timing diagram*. Updated *Section 43.3.3: Enabling the watchdog* and *Section 43.3.7: Debug mode*. Updated *Table 360: WWDG register map and reset values*. ***Section 46: Inter-integrated circuit (I2C) interface*** Updated *Figure 536: I2C block diagram*. Updated *Section 46.4.4: I2C initialization* removing the reference to RCC and modified *Figure 538: Setup and hold timings*. Updated *Table 383: Effect of low-power modes on the I2C*. Updated *Section 46.6: I2C interrupts*. Updated note in *Section : Master communication initialization (address phase)*. Updated START bit 13 description in *Section 46.7.2: I2C2 control register 2 (I2C_CR2)*. ***Section 47: Universal synchronous receiver transmitter (USART) /universal asynchronous receiver transmitter (UART)*** Section title updated to include UART Added DWU formula in *Section 47.5.8: Tolerance of the USART receiver to clock deviation*. Updated *Section 47.5.21: USART low-power management* Added *Section : Determining the maximum USART baud rate that allows to correctly wake up the microcontroller from low-power mode*. Replaced undocumented w_r0 access type by w access type in *Section 47.7.7: USART request register (USART_RQR)*. Updated notes related to reserved bit/bitfield depending on USART feature in *USART control register 1 (USART_CR1)*, *USART control register 2 (USART_CR2)*, *USART control register 3 (USART_CR3)*, *USART interrupt and status register (USART_ISR)* and *USART interrupt flag clear register (USART_ICR)*.

Table 597. Document revision history (continued)

Date	Revision	Changes
15-Jun-2018	4 (continued)	***Section 48: Low-power universal asynchronous receiver transmitter (LPUART)*** Updated *Section : Determining the maximum LPUART baud rate that allows to correctly wake up the MCU from low-power mode*. *Transmit data register (LPUART_TDR)* and *Receive data register (LPUART_RDR)* reset value changed from undefined to 0x0000 0000. Updated notes related to reserved bit/bitfield depending on LPUART feature in *Control register 3 (LPUART_CR3)* and *Interrupt and status register (LPUART_ISR)*. ***Section 50: Serial audio interface (SAI)*** Updated *Section : Clock generator programming in SPDIF generator mode*. ***Section 51: SPDIF receiver interface (SPDIFRX)*** Suppressed references to spdifrx_symb_ck in the whole section. Removed SPDIRX_VERR/IDR/SIDR registers. ***Section 55: FD controller area network (FDCAN)*** Changed $f_{dcan_ker_ck}$ max value from 500 MHz to 100 MHz. Major section update. ***Section 56: USB on-the-go high-speed (OTG_HS)*** Major section update. ***Section 57: Ethernet (ETH): media access control (MAC) with DMA controller*** Updated Rx flow control sequence in *Section : Receive flow control*. Removed column “Behavior of the sbd_perch_tx_intr_o[] and sbd_perch_rx_intr_o[]” in *Table 523: Transfer complete interrupt behavior*. Remove all references to non-user internal signals in register descriptions. Updated SARC[2:0] configurations in *Operating mode configuration register (ETH_MACCR)*. Updated DZPQ bit description in *Tx Queue flow control register (ETH_MACQTXFCR)*. Changed bit 21 to reserved in *LPI control status register (ETH_MACLCSR)*. Changed bits 1:2 to reserved in *Interrupt enable register (ETH_MACIER)*. Changed bits 1, 2 and 15 to reserved in *Interrupt status register (ETH_MACISR)*. Changed bit 28 to reserved in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*. Removed ETH_MTLRxQ1OMR register description and updated *Rx queue operating mode register (ETH_MTLRXQOMR)* to support both Queue 0 and 1. Changed bits 20:16 to reserved in *MMC Rx interrupt mask register (MMC_RX_INTERRUPT_MASK)*.

Table 597. Document revision history (continued)

Date	Revision	Changes
15-Jun-2018	4 (continued)	***Section 59: Debug infrastructure*** Updated *Section 59.5.1: System ROM tables*. Renamed CSTF_TYPID register into CSTF_DEVTYPE. Updated CSTF_CTRL and CSTF_PRIORTY priority in *Table 583: CSTF register map and reset values*. Renamed MAXNUM bit into MUXNUM in *TPIU device configuration register (TPIU_DEVID)*. Renamed SWTF_TYPEID register into SWTF_DEVTYPE. Updated *DBGMCU identity code register (DBGMCU_IDC)* reset value and added revision X. Updated *ITM trace privilege registers (M7_ITM_TPR)*. Removed M7_ETM_DEVID register. Updated *Debug port target identification register (DP_TARGETID)*. DBGMCU_APB1HFZ1 removed.
29-Jun-2018	5	Added STM32H750xB microcontrollers: – Update cover page including *Related documents* – Updated *Table 2: Bus-master-to-bus-slave interconnect* – Updated *Figure 1: System architecture for STM32H743/53xx and STM32H750xB devices* Updated *Section 3.3.4: Flash memory architecture and usage*. Specified that parallel operations are not possible on STM32H750xB devices in *Section 3.3.11: FLASH parallel operations (STM32H743/753 devices only)*. Specified that bank swapping is not available on STM32H750xB devices in *Section 3.3.13: Flash bank and register swapping (STM32H743/753 devices only)* and notes added for bit SWAP_BANK and SWPA_BANK_OPT in FLASH_OPTCR and FLASH_OPTCR_PRG register, respectively. *Section 57: Ethernet (ETH): media access control (MAC) with DMA controller*: – Added *Figure 787: Supported PHY interfaces*. – Updated *Remote wakeup packet filter register (ETH_MACRWKPFR)* description to replace wkuppktfilter_reg by ETH_MACRWKPFR and restrict the number of filters to 4.

Index

A

ADC_AWD2CR 1004
ADC_AWD3CR 1005
ADC_CALFACT 1008
ADC_CALFACT2 1008
ADC_CFGR 986
ADC_CFGR2 990
ADC_CR 981
ADC_DIFSEL 1007
ADC_DR 1000
ADC_HTR1 995
ADC_HTR2 1006
ADC_IER 979
ADC_ISR 977
ADC_JDRy 1004
ADC_JSQR 1001
ADC_LTR1 994
ADC_LTR2 1005
ADC_LTR3 1006
ADC_OFRy 1003
ADC_PCSEL 994
ADC_SMPR1 992
ADC_SMPR2 993
ADC_SQR1 996
ADC_SQR2 997
ADC_SQR3 998
ADC_SQR4 999
ADCx_CCR 1012
ADCx_CDR 1015
ADCx_CDR2 1015
ADCx_CSR 1010
ADCx_HTR3 1007
AP_BASE 3021
AP_CSW 3020
AP_IDR 3022
AXI_COMP_ID_0 109
AXI_COMP_ID_1 110
AXI_COMP_ID_2 110
AXI_COMP_ID_3 111
AXI_INIx_FN_MOD 115
AXI_INIx_FN_MOD_AHB 114
AXI_INIx_FN_MOD2 113
AXI_INIx_READ_QOS 114
AXI_INIx_WRITE_QOS 115
AXI_PERIPH_ID_0 107
AXI_PERIPH_ID_1 108
AXI_PERIPH_ID_2 108
AXI_PERIPH_ID_3 109
AXI_PERIPH_ID_4 107
AXI_TARGx_FN_MOD 113
AXI_TARGx_FN_MOD_ISS_BM 111
AXI_TARGx_FN_MOD_LB 112
AXI_TARGx_FN_MOD2 112

B

BDMA_CCRx 662
BDMA_CM0ARx 667
BDMA_CM1ARx 668
BDMA_CNDTRx 666
BDMA_CPARx 666
BDMA_IFCR 661
BDMA_ISR 658

C

CEC_CFGR 2990
CEC_CR 2989
CEC_IER 2995
CEC_ISR 2993
CEC_RXDR 2993
CEC_TXDR 2993
COMP_CFGR1 1072
COMP_CFGR2 1074
COMP_ICFR 1070
COMP_OR 1071
COMP_SR 1070
CRC_CR 774
CRC_DR 773
CRC_IDR 773
CRC_INIT 774
CRC_POL 775
CRS_CFGR 489
CRS_CR 488
CRS_ICR 492
CRS_ISR 490
CRYP_CR 1298
CRYP_CSGCMCCMxR 1309
CRYP_CSGCMxR 1310
CRYP_DIN 1300
CRYP_DMACR 1302
CRYP_DOUT 1301
CRYP_IMSCR 1302
CRYP_IV0LR 1308
CRYP_IV0RR 1308
CRYP_IV1LR 1309
CRYP_IV1RR 1309

CRYP_K0LR 1304
CRYP_K0RR 1305
CRYP_K1LR 1305
CRYP_K1RR 1306
CRYP_K2LR 1306
CRYP_K2RR 1306
CRYP_K3LR 1307
CRYP_K3RR 1307
CRYP_MISR 1303
CRYP_RISR 1303
CRYP_SR 1300
CSTF_AUTHSTAT 3061
CSTF_CIDR0 3065
CSTF_CIDR1 3065
CSTF_CIDR2 3066
CSTF_CIDR3 3066
CSTF_CLAIMCLR 3060
CSTF_CLAIMSET 3059
CSTF_CTRL 3058
CSTF_DEVID 3062
CSTF_DEVTYPE 3062
CSTF_LAR 3060
CSTF_LSR 3061
CSTF_PIDR0 3063
CSTF_PIDR1 3064
CSTF_PIDR2 3064
CSTF_PIDR3 3065
CSTF_PIDR4 3063
CSTF_PRIORITY 3059
CTI_APPCLEAR 3043
CTI_APPPULSE 3044
CTI_APPSET 3043
CTI_AUTHSTAT 3050
CTI_CHINSTS 3047
CTI_CHOUTSTS 3047
CTI_CIDR0 3054
CTI_CIDR1 3054
CTI_CIDR2 3055
CTI_CIDR3 3055
CTI_CLAIMCLR 3049
CTI_CLAIMSET 3048
CTI_CONTROL 3042
CTI_DEVID 3051
CTI_DEVTYPE 3051
CTI_GATE 3048
CTI_INENx 3044
CTI_INTACK 3042
CTI_LAR 3049
CTI_LSR 3050
CTI_OUTENx 3045
CTI_PIDR0 3052
CTI_PIDR1 3053
CTI_PIDR2 3053
CTI_PIDR3 3054
CTI_PIDR4 3052
CTI_TRGISTS 3046
CTI_TRGOSTS 3046

D

DAC_CCR 1051
DAC_CR 1040
DAC_DHR12L1 1044
DAC_DHR12L2 1046
DAC_DHR12LD 1047
DAC_DHR12R1 1044
DAC_DHR12R2 1045
DAC_DHR12RD 1047
DAC_DHR8R1 1045
DAC_DHR8R2 1046
DAC_DHR8RD 1048
DAC_DOR1 1048
DAC_DOR2 1049
DAC_MCR 1051
DAC_SHHR 1054
DAC_SHRR 1054
DAC_SHSR1 1053
DAC_SHSR2 1053
DAC_SR 1049
DAC_SWTRGR 1043
DBGMCU_APB1LFZ1 3134
DBGMCU_APB2FZ1 3135
DBGMCU_APB3FZ1 3133
DBGMCU_APB4FZ1 3136
DBGMCU_CR 3132
DBGMCU_IDC 3131
DCMI_CR 1166
DCMI_CWSIZE 1177
DCMI_CWSTRT 1177
DCMI_DR 1178
DCMI_ESCR 1175
DCMI_ESUR 1176
DCMI_ICR 1174
DCMI_IER 1172
DCMI_MIS 1173
DCMI_RIS 1171
DCMI_SR 1170
DFSDM_CHyAWSCDR 1128
DFSDM_CHyCFGR1 1125
DFSDM_CHyCFGR2 1127
DFSDM_CHyDATINR 1129
DFSDM_CHyWDATR 1129
DFSDM_FLTxAWCFR 1142
DFSDM_FLTxAWHTR 1140

DFSDM_FLTxAWLTR 1140
DFSDM_FLTxAWSR 1141
DFSDM_FLTxCNVTIMR 1143
DFSDM_FLTxCR1 1130
DFSDM_FLTxCR2 1133
DFSDM_FLTxEXMAX 1142
DFSDM_FLTxEXMIN 1143
DFSDM_FLTxFCR 1137
DFSDM_FLTxICR 1136
DFSDM_FLTxISR 1134
DFSDM_FLTxJCHGR 1137
DFSDM_FLTxJDATAR 1138
DFSDM_FLTxRDATAR 1139
DLYB_CFGR 892
DLYB_CR 892
DMA_HIFCR 637
DMA_HISR 636
DMA_LIFCR 637
DMA_LISR 635
DMA_SxCR 638
DMA_SxFCR 643
DMA_SxM0AR 642
DMA_SxM1AR 642
DMA_SxNDTR 641
DMA_SxPAR 642
DMA2D_AMTCR 726
DMA2D_BGCMAR 721
DMA2D_BGCOLR 720
DMA2D_BGMAR 714
DMA2D_BGOR 714
DMA2D_BGPFCCR 718
DMA2D_CR 709
DMA2D_FGCMAR 720
DMA2D_FGCOLR 717
DMA2D_FGMAR 713
DMA2D_FGOR 713
DMA2D_FGPFCCR 715
DMA2D_IFCR 712
DMA2D_ISR 711
DMA2D_LWR 725
DMA2D_NLR 725
DMA2D_OCOLR 722
DMA2D_OMAR 724
DMA2D_OOR 724
DMA2D_OPFCCR 721
DMAMUX1_CFR 685
DMAMUX1_CSR 684
DMAMUX1_CxCR 682
DMAMUX1_RGCFR 689
DMAMUX1_RGSR 688
DMAMUX1_RGxCR 686
DMAMUX2_CFR 685
DMAMUX2_CSR 684
DMAMUX2_CxCR 683
DMAMUX2_RGCFR 689
DMAMUX2_RGSR 688
DMAMUX2_RGxCR 686
DP_ABORT 3010
DP_CTRL/STAT 3011
DP_DLCR 3013
DP_DLPIDR 3014
DP_DPIDR 3009
DP_RDBUFF 3016
DP_RESEND 3015
DP_SELECT 3015
DP_TARGETID 3014
DP_TARGETSEL 3017

E

ETF_AUTHSTAT 3083
ETF_BUFWM 3077
ETF_CBUFLVL 3076
ETF_CIDR0 3086
ETF_CIDR1 3087
ETF_CIDR2 3087
ETF_CIDR3 3087
ETF_CLAIMCLR 3081
ETF_CLAIMSET 3081
ETF_CTL 3074
ETF_DEVID 3083
ETF_DEVTYPE 3084
ETF_FFCR 3078
ETF_FFSR 3077
ETF_LAR 3082
ETF_LBUFLVL 3076
ETF_LSR 3082
ETF_MODE 3075
ETF_PIDR0 3085
ETF_PIDR1 3085
ETF_PIDR2 3085
ETF_PIDR3 3086
ETF_PIDR4 3084
ETF_PSCR 3080
ETF_RRD 3072
ETF_RRP 3072
ETF_RSZ 3070
ETF_RWD 3075
ETF_RWP 3073
ETF_STS 3071
ETF_TRG 3073
ETH_DMACCARxBR 2859
ETH_DMACCARxDR 2859
ETH_DMACCATxBR 2859

ETH_DMACCATxDR 2858
ETH_DMACCR 2845
ETH_DMACIER 2854
ETH_DMACMFCR 2863
ETH_DMACRxCR 2848
ETH_DMACRxDLAR 2851
ETH_DMACRxDTPR 2853
ETH_DMACRxIWTR 2858
ETH_DMACRxRLR 2854
ETH_DMACSR 2860
ETH_DMACTxCR 2846
ETH_DMACTxDLAR 2850
ETH_DMACTxDTPR 2852
ETH_DMACTxRLR 2853
ETH_DMADSR 2844
ETH_DMAISR 2844
ETH_DMAMR 2841
ETH_DMASBMR 2843
ETH_MAC1USTCR 2912
ETH_MACA0HR 2922
ETH_MACACR 2955
ETH_MACARPAR 2922
ETH_MACATSNR 2956
ETH_MACATSSR 2957
ETH_MACAxHR 2923
ETH_MACAxLR 2923
ETH_MACCR 2879
ETH_MACDR 2913
ETH_MACECR 2885
ETH_MACHT0R 2891
ETH_MACHT1R 2892
ETH_MACHWF1R 2914
ETH_MACHWF2R 2917
ETH_MACIER 2903
ETH_MACISR 2901
ETH_MACIVIR 2897
ETH_MACL3A00R 2938
ETH_MACL3A01R 2943
ETH_MACL3A10R 2938
ETH_MACL3A11R 2944
ETH_MACL3A20 2939
ETH_MACL3A21R 2944
ETH_MACL3A30 2939
ETH_MACL3A31R 2945
ETH_MACL3L4C0R 2935
ETH_MACL3L4C1R 2940
ETH_MACL4A0R 2937
ETH_MACL4A1R 2942
ETH_MACLCSR 2909
ETH_MACLETR 2912
ETH_MACLMIR 2968
ETH_MACLTCR 2911
ETH_MACMDIOAR 2918
ETH_MACMDIODR 2921
ETH_MACPCSR 2905
ETH_MACPFR 2887
ETH_MACPOCR 2966
ETH_MACPPSCR 2960, 2962
ETH_MACPPSIR 2964
ETH_MACPPSTTNR 2964
ETH_MACPPSTTSR 2963
ETH_MACPPSWR 2965
ETH_MACQ0TxFCR 2898
ETH_MACRWKPFR 2907
ETH_MACRxFCR 2900
ETH_MACRxTxSR 2903
ETH_MACSPI0R 2966
ETH_MACSPI1R 2967
ETH_MACSPI2R 2967
ETH_MACSSIR 2948
ETH_MACSTNR 2950
ETH_MACSTNUR 2951
ETH_MACSTSR 2950
ETH_MACSTSUR 2951
ETH_MACTSAR 2952
ETH_MACTSCR 2945
ETH_MACTSEACR 2958
ETH_MACTSECNR 2959
ETH_MACTSIACR 2957
ETH_MACTSICNR 2958
ETH_MACTSSR 2952
ETH_MACTxTSSNR 2954
ETH_MACTxTSSSR 2954
ETH_MACVHTR 2895
ETH_MACVIR 2896
ETH_MACVR 2913
ETH_MACVTR 2893
ETH_MACWTR 2890
ETH_MTLISR 2868
ETH_MTLOMR 2867
ETH_MTLQICSR 2872
ETH_MTLRxQDR 2875
ETH_MTLRxQMPOCR 2875
ETH_MTLRxQOMR 2873
ETH_MTLTxQDR 2870
ETH_MTLTxQOMR 2868
ETH_MTLTxQUR 2870
EXTI_EMR 762, 764
EXTI_FTSR 752, 756, 759
EXTI_IMR 753-754, 757-758, 760-763, 765
EXTI_PR 763-764, 766
EXTI_RTSR 752, 755, 759
EXTI_SWIER 753, 756, 760

F

FCCAN_CCU_CCFG 2556
FCCAN_CCU_CREL 2556
FCCAN_CCU_CSTAT 2558
FCCAN_CCU_CWD 2558
FCCAN_CCU_IE 2560
FCCAN_CCU_IR 2559
FDCAN_ TXBCIE 2528
FDCAN_CCCR 2495
FDCAN_CREL 2492
FDCAN_DBTP 2492
FDCAN_ECR 2500
FDCAN_ENDN 2492
FDCAN_GFC 2512
FDCAN_HPMS 2515
FDCAN_IE 2507
FDCAN_ILE 2511
FDCAN_ILS 2510
FDCAN_IR 2504
FDCAN_NBTP 2497
FDCAN_NDAT1 2515
FDCAN_NDAT2 2516
FDCAN_PSR 2501
FDCAN_RWD 2494
FDCAN_RXBC 2518
FDCAN_RXESC 2521
FDCAN_RXF0A 2518
FDCAN_RXF0C 2516
FDCAN_RXF0S 2517
FDCAN_RXF1A 2521
FDCAN_RXF1C 2519
FDCAN_RXF1S 2520
FDCAN_SIDFC 2513
FDCAN_TDCR 2504
FDCAN_TEST 2493
FDCAN_TOCC 2499
FDCAN_TOCV 2500
FDCAN_TSCC 2498
FDCAN_TSCV 2498
FDCAN_TTCPT 2548
FDCAN_TTCSM 2548
FDCAN_TTCTC 2547
FDCAN_TTGTP 2537
FDCAN_TTIE 2541
FDCAN_TTILS 2543
FDCAN_TTIR 2539
FDCAN_TTLGT 2547
FDCAN_TTMLM 2533
FDCAN_TTOCF 2532
FDCAN_TTOCN 2536
FDCAN_TTOST 2544
FDCAN_TTRMC 2531
FDCAN_TTTMC 2530
FDCAN_TTTMK 2538
FDCAN_TTTS 2549
FDCAN_TURCF 2534
FDCAN_TURNA 2546
FDCAN_TXBAR 2526
FDCAN_TXBC 2522
FDCAN_TXBCF 2527
FDCAN_TXBCR 2526
FDCAN_TXBRP 2525
FDCAN_TXBTIE 2527
FDCAN_TXBTO 2527
FDCAN_TXEFA 2530
FDCAN_TXEFC 2528
FDCAN_TXEFS 2529
FDCAN_TXESC 2524
FDCAN_TXFQS 2523
FDCAN_XIDAM 2514
FDCAN_XIDFC 2513
FLASH_ACR 184
FLASH_BOOT7_CURR 204
FLASH_BOOT7_PRGR 204
FLASH_CCR1 193
FLASH_CCR2 216
FLASH_CR1 185
FLASH_CR2 209
FLASH_CRCCR1 205
FLASH_CRCCR2 221
FLASH_CRCDATAR 207
FLASH_CRCEADD1R 207
FLASH_CRCEADD2R 223
FLASH_CRCSADD1R 206
FLASH_CRCSADD2R 222
FLASH_ECC_FA1R 208
FLASH_ECC_FA2R 223
FLASH_KEYR1 184
FLASH_KEYR2 208
FLASH_OPTCCR 200
FLASH_OPTCR 194
FLASH_OPTKEYR 185
FLASH_OPTSR_CUR 195
FLASH_OPTSR_PRG 198
FLASH_PRAR_CUR1 200
FLASH_PRAR_CUR2 217
FLASH_PRAR_PRG1 201
FLASH_PRAR_PRG2 218
FLASH_SCAR_CUR1 202
FLASH_SCAR_CUR2 218
FLASH_SCAR_PRG1 202
FLASH_SCAR_PRG2 219
FLASH_SR1 190
FLASH_SR2 213

FLASH_WPSN_CUR1R 203
FLASH_WPSN_CUR2R 220
FLASH_WPSN_PRG1R 204
FLASH_WPSN_PRG2R 220
FMC_BCRx 818
FMC_BTRx 822
FMC_BWTRx 825
FMC_ECCR 838
FMC_PATT 837
FMC_PCR 833
FMC_PMEM 836
FMC_SDCMR 853
FMC_SDCRx 850
FMC_SDRTR 854
FMC_SDSR 856
FMC_SDTRx 851
FMC_SR 834

G

GPIOx_AFRH 523
GPIOx_AFRL 522
GPIOx_BSRR 520
GPIOx_IDR 520
GPIOx_LCKR 521
GPIOx_MODER 518
GPIOx_ODR 520
GPIOx_OSPEEDR 519
GPIOx_OTYPER 518
GPIOx_PUPDR 519

H

HASH_CR 1327
HASH_CSR0 1337
HASH_CSRx 1337
HASH_DIN 1330
HASH_HR0 1332
HASH_HR1 1332
HASH_HR2 1333
HASH_HR3 1333
HASH_HR4 1333
HASH_HR5 1334
HASH_HR6 1334
HASH_HR7 1334
HASH_IMR 1335
HASH_SR 1336
HASH_STR 1331
HRTIM_ADC1R 1492
HRTIM_ADC2R 1493
HRTIM_ADC3R 1494
HRTIM_ADC4R 1496
HRTIM_BDMADR 1504
HRTIM_BDMUPR 1502
HRTIM_BDTxUPR 1503
HRTIM_BMCMPR 1487
HRTIM_BMCR 1483
HRTIM_BMPER 1487
HRTIM_BMTRGR 1485
HRTIM_CHPxR 1464
HRTIM_CMP1CxR 1449
HRTIM_CMP1xR 1448
HRTIM_CMP2xR 1449
HRTIM_CMP3xR 1450
HRTIM_CMP4xR 1450
HRTIM_CNTxR 1447
HRTIM_CPT1xCR 1466
HRTIM_CPT1xR 1451
HRTIM_CPT2xCR 1467
HRTIM_CPT2xR 1451
HRTIM_CR1 1474
HRTIM_CR2 1476
HRTIM_DTxR 1452
HRTIM_EECR1 1488
HRTIM_EECR2 1490
HRTIM_EECR3 1491
HRTIM_EEFxR1 1458
HRTIM_EEFxR2 1460
HRTIM_FLTINR1 1498
HRTIM_FLTINR2 1500
HRTIM_FLTxR 1473
HRTIM_ICR 1478
HRTIM_IER 1479
HRTIM_ISR 1477
HRTIM_MCMP1R 1434
HRTIM_MCMP2R 1435
HRTIM_MCMP3R 1435
HRTIM_MCMP4R 1436
HRTIM_MCNTR 1433
HRTIM_MCR 1426
HRTIM_MDIER 1431
HRTIM_MICR 1430
HRTIM_MISR 1429
HRTIM_MPER 1433
HRTIM_MREP 1434
HRTIM_ODISR 1481
HRTIM_ODSR 1482
HRTIM_OENR 1480
HRTIM_OUTxR 1470
HRTIM_PERxR 1447
HRTIM_REPxR 1448
HRTIM_RSTAR 1461
HRTIM_RSTBR 1463
HRTIM_RSTCR 1463
HRTIM_RSTDR 1463

HRTIM_RSTER 1464
HRTIM_RSTx1R 1456
HRTIM_RSTx2R 1457
HRTIM_SETx1R 1454
HRTIM_SETx2R 1456
HRTIM_TIMxCR 1437
HRTIM_TIMxDIER 1444
HRTIM_TIMxICR 1443
HRTIM_TIMxISR 1441
HSEM_CnICR 504
HSEM_CnIER 504
HSEM_CnISR 504
HSEM_CnMISR 505
HSEM_CR 505
HSEM_KEYR 506
HSEM_RLRx 503
HSEM_Rx 502

I

I2C_CR1 1986
I2C_CR2 1989
I2C_ICR 1998
I2C_ISR 1996
I2C_OAR1 1992
I2C_OAR2 1993
I2C_PECR 1999
I2C_RXDR 2000
I2C_TIMEOUTR 1995
I2C_TIMINGR 1994
I2C_TXDR 2000
I2Cx_CR2 1989
IWDG_KR 1883
IWDG_PR 1884
IWDG_RLR 1885
IWDG_SR 1886
IWDG_WINR 1887

J

JPEG_CFR 1226
JPEG_CONFR0 1221
JPEG_CONFR1 1221
JPEG_CONFR2 1222
JPEG_CONFR3 1223
JPEG_CONFR4-7 1223
JPEG_CR 1224
JPEG_DIR 1227
JPEG_DOR 1227
JPEG_SR 1225

L

LPTIM_ARR 1868
LPTIM_CFGR 1863
LPTIM_CFGR2 1869
LPTIM_CMP 1868
LPTIM_CNT 1869
LPTIM_CR 1866
LPTIM_ICR 1861
LPTIM_IER 1862
LPTIM_ISR 1859
LPTIM3_CFGR2 1870
LPUART_BRR 2118
LPUART_CR1 2110
LPUART_CR2 2113
LPUART_CR3 2115
LPUART_ICR 2123
LPUART_ISR 2119
LPUART_PRESC 2125
LPUART_RDR 2124
LPUART_RQR 2119
LPUART_TDR 2125
LTDC_AWCR 1194
LTDC_BCCR 1197
LTDC_BPCR 1193
LTDC_CDSR 1201
LTDC_CPSR 1200
LTDC_GCR 1195
LTDC_ICR 1199
LTDC_IER 1198
LTDC_ISR 1199
LTDC_LIPCR 1200
LTDC_LxBFCR 1208
LTDC_LxCACR 1206
LTDC_LxCFBAR 1209
LTDC_LxCFBLNR 1210
LTDC_LxCFBLR 1209
LTDC_LxCKCR 1205
LTDC_LxCLUTWR 1211
LTDC_LxCR 1202
LTDC_LxDCCR 1207
LTDC_LxPFCR 1205
LTDC_LxWHPCR 1203
LTDC_LxWVPCR 1204
LTDC_SRCR 1197
LTDC_SSCR 1193
LTDC_TWCR 1195

M

M7_CPUROM_CIDR0 3143
M7_CPUROM_CIDR1 3144
M7_CPUROM_CIDR2 3144

M7_CPUROM_CIDR3 3145
M7_CPUROM_MEMTYPE 3141
M7_CPUROM_PIDR0 3142
M7_CPUROM_PIDR1 3142
M7_CPUROM_PIDR2 3142
M7_CPUROM_PIDR3 3143
M7_CPUROM_PIDR4 3141
M7_DWT_CIDR0 3161
M7_DWT_CIDR1 3161
M7_DWT_CIDR2 3161
M7_DWT_CIDR3 3162
M7_DWT_COMPx 3156
M7_DWT_CPICNT 3154
M7_DWT_CTRL 3152
M7_DWT_CYCCNT 3154
M7_DWT_EXCCNT 3155
M7_DWT_FOLDCNT 3156
M7_DWT_FUNCTx 3157
M7_DWT_LSUCNT 3155
M7_DWT_MASKx 3157
M7_DWT_PCSR 3156
M7_DWT_PIDR0 3159
M7_DWT_PIDR1 3159
M7_DWT_PIDR2 3160
M7_DWT_PIDR3 3160
M7_DWT_PIDR4 3158
M7_DWT_SLPCNT 3155
M7_ETM_AUTHSTAT 3203
M7_ETM_CCCTL 3187
M7_ETM_CIDR0 3207
M7_ETM_CIDR1 3207
M7_ETM_CIDR2 3208
M7_ETM_CIDR3 3208
M7_ETM_CLAIMCLR 3202
M7_ETM_CLAIMSET 3201
M7_ETM_CNTRLDV 3190
M7_ETM_CONFIG 3183
M7_ETM_DEVARCH 3204
M7_ETM_DEVTYPE 3204
M7_ETM_EVENTCTL0 3184
M7_ETM_EVENTCTL1 3184
M7_ETM_IDR0 3193
M7_ETM_IDR1 3194
M7_ETM_IDR10 3191
M7_ETM_IDR11 3191
M7_ETM_IDR12 3192
M7_ETM_IDR13 3192
M7_ETM_IDR2 3194
M7_ETM_IDR3 3195
M7_ETM_IDR4 3196
M7_ETM_IDR5 3196
M7_ETM_IDR8 3190
M7_ETM_IDR9 3191
M7_ETM_IMSPEC0 3192
M7_ETM_LAR 3202
M7_ETM_LSR 3203
M7_ETM_PDC 3200
M7_ETM_PDS 3201
M7_ETM_PIDR0 3205
M7_ETM_PIDR1 3206
M7_ETM_PIDR2 3206
M7_ETM_PIDR3 3207
M7_ETM_PIDR4 3205
M7_ETM_PRGCTL 3181
M7_ETM_PROCSEL 3182
M7_ETM_RSCTL2 3197
M7_ETM_RSCTL3 3198
M7_ETM_SSCC0 3198
M7_ETM_SSCS0 3199
M7_ETM_SSPCIC0 3200
M7_ETM_STALLCTL 3185
M7_ETM_STAT 3182
M7_ETM_SYNCP 3186
M7_ETM_TRACEID 3187
M7_ETM_TSCTL 3186
M7_ETM_VICTL 3188
M7_ETM_VIPCSSCTL 3189
M7_ETM_VISSCTL 3189
M7_FPB_CIDR0 3178
M7_FPB_CIDR1 3178
M7_FPB_CIDR2 3179
M7_FPB_CIDR3 3179
M7_FPB_COMPx 3175
M7_FPB_CTRL 3173
M7_FPB_PIDR0 3176
M7_FPB_PIDR1 3176
M7_FPB_PIDR2 3177
M7_FPB_PIDR3 3177
M7_FPB_PIDR4 3175
M7_FPB_REMAP 3174
M7_ITM_CIDR0 3170
M7_ITM_CIDR1 3171
M7_ITM_CIDR2 3171
M7_ITM_CIDR3 3172
M7_ITM_PIDR0 3168
M7_ITM_PIDR1 3169
M7_ITM_PIDR2 3169
M7_ITM_PIDR3 3170
M7_ITM_PIDR4 3168
M7_ITM_STIMx 3165
M7_ITM_TCR 3166
M7_ITM_TER 3165
M7_ITM_TPR 3166
M7_PPBROM_CIDR0 3149

M7_PPBROM_CIDR1 3149
M7_PPBROM_CIDR2 3150
M7_PPBROM_CIDR3 3150
M7_PPBROM_MEMTYPE 3146
M7_PPBROM_PIDR0 3147
M7_PPBROM_PIDR1 3147
M7_PPBROM_PIDR2 3148
M7_PPBROM_PIDR3 3148
M7_PPBROM_PIDR4 3146
MDIOS_CLRFR 2346
MDIOS_CR 2342
MDIOS_CRDFR 2344
MDIOS_CWRFR 2343
MDIOS_DINRx 2347
MDIOS_DOUTRx 2347
MDIOS_RDFR 2344
MDIOS_SR 2345
MDIOS_WRFR 2343
MDMA_CxBNDTR 605
MDMA_CxBRUR 609
MDMA_CxCR 600
MDMA_CxDAR 607
MDMA_CxESR 598
MDMA_CxIFCR 597
MDMA_CxISR 596
MDMA_CxLAR 610
MDMA_CxMAR 612
MDMA_CxMDR 612
MDMA_CxSAR 607
MDMA_CxTBR 611
MDMA_CxTCR 602
MDMA_GISR0 595
MMC_CONTROL 2925
MMC_RX_INTERRUPT 2926
MMC_RX_INTERRUPT_MASK 2929
MMC_TX_INTERRUPT 2927
MMC_TX_INTERRUPT_MASK 2930

O

OPAMP_OR 1091
OPAMP1_CSR 1088
OPAMP1_HSOTR 1091
OPAMP1_OTR 1090
OPAMP2_CSR 1091
OPAMP2_HSOTR 1094
OPAMP2_OTR 1093
OTG_CID 2620
OTG_DAINT 2644
OTG_DAINTMSK 2644
OTG_DCFG 2637
OTG_DCTL 2638
OTG_DEACHINT 2647
OTG_DEACHINTMSK 2648
OTG_DIEPCTLx 2651
OTG_DIEPDMAx 2655
OTG_DIEPEMPMSK 2647
OTG_DIEPINTx 2653
OTG_DIEPMSK 2641
OTG_DIEPTSIZ0 2655
OTG_DIEPTSIZx 2656
OTG_DIEPTXF0 2616
OTG_DIEPTXFx 2624
OTG_DOEPCTL0 2657
OTG_DOEPCTLx 2662
OTG_DOEPDMAx 2662
OTG_DOEPINTx 2659
OTG_DOEPMSK 2642
OTG_DOEPTSIZ0 2661
OTG_DOEPTSIZx 2665
OTG_DSTS 2640
OTG_DTHRCTL 2646
OTG_DTXFSTSx 2656
OTG_DVBUSDIS 2645
OTG_DVBUSPULSE 2645
OTG_GAHBCFG 2598
OTG_GCCFG 2618
OTG_GINTMSK 2609
OTG_GINTSTS 2605
OTG_GLPMCFG 2620
OTG_GOTGCTL 2593
OTG_GOTGINT 2596
OTG_GRSTCTL 2603
OTG_GRXFSIZ 2616
OTG_GRXSTSP 2615
OTG_GRXSTSR 2612-2613
OTG_GUSBCFG 2599
OTG_HAINT 2628
OTG_HAINTMSK 2628
OTG_HCCHARx 2631
OTG_HCDMAx 2636
OTG_HCFG 2625
OTG_HCINTMSKx 2634
OTG_HCINTx 2633
OTG_HCSPLTx 2632
OTG_HCTSIZx 2635
OTG_HFIR 2625
OTG_HFNUM 2626
OTG_HNPTXFSIZ 2616
OTG_HNPTXSTS 2617
OTG_HPRT 2629
OTG_HPTXFSIZ 2624
OTG_HPTXSTS 2627
OTG_HS_DIEPEACHMSK1 2648

OTG_HS_DOEPEACHMSK1 2649
OTG_PCGCCTL 2666

P

purpose 1746
PWR_CSR 287, 291-292

Q

QUADSPI _PIR 887
QUADSPI _PSMAR 886
QUADSPI _PSMKR 886
QUADSPI_ABR 885
QUADSPI_AR 884
QUADSPI_CCR 882
QUADSPI_CR 876
QUADSPI_DCR 879
QUADSPI_DLR 881
QUADSPI_DR 885
QUADSPI_FCR 881
QUADSPI_LPTR 887
QUADSPI_SR 880

R

RCC_BDCR 403
RCC_CFGR) 367
RCC_CICR 401
RCC_CIER 397
RCC_CIFR 399
RCC_CR 361
RCC_CRRCR 366
RCC_CSR 405
RCC_D1AHB1ENR 428
RCC_D1AHB1LPENR 450
RCC_D1AHB1RSTR 406
RCC_D1APB1ENR 437
RCC_D1APB1LPENR 459-460
RCC_D1APB1RSTR 413
RCC_D1CCIPR 388
RCC_D1CFGR 370
RCC_D2AHB1ENR 430
RCC_D2AHB1LPENR 452
RCC_D2AHB1RSTR 408
RCC_D2AHB2ENR 432
RCC_D2AHB2LPENR 454
RCC_D2AHB2RSTR 410
RCC_D2APB1HENR 442
RCC_D2APB1HLPENR 464
RCC_D2APB1HRSTR 417
RCC_D2APB1LENR 438
RCC_D2APB1LRSTR 414
RCC_D2APB2ENR 444
RCC_D2APB2LPENR 466
RCC_D2APB2RSTR 418
RCC_D2CCIP1R 389
RCC_D2CCIP2R 392
RCC_D2CFGR 372
RCC_D3AHB1ENR 434
RCC_D3AHB1LPENR 456
RCC_D3AHB1RSTR 411
RCC_D3AMR 423
RCC_D3APB1ENR 447
RCC_D3APB1LPENR 469
RCC_D3APB1RSTR 420
RCC_D3CCIPR 394
RCC_D3CFGR 373
RCC_GCR 422
RCC_HSICFGR 365
RCC_PLL1DIVR 379
RCC_PLL1FRACR 381
RCC_PLL2DIVR 382
RCC_PLL2FRACR 384
RCC_PLL3DIVR 385
RCC_PLL3FRACR 387
RCC_PLLCFGR 376
RCC_PLLCKSELR 374
RCC_RSR 426
RNG_CR 1240
RNG_DR 1242
RNG_SR 1241
RTC_ALRMAR 1919
RTC_ALRMBR 1920
RTC_ALRMBSSR 1931
RTC_BKPxR 1932
RTC_CALR 1926
RTC_CR 1911
RTC_DR 1909
RTC_ISR 1914
RTC_OR 1932
RTC_PRER 1917
RTC_SHIFTR 1922
RTC_SSR 1921
RTC_TR 284-286, 289-290, 1908
RTC_TSDR 1924
RTC_TSSSR 1925
RTC_TSTR 1923
RTC_WPR 1921
RTC_WUTR 1918
RX_ALIGNMENT_ERROR_PACKETS 2932
RX_CRC_ERROR_PACKETS 2932
RX_LPI_TRAN_CNTR 2934
RX_LPI_USEC_CNTR 2934
RX_UNICAST_PACKETS_GOOD 2933

S

SAI_ACLRFR 2262
SAI_ACR1 2241
SAI_ACR2 2247
SAI_ADR 2264
SAI_AFRCR 2251
SAI_AIM 2255
SAI_ASLOTR 2253
SAI_ASR 2258
SAI_BCLRFR 2263
SAI_BCR1 2244
SAI_BCR2 2249
SAI_BDR 2265
SAI_BFRCR 2252
SAI_BIM 2257
SAI_BSLOTR 2254
SAI_BSR 2260
SAI_GCR 2241
SAI_PDMCR 2265
SAI_PDMDLY 2266
SDMMC_ACKTIMER 2421
SDMMC_ARGR 2407
SDMMC_CLKCR 2405
SDMMC_CMDR 2407
SDMMC_DCNTR 2413
SDMMC_DCTRL 2412
SDMMC_DLENR 2411
SDMMC_DTIMER 2410
SDMMC_FIFOR 2422
SDMMC_ICR 2416
SDMMC_IDMABASE0R 2424
SDMMC_IDMABASE1R 2424
SDMMC_IDMABSIZER 2423
SDMMC_IDMACTRLR 2422
SDMMC_MASKR 2419
SDMMC_POWER 2404
SDMMC_RESPCMDR 2409
SDMMC_RESPxR 2410
SDMMC_STAR 2414
SMPMI_IER 2329
SPDIFRX_CR 2293
SPDIFRX_CSR 2303
SPDIFRX_DIR 2303
SPDIFRX_FMT0_DR 2300
SPDIFRX_FMT1_DR 2301
SPDIFRX_FMT2_DR 2302
SPDIFRX_IFCR 2299
SPDIFRX_IMR 2296
SPDIFRX_SR 2297
SPI_CFG1 2187
SPI_CFG2 2190
SPI_CR2 2186
SPI_CRCPOLY 2197
SPI_I2SCFGR 2199
SPI_RXCRC 2198
SPI_TXCRC 2198
SPI_UDRDR 2199
SPI2S_CR1 2185
SPI2S_IER 2192
SPI2S_IFCR 2195
SPI2S_RXDR 2197
SPI2S_SR 2193
SPI2S_TXDR 2196
SWO_AUTHSTAT 3112
SWO_CIDR0 3117
SWO_CIDR1 3117
SWO_CIDR2 3118
SWO_CIDR3 3118
SWO_CLAIMCLR 3111
SWO_CLAIMSET 3110
SWO_CODR 3109
SWO_DEVID 3113
SWO_DEVTYPE 3114
SWO_FFSR 3110
SWO_LAR 3111
SWO_LSR 3112
SWO_PIDR0 3115
SWO_PIDR1 3115
SWO_PIDR2 3116
SWO_PIDR3 3116
SWO_PIDR4 3114
SWO_SPPR 3109
SWPMI_BRR 2326
SWPMI_CR 2325
SWPMI_ICR 2328
SWPMI_ISR 2327
SWPMI_OR 2332
SWPMI_RDR 2331
SWPMI_RFL 2331
SWPMI_TDR 2331
SWTF_AUTHSTAT 3123
SWTF_CIDR0 3127
SWTF_CIDR1 3128
SWTF_CIDR2 3128
SWTF_CIDR3 3129
SWTF_CLAIMCLR 3122
SWTF_CLAIMSET 3121
SWTF_CTRL 3120
SWTF_DEVID 3124
SWTF_DEVTYPE 3124
SWTF_LAR 3122
SWTF_LSR 3123
SWTF_PIDR0 3125
SWTF_PIDR1 3126

SWTF_PIDR2 3126
SWTF_PIDR3 3127
SWTF_PIDR4 3125
SWTF_PRIORITY 3121
SYSCFG_CCCR 536
SYSCFG_CCCSR 535
SYSCFG_CCVR 536
SYSCFG_CFGR 532
SYSCFG_EXTICR1 528
SYSCFG_EXTICR2 529
SYSCFG_EXTICR3 530
SYSCFG_EXTICR4 531
SYSCFG_PKGR 537
SYSCFG_PMCR 526
SYSCFG_UR0 538
SYSCFG_UR10 542
SYSCFG_UR11 543
SYSCFG_UR12 543
SYSCFG_UR13 544
SYSCFG_UR14 545
SYSCFG_UR15 546
SYSCFG_UR16 547
SYSCFG_UR17 547
SYSCFG_UR2 538
SYSCFG_UR3 539
SYSCFG_UR4 539
SYSCFG_UR5 540
SYSCFG_UR6 540
SYSCFG_UR7 541
SYSCFG_UR8 541
SYSCFG_UR9 542
SYSROM_CIDR0 3028
SYSROM_CIDR1 3028
SYSROM_CIDR2 3029
SYSROM_CIDR3 3029
SYSROM_MEMTYPE 3025
SYSROM_PIDR0 3026
SYSROM_PIDR1 3027
SYSROM_PIDR2 3027
SYSROM_PIDR3 3028
SYSROM_PIDR4 3026

T

TIM1_AF1 1604
TIM1_AF2 1606
TIM1_TISEL 1611
TIM12_ARR 1731
TIM12_CCER 1729
TIM12_CCMR1 1725
TIM12_CCR1 1731
TIM12_CCR2 1731
TIM12_CNT 1730
TIM12_CR1 1719
TIM12_CR2 1720
TIM12_DIER 1723
TIM12_EGR 1724
TIM12_PSC 1730
TIM12_SMCR 1721
TIM12_SR 1723
TIM12_TISEL 1732
TIM13_TISEL 1743
TIM14_TISEL 1743
TIM15_AF1 1804
TIM15_ARR 1798
TIM15_BDTR 1800
TIM15_CCER 1795
TIM15_CCMR1 1791
TIM15_CCR1 1799
TIM15_CCR2 1800
TIM15_CNT 1798
TIM15_CR1 1783
TIM15_CR2 1784
TIM15_DCR 1803
TIM15_DIER 1787
TIM15_DMAR 1803
TIM15_EGR 1790
TIM15_PSC 1798
TIM15_RCR 1799
TIM15_SMCR 1786
TIM15_SR 1788
TIM15_TISEL 1805
TIM16_AF1 1824
TIM16_TISEL 1825
TIM17_AF1 1826
TIM17_TISEL 1827
TIM2_AF1 1685
TIM2_TISEL 1687
TIM3_AF1 1685
TIM3_TISEL 1687
TIM4_AF1 1686
TIM4_TISEL 1688
TIM5_AF1 1686
TIM5_TISEL 1689
TIM8_AF1 1607
TIM8_AF2 1609
TIM8_TISEL 1611
TIMx_ARR 1594, 1681, 1742, 1818, 1841
TIMx_BDTR 1597, 1820
TIMx_CCER 1590, 1679, 1740, 1815
TIMx_CCMR1 . . . . . 1584, 1673, 1738, 1813
TIMx_CCMR2 . . . . . 1588, 1677
TIMx_CCMR3 1602
TIMx_CCR1 . . . . . 1595, 1682, 1742, 1819

TIMx_CCR2 . . . . . 1596, 1682
TIMx_CCR3 . . . . . 1596, 1683
TIMx_CCR4 . . . . . 1597, 1683
TIMx_CCR5 1603
TIMx_CCR6 1604
TIMx_CNT 1594, 1680, 1741, 1817, 1840
TIMx_CR1 . . . . . 1573, 1664, 1735, 1808, 1837
TIMx_CR2 . . . . . 1574, 1665, 1809, 1839
TIMx_DCR 1600, 1684, 1822
TIMx_DIER 1579, 1670, 1736, 1810, 1839
TIMx_DMAR 1601, 1684, 1823
TIMx_EGR 1583, 1672, 1737, 1812, 1840
TIMx_PSC 1594, 1681, 1742, 1818, 1841
TIMx_RCR 1595, 1819
TIMx_SMCR 1577, 1667
TIMx_SR 1581, 1671, 1736, 1811, 1840
TPIU_AUTHSTAT 3100
TPIU_CIDR0 3104
TPIU_CIDR1 3105
TPIU_CIDR2 3105
TPIU_CIDR3 3106
TPIU_CLAIMCLR 3098
TPIU_CLAIMSET 3098
TPIU_CURPSIZE 3091
TPIU_CURTPM 3094
TPIU_DEVID 3101
TPIU_DEVTYPE 3101
TPIU_FFCR 3096
TPIU_FFSR 3095
TPIU_FSCR 3097
TPIU_LAR 3099
TPIU_LSR 3099
TPIU_PIDR0 3102
TPIU_PIDR1 3103
TPIU_PIDR2 3103
TPIU_PIDR3 3104
TPIU_PIDR4 3102
TPIU_SUPPSIZE 3091
TPIU_SUPTPM 3093
TPIU_SUPTRGM 3091
TPIU_TPRCR 3095
TPIU_TRGCNT 3092
TPIU_TRGMULT 3093
TSG_CIDR0 3036
TSG_CIDR1 3036
TSG_CIDR2 3037
TSG_CIDR3 3037
TSG_CNTCR 3032
TSG_CNTCVL 3033
TSG_CNTCVU 3033
TSG_CNTFID0 3033
TSG_CNTSR 3032
TSG_PIDR0 3034
TSG_PIDR1 3035
TSG_PIDR2 3035
TSG_PIDR3 3036
TSG_PIDR4 3034
TX_LPI_TRAN_CNTR 2933
TX_LPI_USEC_CNTR 2933
TX_MULTIPLE_COLLISION_GOOD_PACKETS 2931
TX_PACKET_COUNT_GOOD 2931
TX_SINGLE_COLLISION_GOOD_PACKETS . . 2931

U

USART_BRR 2067
USART_CR1 . . . . . 406, 2055
USART_CR2 2059
USART_CR3 2062
USART_GTPR 2067
USART_ICR 2076
USART_ISR 2070
USART_PRESC 2079
USART_RDR 2078
USART_RQR 2069
USART_RTOR 2068
USART_TDR 2078

V

VREFBUF_CCR 1060
VREFBUF_CSR 1059

W

WWDG_CFR 1877
WWDG_CR 1877
WWDG_SR 1878

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved