QingKeV3 Microprocessor Manual

V1.2

Overview

QingKe V3 series microprocessors are self-developed 32-bit general-purpose MCU microprocessors based on standard RISC-V instruction set architecture. This series includes V3A, V3B and V3C, of which V3A supports RV32IMAC standard instruction set extension and V3B/C supports RV32IMCB standard instruction set extension and customized instruction set extension XW. Both of them support single-cycle multiplication and hardware division, in addition to hardware pressure stack (HPE), table-free interrupt (VTF), streamlined 1- and 2-wire debugging interfaces, "WFE" instructions and other special features. In addition, it also supports Hardware Prologue/Epilogue (HPE), Vector Table Free (VTF), streamlined 1-/2-wire debugging interface, and support for "WFE" instruction.

Features

Features	Description
ISA	RV32IM[A]C[B]
Pipeline	3
FPU	Not supported
Branch prediction	Static branch prediction
Interrupt	Support a total of 256 interrupts including exceptions, and supports VTF
HPE	Support 2 levels of HPE
Physical Memory Protection (PMP)	Supported
Low-power consumption mode	Support Sleep and Deep sleep modes, and support WFI and WFE sleep methods
Extended Instruction Set	Supported
Debug	1/2-wire SDI, standard RISC-V debug

Chapter 1 Overview

QingKe V3 series microprocessors include V3A, V3B and V3C, there are some differences between the series according to the application, the specific differences are detailed in Table 1-1.

Table 1-1 Overview of QingKe V3 microprocessor

Feature / Model	ISA	HPE number of levels	Interruptions nesting number of levels	VTF number of channels	Pipeline	Vector table mode	Extended Instruction (XW)	Number of memory protection areas
V3A	RV32IMAC	2	2	4	3	Instruction	×	×
V3B	RV32IMCB	2	2	4	3	Address/ Instruction	√	×
V3C	RV32IMCB	2	2	4	3	Address/ Instruction	√	4

Note: OS task switching generally uses stack push, which are not limited in number of levels

1.1 Instruction Set

QingKe V3 series microprocessors follow the standard RISC-V Instruction Set Architecture (ISA). Detailed documentation of the standard can be found in "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2" on the RISC-V International website. The RISC-V instruction set has a simple architecture and supports a modular design, allowing for flexible combinations based on different needs, and the V3 series supports the following instruction set extensions.

- RV32: 32-bit architecture, general-purpose register bit width of 32 bits
- I: Support shaping operation, with 32 shaping registers
- M: Support shaping multiplication and division instructions
- A: Support atomic commands
- C: Support 16-bit compression instruction
- B: Support for bit manipulation instructions
- XW: 16-bit compression instructions for self-extending byte and halfword operations

Note:

1: The subset of instructions supported by different models may be different, please refer to Table 1-1 for details;

2: In order to further improve the code density, extend the XW subset, add the following compression instructions c.lbu/c.lhu/c.sb/c.sh/c.lbusp/c.lhusp/c.sbsp/c.shsp, use of which needs to be based on the MRS compiler or the toolchain it provides;

3: V3B supports extracting a word (32bit) instruction from a doubleword (64bit) and extracting a word (32bit) instruction from a multiplication result (64bit). The specific usage method can refer to the library function and cooperate with the MRS compiler or the tool chain provided by it;

4: V3B/C supports memory copy instruction. For specific usage, please refer to library function and cooperate with MRS compiler or its tool chain.

1.2 Register Set

The RV32I has 32 register sets from x0-x31. The V3 series does not support the "F" extension, i.e., there is no

floating-point register set. In the RV32, each register is 32 bits. Table 1-2 below lists the registers of RV32I and their descriptions.

Table 1-2 RISC-V registers

Register	ABI Name	Description	Storer
x0	zero	Hardcoded 0	-
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	-
x4	tp	Thread pointer	-
x5-7	t0-2	Temporary register	Caller
x8	s0/fp	Save register/frame pointer	Callee
x9	s1	Save register	Callee
x10-11	a0-1	Function parameters/return values	Caller
x12-17	a2-7	Function parameters	Caller
x18-27	a2-11	Save register	Callee
X28-31	t3-6	Temporary register	Caller

The Caller attribute in the above table means that the called procedure does not save the register value, and the Callee attribute means that the called procedure saves the register.

1.3 Privilege Mode

The standard RISC-V architecture includes three privileged modes: Machine mode, Supervisor mode, and User mode, as shown in Table 1-3 below. The machine mode is a mandatory mode, and the other modes are optional modes. For details, you can refer to "The RISC-V Instruction Set Manual Volume II: Privileged Architecture", which can be downloaded for free from the RISC-V International website.

Table 1-3 RISC-V architecture privilege mode

Code	Name	Abbreviations
0b00	User Mode	U
0b01	Supervisor Model	S
0b10	Reserved	Reserved
0b11	Machine mode	M

QingKe V3 series microprocessors support two of these privileged modes.

- Machine mode

Machine mode has the highest authority, the program in this mode can access all the Control and Status Register (CSR), but also can access all the physical address area. The power-up default is in machine mode, when the execution of mret (Machine mode return instruction) returns, according to the CSR register mstatus (Machine mode status register) in the MPP bit, if MPP = 0b00, then exit the Machine mode into the User mode, MPP = 0b11, then continue to retain the Machine mode.

- User mode

User mode has the lowest privilege, and only limited CSR registers can be accessed in this mode. When an exception or interrupt occurs, the microprocessor goes from User mode to Machine mode to handle exceptions and interrupts.

1.4 CSR Register

A series of CSR registers are defined in the RISC-V architecture to control and record the operating state of the microprocessor. These CSRs can be extended by 4096 registers using an internal dedicated 12-bit address coding space. And use the high two CSR[11:10] to define the read/write permission of this register, 0b00, 0b01, 0b10 for read/write allowed and 0b11 for read only. Use the two bits CSR[9:8] to define the lowest privilege

level that can access this register, and the value corresponds to the privilege mode defined in Table 1-3. The CSR registers implemented in the QingKe V3 microprocessor are detailed in Chapter 8.

Chapter 2 Exception

Exception mechanism, which is a mechanism to intercept and handle "unusual operation events". QingKe V3 series microprocessors are equipped with an exception response system that can handle up to 256 exceptions, including interrupts. When an exception or interruption occurs, the microprocessor can quickly respond and handle the exception and interruption events.

2.1 Exception Types

The hardware behavior of the microprocessor is the same whether an exception or an interrupt occurs. The microprocessor suspends the current program, moves to the exception or interrupt handler, and returns to the previously suspended program when processing is complete. Broadly speaking, interrupts are also part of exceptions. Whether exactly the current occurrence is an interrupt or an exception can be viewed through the Machine mode exception cause register mcause. The mcause[31] is the interrupt field, which is used to indicate whether the cause of the exception is an interrupt or an exception. mcause[31]=1 means interrupt, mcause[31]=0 means exception. mcause[30:0] is the exception code, which is used to indicate the specific cause of the exception or the interrupt number, as shown in the following table.

Table 2-1 V3 microprocessor exception codes

Interrupt	Exception codes	Synchronous / Asynchronous	Reason for exception
1	0-1	-	Reserved
1	2	Precise asynchronous	NMI interrupts
1	3-11	-	Reserved
1	12	Precise asynchronous	SysTick interrupts
1	13	-	Reserved
1	14	Synchronous	Software interrupts
1	15	-	Reserved
1	16-255	Precise asynchronous	External interrupt 16-255
0	0	Synchronous	Instruction address misalignment
0	1	Synchronous	Fetch command access error
0	2	Synchronous	Illegal instructions
0	3	Synchronous	Breakpoints
0	4	Synchronous	Load instruction access address misalignment
0	5	Non- precision asynchronous	Load command access error
0	6	Synchronous	Store/AMO instruction access address misalignment
0	7	Non-precision asynchronous	Store/AMO command access error
0	8	Synchronous	Environment call in User mode
0	11	Synchronous	Environment call in Machine mode

"Synchronous" in the table means that an instruction can be located exactly where it is executed, such as an ebreak or ecall instruction, and each execution of that instruction will trigger an exception. "Asynchronous" means that it is not possible to pinpoint an instruction, and the instruction PC value may be different each time an exception occurs. " Precise asynchronous" means that an exception can be located exactly at the boundary of an instruction, i.e., the state after the execution of an instruction, such as an external interrupt. "Non-precision asynchronous" means that the boundary of an instruction cannot be precisely located, and may be the state after an instruction has been interrupted halfway through execution, such as a memory access error.

Access to memory takes time, and the microprocessor usually does not wait for the end of the access when accessing memory, but continues to execute the instruction, when the access error exception occurs again, the microprocessor has already executed the subsequent instructions, and cannot be precisely located.

2.2 Entering Exception

When the program is in the process of normal operation, if for some reason, triggered into an exception or interrupt. The hardware behavior of the microprocessor at this point can be summarized as follows.

(1) Suspend the current program flow and move to the execution of exception or interrupt handling functions. The entry base address and addressing mode of the exception or interrupt function are defined by the exception entry base address register mtvec. mtvec[31:2] defines the base address of the exception or interrupt function. mtvec[1:0] defines the addressing mode of the handler function. when mtvec[1:0]=0, all exceptions and interrupts use a unified entry, i.e., when an exception or interrupt occurs, it turns to the mtvec[31:2] defines the base address to execute. When mtvec[1:0]=1, exceptions and interrupts use vector table mode, i.e., each exception and interrupt is numbered, and the address is offset according to interrupt number*4, and when an exception or interrupt occurs, it is shifted to the base address defined by mtvec[31:2] + interrupt number*4 Execution. The interrupt vector table holds an instruction to jump to the interrupt handler function, or it can be other instructions.

(2) Update CSR register

When an exception or interrupt is entered, the microprocessor automatically updates the relevant CSR registers, including the Machine mode exception cause register mcause, the Machine mode exception pointer register mepc, the Machine mode exception value register mtval, and the Machine mode status register mstatus.

- Update mcause

As mentioned before, after entering an exception or interrupt, its value reflects the current exception type or interrupt number, and the software can read this register value to check the cause of the exception or determine the source of the interrupt, as detailed in Table 2-1.

- Update mepc

The standard definition of the return address of the microprocessor after exiting an exception or interrupt is stored in mepc. So when an exception or interrupt occurs, the hardware automatically updates the mepc value to the current instruction PC value when the exception is encountered, or the next pre-executed instruction PC value before the interrupt. After the exception or interrupt is processed, the microprocessor uses its saved value as the return address to return to the location of the interrupt to continue execution.

However, it is worth noting that.

1. mepc is a readable and writable register, and the software can also modify the value for the purpose of modifying the location of the PC pointer running after the return.
2. When an interrupt occurs, i.e., when the exception cause register mcause[31]=1, the value of mepc is updated to the PC value of the next unexecuted instruction at the time of the interrupt.

And when an exception occurs, the value of mepc is updated to the instruction PC value of the current exception when the exception cause register mcause[31]=0. So at this time when the exception returns, if we return directly using the value of mepc, we still continue to execute the instruction that generated the exception before, and at this time, we will continue to enter the exception. Usually, after we handle the exception, we can modify the value of mepc to the value of the next unexecuted instruction and then return. For example, if we cause an exception due to ecall/ebreak, after handling the exception, since ecall/ebreak (c.ebreak is 2 bytes)

is a 4-byte instruction, we only need the software to modify the value of mepc to mepc+4 (c.ebreak is mepc+2) and then return.

- Update mtval

When exceptions and interrupts are entered, the hardware will automatically update the value of mtval, which is the value that caused the exception. The value is typically.

1. If an exception is caused by a memory access, the hardware will store the address of the memory access at the time of the exception into mtval.
2. If the exception is caused by an illegal instruction, the hardware will store the instruction code of the instruction into mtval.
3. If the exception is caused by a hardware breakpoint, the hardware will store the PC value at the breakpoint into mtval.
4. For other exceptions, the hardware sets the value of mtval to 0, such as ebreak, the exception caused by ecall instruction.
5. When entering the interrupt, the hardware sets the value of mtval to 0.

- Update mstatus

Upon entering exceptions and interrupts, the hardware updates certain bits in mstatus.

1. MPIE is updated to the MIE value before entering the exception or interrupt, and MPIE is used to restore the MIE after the exception and interrupt are over.
2. MPP is updated to the privileged mode before entering exceptions and interrupts, and after the exceptions and interrupts are over, MPP is used to restore the previous privileged mode.
3. QingKe V3 microprocessor supports interrupt nesting in Machine mode, and MIE will not be cleared after entering exceptions and interrupts.

(3) Update microprocessor privilege mode

When exceptions and interrupts occur, the privileged mode of the microprocessor is updated to Machine mode.

2.3 Exception Handling Functions

Upon entering an exception or interrupt, the microprocessor executes the program from the address and mode defined by the mtvec register. When using the unified entry, the microprocessor takes a jump instruction from the base address defined by mtvec[31:2] based on the value of mtvec[1], or gets the exception and interrupt handling function entry address and goes to execute it instead. At this time, the exception and interrupt handling function can determine whether the cause is an exception or interrupt based on the value of mcause[31], and the type and cause of the exception or the corresponding interrupt can be judged by the exception code and handled accordingly.

When using the base address + interrupt number *4 for offset, the hardware automatically jumps to the vector table to get the entry address of the exception or interrupt function based on the interrupt number and jumps to execute it.

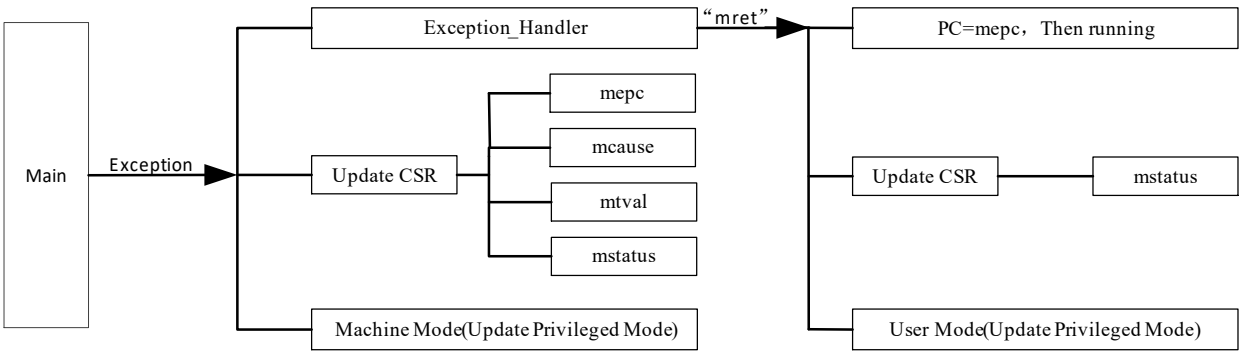
2.4 Exception Exit

After the exception or interrupt handler is completed, it is necessary to exit from the service program. After entering exceptions and interrupts, the microprocessor enters Machine mode from User mode, and the processing of exceptions and interrupts is also completed in Machine mode. When it is necessary to exit exceptions and interrupts, it is necessary to use the mret instruction to return. At this time, the microprocessor hardware will automatically perform the following operations.

- The PC pointer is restored to the value of CSR register mepc, i.e., execution starts at the instruction address saved by mepc. It is necessary to pay attention to the offset operation of mepc after the exception handling is completed.
- Update CSR register mstatus, MIE is restored to MPIE, and MPP is used to restore the privileged mode of the previous microprocessor.

The entire exception response process can be described by the following Figure 2-1.

Figure 2-1 Exception response process diagram



Chapter 3 PFIC and Interrupt Control

QingKe V3 microprocessor is designed with a Programmable Fast Interrupt Controller (PFIC) that can manage up to 256 interrupts including exceptions. The first 16 of them are fixed as internal interrupts of the microprocessor, and the rest are external interrupts, i.e. the maximum number of external interrupts can be extended to 240. Its main features are as follows.

- 240 external interrupts, each interrupt request has independent trigger and mask control bits, with dedicated status bits
- Programmable interrupt priority, supports 2 levels of nesting
- Special fast interrupt in/out mechanism, hardware automatic stacking and recovery, maximum HPE depth of 2 levels
- Vector Table Free (VTF) interrupt response mechanism, 2-channel programmable direct access to interrupt vector addresses

Note: The maximum nesting depth and HPE depth supported by interrupt controllers vary for different microprocessor models, which can be found in Table 1-1.

The vector table of interrupts and exceptions is shown in Table 3-1 below.

Table 3-1 Exception and interrupt vector table

Number	Priority	Type	Name	Description
0	-	-	-	-
1	-	-	-	-
2	-5	Fixed	NMI	Non-maskable interrupt
3	-4	Fixed	EXC	Exception interrupt
4	-	-	-	-
5	-3	Fixed	ECALL-M	Machine mode callback interrupt
6-7	-	-	-	-
8	-2	Fixed	ECALL-U	User mode callback interrupt
9	-1	Fixed	BREAKPOINT	Breakpoint callback interrupt
10-11	-	-	-	-
12	0	Programmable	SysTick	System timer interrupt
13	-	-	-	-
14	1	Programmable	SWI	Software interrupt
15	-	-	-	-
16-255	2-241	Programmable	External Interrupt	External interrupt 16-255

Note: ECALL-M, ECALL-U, BREAKPOINT are all different types of exception EXC, which are independent in V3B/C for ease of use, and the above 3 entry addresses are shared with EXC in V3A.

3.1 PFIC Register Set

Table 3-2 PFIC Registers

Name	Access address	Access	Description	Reset value
PFIC_ISRx	0xE000E000 -0xE000E01C	RO	Interrupt enable status register x	0x00000000
PFIC_IPRx	0xE000E020 -0xE000E03C	RO	Interrupt pending status register x	0x00000000

PFIC_ITHRESDR	0xE000E040	RW	Interrupt priority threshold configuration register	0x00000000
PFIC_VTFBADDRR	0xE000E044	RW	VTF base address register *Note: Valid only for V3A*	0x00000000
PFIC_CFGR	0xE000E048	RW	Interrupt configuration register *Note: Valid only for V3A*	0x00000000
PFIC_GISR	0xE000E04C	RO	Interrupt global status register	0x00000002
PFIC_VTFIDR	0xE000E050	RW	VTF interrupt ID configuration register *Note: Valid only for V3B/C.*	0x00000000
PFIC_VTFADDRRx	0xE000E060 -0xE000E06C	RW	VTF x offset address register	0xXXXXXXXX
PFIC_IENRx	0xE000E100 -0xE000E11C	WO	Interrupt enable setting register x	0x00000000
PFIC_IRERx	0xE000E180 -0xE000E19C	WO	Interrupt enable clear register x	0x00000000
PFIC_IPSRx	0xE000E200 -0xE000E21C	WO	Interrupt pending setting register x	0x00000000
PFIC_IPRRx	0xE000E280 -0xE000E29C	WO	Interrupt pending clear register x	0x00000000
PFIC_IACTRx	0xE000E300 -0xE000E31C	RO	Interrupt activation status register x	0x00000000
PFIC_IPRIORx	0xE000E400 -0xE000E43C	RW	Interrupt priority configuration register	0x00000000
PFIC_SCTLR	0xE000ED10	RW	System control register	0x00000000

Note: 1. NMI, EXC, ECALL-M, ECALL-U, BREAKPOINT are always enabled by default.

2. ECALL-M, ECALL-U, and BREAKPOINT are a case of EXC.

3. NMI, EXC, ECALL-M, ECALL-U, BREAKPOINT support interrupt pending clear and setting operation, but not interrupt enable clear and setting operation.

Each register is described as follows:

Interrupt enable status and interrupt pending status registers (PFIC_ISR<0-7>/PFIC_IPR<0-7>)

Name	Access address	Access	Description	Reset value
PFIC_ISR0	0xE000E000	RO	Interrupt 0-31 enable status register, a total of 32 status bits [n], indicating #n interrupt enable status *Note: NMI and EXC are enabled by default*	For V3A: 0x0000000C For V3B/C: 0x0000032C
PFIC_ISR1	0xE000E004	RO	Interrupt 32-63 enable status register, total 32 status bits	0x00000000
…	…	…	…	…
PFIC_ISR7	0xE000E01C	RO	Interrupt 224-255 enable status register, total 32 status bits	0x00000000
PFIC_IPR0	0xE000E020	RO	Interrupt 0-31 pending status	0x00000000

			register, a total of 32 status bits [n], indicating the pending status of interrupt #n	
PFIC_IPR1	0xE000E024	RO	Interrupt 32-63 pending status registers, 32 status bits in total	0x00000000
…	…	…	…	…
PFIC_IPR7	0xE000E03C	RO	Interrupt 244-255 pending status register, 32 status bits in total	0x00000000

Two sets of registers are used to enable and de-enable the corresponding interrupts.

Interrupt enable setting and clear registers (PFIC_IENR<0-7>/PFIC_IRER<0-7>)

Name	Access address	Access	Description	Reset value
PFIC_IENR0	0xE000E100	WO	Interrupt 0-31 enable setting register, a total of 32 setting bits [n], for interrupt #n enable setting *Note: NMI and EXC are enabled by default*	0x00000000
PFIC_IENR1	0xE000E104	WO	Interrupt 32-63 enable setting register, total 32 setting bits	0x00000000
…	…	…	…	…
PFIC_IENR7	0xE000E11C	WO	Interrupt 224-255 enable setting register, total 32 setting bits	0x00000000
-	-	-	-	-
PFIC_IRER0	0xE000E180	WO	Interrupt 0-31 enable clear register, a total of 32 clear bits [n], for interrupt #n enable clear *Note: NMI and EXC cannot be operated*	0x00000000
PFIC_IRER1	0xE000E184	WO	Interrupt 32-63 enable clear register, total 32 clear bits	0x00000000
…	…	…	…	…
PFIC_IRER7	0xE000E19C	WO	Interrupt 244-255 enable clear register, total 32 clear bits	0x00000000

Two sets of registers are used to enable and de-enable the corresponding interrupts.

Interrupt pending setting and clear registers (PFIC_IPSR<0-7>/PFIC_IPRR<0-7>)

Name	Access address	Access	Description	Reset value
PFIC_IPSR0	0xE000E200	WO	Interrupt 0-31 pending setting register, 32 setting bits [n], for interrupt #n pending setting	0x00000000
PFIC_IPSR1	0xE000E204	WO	Interrupt 32-63 pending setup register, total 32 setup bits	0x00000000
…	…	…	…	…

PFIC_IPSR7	0xE000E21C	WO	Interrupt 224-255 pending setting register, 32 setting bits in total	0x00000000
-	-	-	-	-
PFIC_IPRR0	0xE000E280	WO	Interrupt 0-31 pending clear register, a total of 32 clear bits [n], for interrupt #n pending clear	0x00000000
PFIC_IPRR1	0xE000E284	WO	Interrupt 32-63 pending clear register, total 32 clear bits	0x00000000
…	…	…	…	…
PFIC_IPRR7	0xE000E29C	WO	Interrupt 244-255 pending clear register, total 32 clear bits	0x00000000

When the microprocessor enables an interrupt, it can be set directly through the interrupt pending register to trigger into the interrupt. Use the interrupt pending clear register to clear the pending trigger.

Interrupt activation status register (PFIC_IACTR<0-7>)

Name	Access address	Access	Description	Reset value
PFIC_IACTR0	0xE000E300	RO	Interrupt 0-31 activates the status register with 32 status bits [n], indicating that interrupt #n is being executed	0x00000000
PFIC_IACTR1	0xE000E304	RO	Interrupt 32-63 activation status registers, 32 status bits in total	0x00000000
…	…	…	…	…
PFIC_IACTR7	0xE000E31C	RO	Interrupt 224-255 activation status register, total 32 status bits	0x00000000

Each interrupt has an active status bit that is set up when the interrupt is entered and cleared by hardware when mret returns.

Interrupt priority and priority threshold registers (PFIC_IPRIOR<0-7>/PFIC_ITHRESDR)

Name	Access address	Access	Description	Reset value
PFIC_IPRIOR0	0xE000E400	RW	Interrupt 0 priority configuration. V3A: [7:4]: Priority control bits If the configuration is not nested, no preemption bit If nesting is configured, bit7 is the preempted bit. [3:0]: Reserved, fixed to 0 V3B: [7:6]: Priority control bits If configuration is not nested, no preemptive bits	0x00

			If configured nested, all bits are preempted, but up to two levels of interrupts are allowed to occur [5:0]: Reserved, fixed to 0 V3C: [7:5]: Priority control bits If configuration is not nested, no preemptive bits If configured nested, all bits are preempted, but up to two levels of interrupts are allowed to occur [4:0]: Reserved, fixed to 0 *Note: The smaller the priority value, the higher the priority. If the same preemption priority interrupt hangs at the same time, the interrupt with the higher priority will be executed first.*	
PFIC_IPRIOR1	0xE000E401	RW	Interrupt 1 priority setting, same function as PFIC_IPRIOR0	0x00
PFIC_IPRIOR2	0xE000E402	RW	Interrupt 2 priority setting, same function as PFIC_IPRIOR0	
…	…	…	…	…
PFIC_IPRIOR254	0xE000E4FE	RW	Interrupt 254 priority setting, same function as PFIC_IPRIOR0	0x00
PFIC_IPRIOR255	0xE000E4FF	RW	Interrupt 255 priority setting, same function as PFIC_IPRIOR0	0x00
-	-	-	-	-
PFIC_ITHRESDR	0xE000E040	RW	Interrupt priority threshold setting V3A: [31:8]: Reserved, fixed to 0 [7:4]: Priority threshold [3:0]: Reserved, fixed to 0 V3B: [31:8]: Reserved, fixed to 0 [7:5]: Priority threshold [4:0]: Reserved, fixed to 0 V3C: [31:8]: Reserved, fixed to 0 [7:5]: Priority threshold [4:0]: Reserved, fixed to 0	0x00

			Note: For interrupts with priority value ⩾ threshold, the interrupt service function is not executed when a hang occurs, and when this register is 0, it means the threshold register is invalid.	

Interrupt configuration register (PFIC_CFGR)

Name	Access address	Access	Description	Reset value
PFIC_CFGR	0xE000E048	RW	Interrupt configuration register	0x00000000

This register is valid only for V3A, its bits are defined as:

Bit	Name	Access	Description	Reset value
[31:16]	KEYCODE	WO	Corresponding to different target control bits, the corresponding security access identification data needs to be written simultaneously in order to be modified, and the readout data is fixed to 0. KEY1 = 0xFA05; KEY2 = 0xBCAF; KEY3 = 0xBEEF。	0
[15:8]	Reserved	RO	Reserved	0
7	SYSRESET	WO	System reset (simultaneous writing to KEY3). Auto clear 0. Writing 1 is valid, writing 0 is invalid. *Note: Same function as the PFIC_SCTLR register SYSRESET bit.*	0
6	PFICRESET	WO	PFIC module reset. Auto clear 0. Writing 1 is valid, writing 0 is invalid.	0
5	EXCRESET	WO	Exception interrupt pending clear (simultaneous writing to KEY2) Writing 1 is valid, writing 0 is invalid.	0
4	EXCSET	WO	Exception interrupt pending setting (simultaneous writing to KEY2) Writing 1 is valid, writing 0 is invalid.	0
3	NMIRESET	WO	NMI interrupt pending clear (simultaneous writing to KEY2) Writing 1 is valid, writing 0 is invalid.	0
2	NMISET	WO	NMI interrupt pending setting (Simultaneous writing to KEY2) Writing 1 is valid, writing 0 is invalid.	0
1	NESTCTRL	RW	Interrupt nesting enable control. 1: off; 0: on (synchronous writing to KEY1)	0
0	HWSTKCTRL	RW	HPE enable control 1: off; 0: on (synchronous writing to KEY1)	0

Interrupt global status register (PFIC_GISR)

Name	Access address	Access	Description	Reset value
PFIC_GISR	0xE000E04C	RO	Interrupt global status register	0x00000000

Its folks are defined as.

Bit	Name	Access	Description	Reset value
[31:14]	Reserved	RO	Reserved	0
13	LOCKSTA	RO	Whether the processor is currently in a locked state: 1: Locked state; 0: Non-locked state. *Note: This bit is only valid for the V3B/C.*	0
12	DBGMODE	RO	Whether the processor is currently in debug state: 1: Debug state; 0: Non-debug state. *Note: This bit is only valid for the V3B/C.*	0
11	GLOBLIE	RO	Global interrupt enable: 1: Enable interrupt; 0: Disable interrupt. *Note: This bit is only valid for the V3B/C.*	
10	Reserved	RO	Reserved	0
9	GPENDSTA	RO	Whether an interrupt is currently pending. 1: Yes; 0: No.	0
8	GACTSTA	RO	Whether an interrupt is currently being executed. 1: Yes; 0: No.	0
[7:0]	NESTSTA	RO	Current interrupt nesting status. 0x03: in level 2 interrupt. 0x01: in level 1 interrupt. 0x00: no interrupts occur. Other: Impossible situation.	0

VTF ID base address and offset address registers (PFIC_VTFBADDRR/PFIC_VTFADDRR<0-3>)

Name	Access address	Access	Description	Reset value
PFIC_VTFBADDRR	0xE000E044	RW	[31:28]: High 4 bits of the target address of VTF [27:0]: Reserved *This register is valid only for V3A.*	0x00000000
PFIC_VTFIDR	0xE000E050	RW	[31:24]: Number of VTF 3 [23:16]: Number of VTF 2 [15:8]: Number of VTF 1 [7:0]: Number of VTF 0 *This register is valid only for V3B/C.*	0x00000000
-	-	-	-	-

PFIC_VTFADDRR0	0xE000E060	RW	V3A: [31:24]: VTF 0 interrupt number [23:0]: the low 24 bits of the VTF target address, of which the low 20 bits are configured to be valid and [23:20] is fixed to 0. V3B/C: [31:1]: VTF 0 address, 2-byte aligned [0]: 1: Enable VTF 0 channel 0: Disable	For V3A: 0x00000000 For V3B/C: 0xXXXXXXXX
PFIC_VTFADDRR1	0xE000E064	RW	V3A: [31:24]: VTF 1 interrupt number [23:0]: The low 24 bits of the VTF target address, of which the low 20 bits are configured to be valid and [23:20] is fixed to 0. V3B/C: [31:1]: VTF 1 address, 2-byte aligned [0]: 1: Enable VTF 1 channel 0: Disable	For V3A: 0x00000000 For V3B/C: 0xXXXXXXXX
PFIC_VTFADDRR2	0xE000E068	RW	V3A: [31:24]: VTF 2 interrupt number [23:0]: the low 24 bits of the VTF target address, of which the low 20 bits are configured to be valid and [23:20] is fixed to 0. V3B/C: [31:1]: VTF 2 address, 2-byte aligned [0]: 1: Enable VTF 2 channel 0: Disable	For V3A: 0x00000000 For V3B/C: 0xXXXXXXXX
PFIC_VTFADDRR3	0xE000E06C	RW	V3A:	For V3A:

			[31:24]: VTF 3 interrupt number [23:0]: the low 24 bits of the VTF target address, of which the low 20 bits are configured to be valid and [23:20] is fixed to 0. V3B/C: [31:1]: VTF 3 address, 2-byte aligned [0]: 1: Enable VTF 3 channel 0: Disable	0x00000000 For V3B/C: 0xXXXXXXXX

System control register (PFIC_SCTLR)

Name	Access address	Access	Description	Reset value
PFIC_SCTLR	0xE000ED10	RW	System control register	0x00000000

Each of them is defined as follows.

Bit	Name	Access	Description	Reset value
31	SYSRESET	WO	System reset, auto clear 0. Write 1 is valid, write 0 is invalid. *Note: This bit is only valid for V3B/C*	0
[30:6]	Reserved	RO	Reserved	0
5	SETEVENT	WO	Set the event to wake up the WFE case.	0
4	SEVONPEND	RW	When an event occurs or interrupts a pending state, the system can be woken up from after the WFE instruction, or if the WFE instruction is not executed, the system will be woken up immediately after the next execution of the instruction. 1: Enabled events and all interrupts (Including unenabled interrupts) can wake up the system. 0: Only enabled events and enabled interrupts can wake up the system.	0
3	WFITOWFE	RW	Execute the WFI command as if it were a WFE. 1: Treat the subsequent WFI instruction as a WFE instruction. 0: No effect.	0
2	SLEEPDEEP	RW	Low power mode of the control system.	0

			1: deepsleep 0: sleep	
1	SLEEPONEXIT	RW	System status after control leaves the interrupt service program. 1: The system enters low-power mode. 0: The system enters the main program.	0
0	Reserved	RO	Reserved	0

3.2 Interrupt-related CSR Registers

In addition, the following CSR registers also have a significant impact on the processing of interrupts.

Interrupt system control register (intsyscr)

This register is not valid for V3A only:

Name	CSR Address	Access	Description	Reset value
intsyscr	0x804	URW	Interrupt system control register	0x0000E002

Its folks are defined as:

Bit	Name	Access	Description	Reset value
31	LOCK	URO	0: This register can be read and written in user mode; 1: This register can only be read and written in machine mode. *Note: This configuration bit is valid from version 1.0 onwards.*	0
[30:6]	Reserved	URO	Reserved	0x380
5	GIHWSTKNEN	URW1	Global interrupt and hardware stack shutdown are enabled. *Note: This bit is often used in real-time operating systems. When the context is switched during an interrupt, setting this bit can turn off the global interrupt and push the hardware stack. When the context switch is completed and the interrupt returns, the hardware will automatically clear this bit.*	0
4	Reserved	URO	Reserved	0
[3:2]	PMTCFG	URW	Configuration of priority preemption bits: 00: The number of preemption bits is 0; 01: The number of preemption bits is 1; 10: The number of preemption bits is 2; 11: The number of preemption bits is 3; *Note: This configuration bit is valid after 1.0.*	0
1	INESTEN	URW	The interrupt nesting function is enabled, and the fixed value is 1:	1

			0: Disable; 1: Enable. *Note: 1. The actual nesting level is controlled by NEST_LVL in CSR 0xBC1;* *2. Only versions after 1.0 can be written.*	
0	HWSTKEN	URW	Hardware stack enable: 0: Hardware stack pressing function is disabled; 1: Hardware stack pressing function is enabled.	0

Machine mode exception base address register (mtvec)

Name	CSR Address	Access	Description	Reset value
mtvec	0x305	MRW	Exception base address register	0x00000000

Its folks are defined as.

Bit	Name	Access	Description	Reset value
[31:2]	BASEADDR[31:2]	MRW	Interrupt vector table base address, where bits [9:2] are fixed to 0.	0
1	MODE1	MRO	Interrupt vector table recognition mode: 0: Identify by jump instruction, with limited scope, and support non-jump instruction; 1: Identify by absolute address, support full range, but must jump. *Note: This bit is only valid for V3B/C.*	0
0	MODE0	MRW	Interrupt or exception entry address mode selection. 0: Use of the uniform entry address. 1: Address offset based on interrupt number *4.	0

For MCUs with V3 series microprocessors, MODE0 is configured to be 1 by default in the startup file, and the entries for exceptions or interrupts are offset according to the interrupt number *4. Note that the V3A microprocessor stores a jump instruction at the vector table, while the V3B/C microprocessor can be either a jump instruction or use the absolute address of the interrupt function, which is configured as an absolute address in the default startup file.

Microprocessor configuration register (corecfgr)

This register is invalid for V3A:

Name	CSR Address	Access	Description	Reset value
corecfgr	0xBC0	MRW	Microprocessor configuration register	0x00000001

Its folks are defined as.

Bit	Name	Access	Description	Reset value
[31:8]	Reserved	MRO	Reserved	0

7	CSTA_FAULT_IE	MRW	Core status error interrupt enable: 0: On status error, no NMI interrupt is generated; 1: On status error, NMI interrupt is generated.	0
6	Reserved	MRO	Keep it 0.	0
5	IE_REMAP_EN	MRW	MIE register mapping enable: 0: CSR address 0x800 is a read-only register and the return value is the value of MSTATUS; 1: Bits 3 and 7 of CSR address 0x800 are mapped to bit MIE of the MSTATUS register and bit MPIE of the MSTATUS register, respectively.	0
4	Reserved	MRO	Reserved	0
3	ROM_LOOP_ACC	MRW	ROM area instruction loop acceleration enable: 0: Turn off the cyclic acceleration function in ROM area; 1: Continuous instructions with loop body within 128 bytes will be fully accelerated, while those with loop body within 256 bytes will be partially accelerated;	0
2	ROM_JUMP_ACC	MRW	ROM area instruction jump acceleration enabled: 0: Disable ROM area instruction jump acceleration; 1: Enable instruction jump acceleration in ROM area.	0
[1:0]	FETCH_MODE	MRW	Fetching mode: 00: Prefetch is off. The instruction prefetch function is turned off to avoid invalid instruction fetching operation, and there is at most one valid instruction on the CPU pipeline. This mode has the lowest power consumption, and its performance drops by about 2 ~ 3 times. 01: Prefetch Mode 1. When the instruction prefetch function is turned on, the CPU will continue to access the instruction memory until the number of instructions to be executed in the internal instruction buffer exceeds a certain number, or the instruction buffer is full, and instruction fetching will be suspended; (Failure of CPU prediction will lead to redundant	0x1

			fetch operation, and in some cases, the execution unit will introduce 0 ~ 2 cycles of bubbles, and the performance of most programs will not decrease obviously); 10: Reserved; 11: Prefetch Mode 2. When the instruction prefetch function is turned on, the CPU will continue to access the instruction memory, and if the instruction buffer is full, the CPU will continue to retry the address. This mode has the highest performance and power consumption. CPU prediction failure and retry will introduce redundant fetch operations and may continue to occupy memory bandwidth. (For ROM area, retry means discontinuous address access, so it is recommended to turn on ROM_ACC_EN).	

Interrupt nested control register (inestcr)

This register is invalid only for V3A:

Name	CSR Address	Access	Description	Reset value
inestcr	0xBC1	MRW	Interrupt nested control register	0x00000000

Its folks are defined as.

Bit	Name	Access	Description	Reset value
31	Reserved	MRO	Reserved	0
30	NEST_OV	MRW	Interrupt/exception nested overflow flag bit, write 1 to clear: 0: Interrupt did not overflow; 1: Interrupt overflow flag. *Note: Interrupt overflow will only occur when executing secondary interrupt service function to generate instruction exception or NMI interrupt. At this time, the exception and NMI interrupt enter normally, but the CPU stack overflows, so you cannot exit from this exception and NMI interrupt.*	0
[29:12]	Reserved	MRO	Reserved	0
[11:8]	NEST_STA	MRO	Nested status flag bit: 0000: No interrupt; 0001: Level 1 interrupt; 0011: level 2 interrupt (1-level nesting);	0

			0111: Level 3 interrupt (overflow); 1111: Level 4 interrupt (overflow).	
[7:2]	Reserved	MRO	Reserved	0
[1:0]	NEST_LVL	MRW	Nesting level: 00: Nesting is prohibited and the nesting function is turned off; 01: First-level nesting, which turns on the nesting function; Other: Invalid. *Note: Write 10 or 11 to this field, and the field will be set to 01. When write 11 to this field, read this register to get the highest nesting level of the chip.*	0

User mode global interrupt enable register (gintenr)

This register is invalid only for V3A:

Name	CSR Address	Access	Description	Reset value
gintenr	0x800	URW	Global interrupt enable register	0x00000000

This register is used to control the enable and mask of global interrupt. The enable and mask of global interrupt in machine mode can be controlled by the MIE and MPIE bits in mstatus, but this register cannot be operated in user mode. The global interrupt enable register gintenr is the mapping of MIE and MPIE in mstatus, and can be used to set and clear MIE and MPIE by operating gintenr in user mode. Each of them is defined as:

Bit	Name	Access	Description	Reset value
[31:13]	Reserved	URO	Reserved	0
[12:11]	MPP	URO	Enter privileged mode before interruption.	0
[10:8]	Reserved	URO	Reserved	0
7	MPIE	URW	When 0xBC0(CSR)bit5 is enabled, this bit can be read and written in user mode.	0
[6:4]	Reserved	URO	Reserved	0
3	MIE	URW	When 0xBC0(CSR)bit5 is enabled, this bit can be read and written in user mode.	0
[1:0]	Reserved	URO	Reserved	0

3.3 Interrupt Nesting

In conjunction with the interrupt configuration register PFIC_CFGR and the interrupt priority register PFIC_IPRIOR, nesting of interrupts can be allowed to occur. Enable nesting in the interrupt configuration register (Nesting is turned on by default for V3 series microprocessors) and configure the priority of the corresponding interrupt. The smaller the priority value, the higher the priority. The smaller the value of the preemption bit, the higher the preemption priority. If there are interrupts hanging at the same time under the same preemption priority, the microprocessor responds to the interrupt with the lower priority value (higher priority) first.

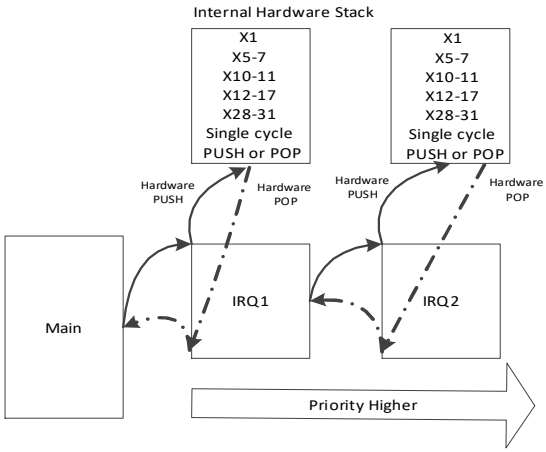
3.4 Hardware Prologue/Epilogue (HPE)

When an exception or interrupt occurs, the microprocessor stops the current program flow and shifts to the execution of the exception or interrupt handling function, the site of the current program flow needs to be saved. After the exception or interrupt returns, it is necessary to restore the site and continue the execution of the stopped program flow. For V3 series microprocessors, the "site" here refers to all the Caller Saved registers in Table 1-2.

The V3 series microprocessors support hardware single-cycle automatic saving of 16 of the shaped Caller Saved registers to an internal stack area that is not visible to the user. When an exception or interrupt returns, the hardware single cycle automatically restores the data from the internal stack area to the 16 shaped registers. HPE supports nesting up to 2 levels deep.

A schematic of the microprocessor pressure stack is shown in the following figure.

Figure 3-1 Schematic diagram of pressure stack



Note: 1. Interrupt functions using the HPE need to be compiled using MRS or its provided toolchain and the interrupt function needs to be declared with __attribute__((interrupt("WCH-Interrupt-fast"))).

2. The interrupt function using stack push is declared by __attribute__((interrupt())).

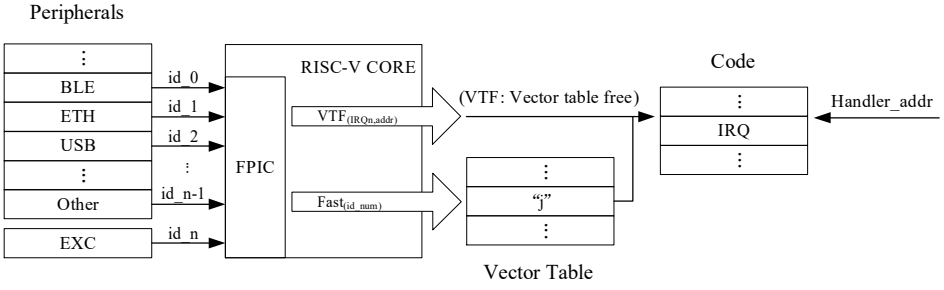
3.5 Vector Table Free (VTF)

The Programmable Fast Interrupt Controller (PFIC) provides 4 VTF channels, i.e., direct access to the interrupt function entry without going through the interrupt vector table lookup process.

The VTF channel can be enabled by writing its interrupt number, interrupt service function base address and offset address into the corresponding PFIC controller register while configuring an interrupt function normally.

The PFIC response process for fast and table-free interrupts is shown in Figure 3-2 below.

Figure 3-2 Schematic diagram of programmable fast interrupt controller



Chapter 4 Physical Memory Protection (PMP)

In order to improve the security of the system, the physical memory protection (PMP) module is designed according to RISC-V architecture standard for the V3 series microprocessors of highland barley. Access rights management of up to 4 physical region is supported. Permissions include read (R), write (W) and execute (X) attributes, and the length of the protected area can be set to 4 bytes at least. PMP module always takes effect in user mode, but it can take effect optionally by locking (L) attribute in machine mode.

If the access violates the current permission limit, it will trigger an abnormal interrupt. The PMP module includes four groups of 8-bit configuration registers (One group of 32-bit) and four groups of address registers, all of which need to be accessed in machine mode by CSR instruction.

Note: The number of protected areas supported by PMP in different models of microprocessors may be different, and the number supported by pmpcfg and pmpaddr registers is also different. See Table 1-1 for details.

4.1 PMP Register Set

The list of CSR registers supported by PMP module of V3 microprocessor is shown in Table 4-1 below.

Table 4-1 PMP module register set

Name	CSR address	Access	Description	Reset value
pmpcfg0	0x3A0	MRW	PMP configuration register 0	0x00000000
pmpaddr0	0x3B0	MRW	PMP address register 0	0xXXXXXXXX
pmpaddr1	0x3B1	MRW	PMP address register 1	0xXXXXXXXX
pmpaddr2	0x3B2	MRW	PMP address register 2	0xXXXXXXXX
pmpaddr3	0x3B3	MRW	PMP address register 3	0xXXXXXXXX

4.2 pmp<i>cfg Register

pmpcfg is the configuration register of PMP unit, and each register contains four 8-bit pmpcfg fields, corresponding to the configuration of four regions, and pmpcfg represents the configuration value of region i. Its format is shown in the following table 4-2.

Table 4-2 pmpcfg0 register

31 24	23 16	15 8	7 0	
pmp3cfg	pmp2cfg	pmp1cfg	pmp0cfg	pmpcfg0

pmpcfg is used to configure area i, and its bit definition is described in the following table 4-3.

Table 4-3 pmp<i>cfg register

Bit	Name	Description
7	L	Locking is enabled and can be unlocked in machine mode. 0: Not locked; 1: Lock the relevant register.
[6:5]	-	Reserved
[4:3]	A	Address alignment and protection area range selection. 00: OFF (PMP off) 01: TOR (Top alignment protection) 10: NA4 (Fixed four-byte protection) 11: NAPOT ($2^{(G+2)}$ Byte protection, G≥1)
2	X	Executable attribute.

		0: No execute permission; 1: Execute permission.
1	W	Writeable attribute. 0: No write permission 1: Write permission.
0	R	Readable attribute 0: No read permission 1: Read permission.

4.3 pmpaddr<i> Register

The pmpaddr register is used to configure the address of area I. The standard definition is under RV32 architecture, which is the encoding of the upper 32 bits of a 34-bit physical address, and its format is shown in the following table 4-4. The whole physical address space of V3 microprocessor is 4G, so the upper two bits of this register are not used.

Table 4-4 pmpaddr<i> register

31	0
address[33:2]	

When NAPOT is selected, the low bit of the address register is also used to indicate the size of the current protection area, as shown in the following table, where 'y' is a bit of the register.

Table 4-5 Relationship table between PMP configuration and address register and protected area

pmpaddr	pmpcfg.A	Matching base address and size
yyyy…yyyy	NA4	With 'yy…yyyy00' as the base address, the 4-byte area is protected.
yyyy…yyy0	NAPOT	With 'yy…yyy000' as the base address, the 8-byte area is protected.
yyyy…yy01	NAPOT	With 'yy…yy0000' as the base address, the 16-byte area is protected.
yyyy…y011	NAPOT	With 'yy…y00000' as the base address, the 16-byte area is protected.
…	…	…
yyy01…111	NAPOT	With 'y0…000000' as the base address, the 2^{31}-byte area is protected.
yy011…111	NAPOT	Protect the entire 2^{32}-byte area.

4.4 Protection Mechanism

X/W/R in pmpcfg is used to set the protection authority of area I, and violation of relevant authority will cause corresponding exception:

(1) When trying to fetch instructions in the PMP area without execution authority, it will cause an instruction fetch access error exception (mcause=1).

(2) When trying to write data in the PMP area without write permission, it will cause an error exception (mcause=7) in the store instruction access.

(3) When trying to read data in the PMP area without read permission, it will cause an abnormal memory access error (mcause=5) for the load instruction.

A in pmpcfg is used to set the protection range and address alignment of region i, and to protect the memory of A_ADDR ≤ region<i><B_ADDR (both A_ADDR and B_ADDR are required to be aligned in 4 bytes):

(1) If B _ ADDR–A_ADDR = = 2^2, NA4 mode is adopted;

(2) If B _ ADDR - A_ADDR = = $2^{(G+2)}$, G≥1, and a _ addr is $2^{(g+2)}$, the NAPOT method is adopted;

(3) Otherwise, the TOP mode is adopted.

Table 4-6 PMP address matching methods

A value	Name	Description
0b00	OFF	No area to protect
0b01	TOR	Top Aligned Area Protection. Under pmp<i>cfg, $pmpaddr_{i-1}$≤ region<i> <$pmpaddr_i$; $pmpaddr_{i-1}$ = A_ADDR >> 2; $pmpaddr_i$ = B_ADDR >> 2. *Note: If area 0 of PMP is configured as TOR mode (i=0), the lower boundary of the protection area is 0 address, i.e. 0 ≤ addr < pmpaddr0, all within the matching range.*
0b10	NA4	Fixed 4-byte area protection. pmp<i>cfg under pmpaddri as the starting address of the 4-byte pmpaddri = A_ADDR>>2.
0b11	NAPOT	Protect the $2^{(G+2)}$ region with G ≥ 1, when A_ADDR is $2^{(G+2)}$ aligned. pmpaddri = ((A_ADDR|($2^{(G+2)}$-1)) &~(1<<G+1))>>2.

The L bit in pmp<i>cfg is used to lock the PMP entry. After locking, the configuration register pmp<i>cfg and the address register pmpaddr<i> will not be able to be modified. If A in pmp<i>cfg is set to TOR mode, pmpaddr<i-1> will also not be modified. when L is set, the X/W/R permissions defined in pmp<i>cfg are also valid for machine mode, and when L is cleared, X/W/R is only valid for user mode, and L is cleared only after system reset.

QingKe V3 series microprocessors support protection of multiple zones. When the same operation matches multiple zones at the same time, the zone with the smaller number is matched first.

Chapter 5 System Timer (SysTick)

QingKe V3 series microprocessor is designed with a 32-bit or 64-bit counter (SysTick) inside. Its clock source is the system clock or its 8-frequency division, and V3A only supports 8-frequency division. It can provide time base, timing and measuring time for real-time operating system. Different types of registers involved in the timer have different mapping addresses, as shown in the following tables 5-1 and 5-2.

Table 5-1 V3A SysTick register list

Name	Access address	Description	Reset value
STK_CTLR	0xE000F000	System counter control register	0x00000000
STK_CNTL	0xE000F004	System counter low register	0xXXXXXXXX
STK_CNTH	0xE000F008	System counter high register *Note: Only valid for V3A.*	0xXXXXXXXX
STK_CMPLR	0xE000F00C	System count comparison value low register	0xXXXXXXXX
STK_CMPHR	0xE000F010	System count comparison value high register *Note: Only valid for V3A.*	0xXXXXXXXX

Table 5-2 V3 SysTick register list of other models

Name	Access address	Description	Reset value
STK_CTLR	0xE000F000	System counter control register	0x00000000
STK_SR	0xE000F004	System counter status register	0x00000000
STK_CNTL	0xE000F008	Low register of system counter	0xXXXXXXXX
STK_CMPLR	0xE000F010	Count comparison value low register	0xXXXXXXXX

Each register is described in detail as follows.

System counter control register (STK_CTLR)

Table 5-3 SysTick control registers

Bit	Name	Access	Description	Reset value
[31:5]	Reserved	RO	Reserved	0
4	MODE	RW	Counting mode: 1: Count down; 0: Count up. *Note: Invalid for V3A.*	0
3	STRE	RW	Automatic reload count enable bit: 1: Count from 0 again after counting up to the comparison value, and count from the comparison value again after counting down to 0; 0: Continue counting up/down. *Note: Invalid for V3A.*	0
2	STCLK	RW	Counter clock source selection bit: 1: HCLK as time base; 0: HCLK/8 as time base. *Note: It is invalid for V3A, which only supports HCLK/8 as time base.*	0
1	STIE	RW	Counter interrupt enable control bits:	0

			1: Enable counter interrupt; 0: Disable counter interrupt. *Note: Invalid for V3A.*	
0	STE	RW	System counter enable control bit. 1: Enable system counter STK; 0: Disable the system counter STK and the counter stops counting.	0

System counter status register (STK_SR)

This register does not apply to V3A:

Table 5-4 SysTick counter low register

Bit	Name	Access	Description	Reset value
31	SWIE	RW	Software interrupt trigger enable (SWI): 1: Trigger software interrupt; 0: Turn off the trigger. *Note: This bit must be cleared after entering the software interrupt, otherwise it will always trigger.*	0
[30:1]	Reserved	RO	Reserved	0
0	CNTIF	RW	Count comparison flag, write 0 clearly, write 1 is invalid: 1: Count up to the comparison value and count down to 0; 0: The comparison value is not reached.	0

System counter low register (STK_CNTL)

Table 5-5 SysTick counter low register

Bit	Name	Access	Description	Reset value
[31:0]	CNTL	RW	The current counter count value is 32 bits lower. For V3A, this register can be read as 8-bit /16-bit /32-bit, but can only be written as 8-bit, and other models are not limited.	0xXXXXXXXX

Note: Register STK_CNTL and register STK_CNTH in V3A together constitute a 64-bit system counter.

System counter high register (STK_CNTH)

Table 5-6 SysTick counter high register

Bit	Name	Access	Description	Reset value
[31:0]	CNTH	RW	The current counter count value is 32 bits higher. This register can be read by 8-bit/16-bit/32-bit, but can only be written by 8-bit. *Note: Only valid for V3A.*	0xXXXXXXXX

Note: Register STK_CNTL and register STK_CNTH in V3A together constitute a 64-bit system counter.

System count comparison value low register (STK_CMPLR)

Table 5-7 SysTick comparison value low register

Bit	Name	Access	Description	Reset value

[31:0]	CMPL	RW	Set the counter comparison value 32 bits lower. When CMP value and CNT value are equal, STK interrupt will be triggered. For V3A, this register can be read as 8-bit /16-bit /32-bit, but can only be written as 8-bit, and other models are not limited.	0xXXXXXXXX

Note: The register STK_CMPLR and the register STK_CMPHR in V3A together constitute the 64-bit counter comparison value.

System count comparison value high register (STK_CMPHR)

Table 5-8 SysTick comparison value high register

Bit	Name	Access	Description	Reset value
[31:0]	CMPH	RW	Set the counter comparison value 32 bits higher. The STK interrupt will be triggered when the CMP value and CNT value are equal. This register can be read by 8-bit/16-bit/32-bit, but can only be written by 8-bit. *Note: Only valid for V3A.*	0xXXXXXXXX

Note: The register STK_CMPLR and the register STK_CMPHR in V3A together constitute the 64-bit counter comparison value.

Chapter 6 Processor Low-power Settings

QingKe V3 series microprocessors support sleep state via WFI (Wait for Interrupt) instruction to achieve low static power consumption. Together with PFIC's system control register (PFIC_SCTLR), various Sleep modes and WFE instructions can be implemented.

6.1 Enter Sleep

QingKe V3 series microprocessors can go to sleep in two ways, Wait for Interrupt (WFI) and Wait For Event (WFE). The WFI method means that the microprocessor goes to sleep, waits for an interrupt to wake up, and then wakes up to the corresponding interrupt to execute. The WFE method means that the microprocessor goes to sleep, waits for an event to wake up, and wakes up to continue executing the previously stopped program flow.

The standard RISC-V supports WFI instruction, and the WFI command can be executed directly to enter sleep by WFI method. For the WFE method, the WFITOWFE bit in the system control register PFIC_SCTLR is used to control the subsequent WFI commands as WFE processing to achieve the WFE method to enter sleep.

The depth of sleep is controlled according to the SLEEPDEEP bit in PFIC_SCTLR.

- If the SLEEPDEEP in PFIC_SCTLR register is cleared to zero, the microprocessor enters Sleep mode and the internal unit clock is allowed to be turned off except for SysTick and part of the wake-up logic.
- If SLEEPDEEP in the PFIC_SCTLR register is set, the microprocessor enters Deep sleep mode and all cell clocks are allowed to be turned off.

When the microprocessor is in Debug mode, it is not possible to enter any kind of Sleep mode.

6.2 Sleep Wakeup

QingKe V3 series microprocessors can be woken up after sleep due to WFI and WFE in the following ways.

- After the WFI method goes to sleep, it can be awakened by

(1) The microprocessor can be woken up by the interrupt source responded by the interrupt controller. After waking up, the microprocessor executes the interrupt function first.

(2) Enter Sleep mode, debug request can make the microprocessor wake up and enter deep sleep, debug request cannot wake up the microprocessor.

- After the WFE method goes to sleep, the microprocessor can be woken up by the following.

(1) Internal or external events, when there is no need to configure the interrupt controller, wake up and continue to execute the program.

(2) If an interrupt source is enabled, the microprocessor is woken up when an interrupt is generated, and after waking up, the microprocessor executes the interrupt function first.

(3) If the SEVONPEND bit in PFIC_SCTLR is configured, the interrupt controller does not enable the interrupt under, but when a new interrupt pending signal is generated (the previously generated pending signal does not take effect), it can also make the microprocessor wake up, and the corresponding interrupt pending flag needs to be cleared manually after waking up.

(4) Enter Sleep mode debug request can make the microprocessor wake up and enter deep sleep, debug request cannot wake up the microprocessor.

In addition, the state of the microprocessor after wake-up can be controlled by configuring the SLEEPONEXIT

bit in PFIC_SCTLR.

- SLEEPONEXIT is set and the last level interrupt return instruction (mret) will trigger the WFI mode sleep.

- SLEEPONEXIT is cleared with no effect.

Various MCU products equipped with V3 series microprocessors can adopt different sleep modes, turn off different peripherals and clocks, implement different power management policies and wake-up methods according to different configurations of PFIC_SCTLR, and realize various low-power modes.

Chapter 7 Debug Support

QingKe V3 series microprocessors include a hardware debug module that supports complex debugging operations. When the microprocessor is suspended, the debug module can access the microprocessor's GPRs, CSRs, Memory, external devices, etc. through abstract commands, program buffer deployment instructions, etc. The debug module can suspend and resume the microprocessor's operation.
The debug module follows the RISC-V External Debug Support Version0.13.2 specification, detailed documentation can be downloaded from RISC-V International website.

7.1 Debug Module

The debug module inside the microprocessor, capable of performing debug operations issued by the debug host, includes.

- Access to registers through the debug interface
- Reset, suspend and resume the microprocessor through the debug interface
- Read and write memory, instruction registers and external devices through the debug interface
- Deploy multiple arbitrary instructions through the debug interface
- Set software breakpoints through the debug interface
- Set hardware breakpoints through the debug interface
- Support abstract command auto-execution
- Support single-step debugging

Note: V3A does not support hardware breakpoints, V3B hardware breakpoints support instruction address matching, and V3C hardware breakpoints support instruction address and data address matching.

The internal registers of the debugging module use a 7-bit address code, and the following registers are implemented inside QingKe V3 series microprocessors.

Table 7-1 Debug module register List

Name	Access address	Description
data0	0x04	Data register 0, can be used for temporary storage of data
data1	0x05	Data register 1, can be used for temporary storage of data
dmcontrol	0x10	Debug module control register
dmstatus	0x11	Debug module status register
hartinfo	0x12	Microprocessor status register
abstractcs	0x16	Abstract command status register
command	0x17	Abstract command register
abstractauto	0x18	Abstract command auto-execution
progbuf0-7	0x20-0x27	Instruction cache registers 0-7
haltsum0	0x40	Pause status register

The debug host can control the microprocessor's suspend, resume, reset, etc. by configuring the dmcontrol register. The RISC-V standard defines three types of abstract commands: access register, fast access, and access memory. QingKe V3A microprocessor only supports register access, other models support register and memory access, but not fast access. Access to registers (GPRs, CSRs) and continuous access to memory can be realized by abstract commands.
The debug module implements 8 instruction cache registers progbuf0-7, and the debug host can cache multiple

instructions (which can be compressed instructions) to the buffer, and can choose to continue executing the instructions in the instruction cache registers after executing the abstract command, or execute the cached instructions directly. Note that the last instruction in the progbufs needs to be an "ebreak" or "c.ebreak" instruction. Access to storage, peripherals, etc. is also possible through abstract commands and instructions cached in the progbufs.

Each register is described in detail as follows.

Data register 0 (data0)

Table 7-2 data0 register definition

Bit	Name	Access	Description	Reset Value
[31:0]	data0	RW	Data register 0, used for temporary storage of data	0

Data register 1 (data1)

Table 7-3 data1 register definition

Bit	Name	Access	Description	Reset Value
[31:0]	data1	RW	Data register 1, used for temporary storage of data	0

Debug module control register (dmcontrol)

This register controls the pause, reset, and resume of the microprocessor. Debug host write data to the corresponding field to achieve pause (haltreq), reset (ndmreset), resume (resumereq). You describe into the following.

Table 7-4 dmcontrol register definition

Bit	Name	Access	Description	Reset Value
31	haltreq	WO	0: Clear the pause request 1: Send a pause request	0
30	resumereq	W1	0: Invalid 1: Restore the current microprocessor *Note: Write 1 is valid and the hardware is cleared after the microprocessor is recovered*	0
29	Reserved	RO	Reserved	0
28	ackhavereset	W1	0: Invalid 1: Clear the haverest status bit of the microprocessor	0
[27:2]	Reserved	RO	Reserved	0
1	ndmreset	RW	0: Clear reset 1: Reset the entire system other than the debug module	0
0	dmactive	RW	0: Reset debug module 1: Debug module works properly	0

Debug module status register (dmstatus)

This register is used to indicate the status of the debug module and is a read-only register with the following description of each bit.

Table 7-5 dmstatus register definition

Bit	Name	Access	Description	Reset Value
[31:20]	Reserved	RO	Reserved	0
19	allhavereset	RO	0: Invalid 1: Microprocessor reset	0
18	anyhavereset	RO	0: Invalid 1: Microprocessor reset	0
17	allresumeack	RO	0: Invalid 1: Microprocessor reset	0
16	anyresumeack	RO	0: Invalid 1: Microprocessor reset	0
[15:14]	Reserved	RO	Reserved	0
13	allavail	RO	0: Invalid 1: Microprocessor is not available	0
12	anyavail	RO	0: Invalid 1: Microprocessor is not available	0
11	allrunning	RO	0: Invalid 1: Microprocessor is running	0
10	anyrunning	RO	0: Invalid 1: Microprocessor is running	0
9	allhalted	RO	0: Invalid 1: Microprocessor is in suspension	0
8	anyhalted	RO	0: Invalid 1: Microprocessor out of suspension	0
7	authenticated	RO	0: Authentication is required before using the debug module 1: The debugging module has been certified	0x1
[6:4]	Reserved	RO	Reserved	0
[3:0]	version	RO	Debugging system support architecture version 0010: V0.13	0x2

Microprocessor status register (hartinfo)

This register is used to provide information about the microprocessor to the debug host and is a read-only register with each bit described as follows.

Table 7-6 hartinfo register definition

Bit	Name	Access	Description	Reset Value
[31:24]	Reserved	RO	Reserved	0
[23:20]	nscratch	RO	Number of dscratch registers supported	0x3
[19:17]	Reserved	RO	Reserved	0
16	dataaccess	RO	0: Data register is mapped to CSR address 1: Data register is mapped to memory address	0x1
[15:12]	datasize	RO	Number of data registers	0x2
[11:0]	dataaddr	RO	The offset address of the data register data0, whose base address is 0xe0000000, is subject to specific reading.	0xXXX

Abstract command control and status registers (abstractcs)

This register is used to indicate the execution of the abstract command. The debug host can read this register to know whether the last abstract command is executed or not, and can check whether an error is generated during the execution of the abstract command and the type of the error, which is described in detail as follows.

Table 7-7 abstractcs register definitions

Bit	Name	Access	Description	Reset Value
[31:29]	Reserved	RO	Reserved	0
[28:24]	progbufsize	RO	Indicates the number of program buffer program cache registers	0x8
[23:13]	Reserved	RO	Reserved	0
12	busy	RO	0: No abstract command is executing 1: There are abstract commands being executed *Note: After execution, the hardware is cleared.*	0
11	Reserved	RO	Reserved	0
[10:8]	cmder	RW	Abstract command error type 000: No error 001: Abstract command execution to write to command, abstractcs, abstractauto registers or read and write to data and progbuf registers 010: Does not support current abstract command 011: Execution of abstract command with exception 100: The microprocessor is not suspended or unavailable and cannot execute abstract commands 101: Bus error 110: Parity bit error during communication 111: Other errors *Note: For bit writing 1 is used to clear the zero.*	0
[7:4]	Reserved	RO	Reserved	0
[3:0]	datacount	RO	Number of data registers	0x2

Abstract command register(command)

Debugging host can access GPRs, CSRs registers and memory by writing different configuration values into the abstract command register.

When accessing the registers, the command register bits are defined as follows.

Table 7-8 Definition of command register when accessing registers

Bit	Name	Access	Description	Reset Value
[31:24]	cmdtype	WO	Abstract command type 0: Access register; 1: Quick access (not supported); 2: Access to memory.	0
23	Reserved	WO	Reserved	0
[22:20]	aarsize	WO	Access register data bit width 000: 8-bit 001: 16-bit 010: 32-bit	0

			011: 64-bit (not supported) 100: 128-bit (not supported) *Note: When accessing floating-point registers FPRs, only 32-bit access is supported.*	
19	aarpostincrement	WO	0: No effect 1: Automatically increase the value of regno after accessing the register	0
18	postexec	WO	0: No effect 1：Execute the abstract command and then execute the command in progbuf	0
17	transfer	WO	0: Do not execute the operation specified by write 1: Execute the manipulation specified by write	0
16	write	WO	0: Copy data from the specified register to data0 1: Copy data from data0 register to the specified register	0
[15:0]	regno	WO	Specify access registers 0x0000-0x0fff are CSRs 0x1000-0x101f are GPRs	0

When accessing the memory of, the bits in the command register are defined as follows:

Table 7-9 Definition of command Register when Accessing Memory

Bit	Name	Access	Description	Reset Value
[31:24]	cmdtype	WO	Abstract command type 0: Access register; 1: Fast access (not supported); 2: Access memory.	0
23	aamvirtual	WO	0: Access physical address; 1: Access virtual address.	0
[22:20]	aamsize	WO	Access memory data bit width 000: 8-bit; 001: 16-bit; 010: 32-bit; 011: 64-bit (not supported); 100: 128-bit (not supported);	0
19	aampostincrement	WO	0: No influence; 1: After accessing the memory successfully, increase the address stored in the data1 register by the number of bytes corresponding to the bit width configured by aamsize. Aamsize=0, accessed by byte, data1 plus 1. Aamsize=1, accessed by half-word, data1 plus 2. aamsize=2, accessed by bit, data1 plus 4.	0
18	postexec	WO	0: No influence; 1: Execute the command in progbuf after executing the abstract command.	0

17	Reserve	RO	Reserved	0
16	write	WO	0: Read data from the address specified by data1 to data0 1: Write data in data0 to the address specified by data1.	0
[15:14]	target-specific	WO	Definition of reading and writing mode Write: 00, 01: Write directly to the memory; 10: After the data in data0 is OR with the data bits in the memory, the result is written into the memory (Only word access is supported). 11: After summing the data in data0 with the data bits in the memory, write the result into the memory (Only word access is supported). Read: 00, 01, 10, 11: Read 0 directly from the memory.	0
[13:0]	Reserve	RO	Reserved	

Abstract command automatic execution register (abstractauto)

This register is used to configure the debugging module. When reading and writing progbufx and datax of the debugging module, the abstract command can be executed again. The description of this register is as follows:

Table 7-10 abstractauto register definition

Bit	Name	Access	Description	Reset Value
[31:16]	autoexecprogbuf	RW	If a bit is set, the corresponding reading and writing of progbufx will cause the abstract command in the command register to be executed again. *Note: V3 series is designed with 8 progbufs, corresponding to bits [23:16].*	0
[15:12]	Reserve	RO	Reserved	0
[11:0]	autoexecdata	RW	If a bit is set to 1, the corresponding reading and writing of the datax register will cause the abstract command in the Command register to be executed again. *Note: V3 series is designed with two data registers, corresponding to bits [1:0].*	0

Instruction cache register (progbufx)

This register is used to store any instruction, deploy the corresponding operation, including 8, need to pay attention to the last execution needs to be "ebreak" or "c.ebreak".

Table 7-11 progbuf register definition

Bit	Name	Access	Description	Reset Value
[31:0]	progbuf	RW	Instruction encoding for cache operations, which may include compression instructions	0

Pause status register (haltsum0)

This register is used to indicate whether the microprocessor is suspended or not. Each bit indicates the suspended status of a microprocessor, and when there is only one core, only the lowest bit of this register is used to indicate it.

Table 7-12 haltsum0 register definition

Bit	Name	Access	Description	Reset Value
[31:1]	Reserved	RO	Reserved	0
0	haltsum0	RO	0: Microprocessor operates normally 1: Microprocessor stop	0

In addition to the above-mentioned registers of the debug module, the debug function also involves some CSR registers, mainly the debug control and status register dcsr and the debug instruction pointer dpc, which are described in detail as follows.

Debug control and status register (dcsr)

Table 7-13 dcsr register definition

Bit	Name	Access	Description	Reset Value
[31:28]	xdebugver	DRO	0000: External debugging is not supported 0100: Support standard external debugging 1111: External debugging is supported, but does not meet the specification	0x4
[27:16]	Reserved	DRO	Reserved	0
15	ebreakm	DRW	0: The ebreak command in machine mode behaves as described in the privilege file 1: The ebreak command in machine mode can enter debug mode	0
[14:13]	Reserved	DRO	Reserved	0
12	ebreaku	DRW	0: The ebreak command in user mode behaves as described in the privilege file 1: The ebreak command in user mode can enter debug mode	0
11	stepie	DRW	0: Interrupts are disabled under single-step debugging 1: Enable interrupts under single-step debugging	0
10	Reserved	DRO	Reserved	0
9	stoptime	DRW	0: System timer running in Debug mode 1: System timer stop in Debug mode	0
[8:6]	cause	DRO	Reasons for entering debugging 001: Entering debugging in the form of ebreak command (priority 3) 010: Entering debugging in the form of trigger module (priority 4, the highest) 011: Entering debugging in the form of pause request (priority 1) 100: debugging in the form of single-step debugging (priority 0, the lowest)	0

			101: enter debug mode directly after microprocessor reset (priority 2) Others: Reserved	
[5:3]	Reserved	DRO	Reserved	0
2	step	DRW	0: Turn off single-step debugging 1: Enable single-step debugging	0
[1:0]	prv	DRW	Privilege mode 00: User mode 01: Supervisor mode (not supported) 10: Reserved 11: Machine mode *Note: Record the privileged mode when entering debug mode, the debugger can modify this value to modify the privileged mode when exiting debug*	0

Debug mode program pointer (dpc)

This register is used to store the address of the next instruction to be executed after the microprocessor enters debug mode, and its value is updated with different rules depending on the reason for entering debug. dpc register is described in detail as follows.

Table 7-14 dpc register definitions

Bit	Name	Access	Description	Reset Value
[31:0]	dpc	DRW	Instruction address	0

The rules for updating the registers are shown in the following table.

Table 7-15 dpc update rules

Enter the debugging method	dpc Update rules
ebreak	Address of the Ebreak instruction
single step	Instruction address of the next instruction of the current instruction
trigger module	Temporarily not supported
halt request	Address of the next instruction to be executed when entering Debug

7.2 Debug Interface

Different from the standard JTAG interface defined by RISC-V, QingKe V3 series microprocessor adopts 1-wire/2-wire serial debug interface and follows WCH debug interface protocol V1.0. The debug interface is responsible for the communication between the debug host and the debug module, and realizes the read/write operation of the debug host to the debug module registers. WCH designed WCH_Link and open source its schematic and program binary files, which can be used for debugging all microprocessors of RISC-V architecture.

Refer to WCH Debug Protocol Manual for specific debug interface protocols.

Chapter 8 CSR Register List

The RISC-V architecture defines a number of Control and Status Registers (CSRs) for controlling and recording the operating status of the microprocessor. Some of the CSRs have been introduced in the previous section, and this chapter will detail the CSR registers implemented in the QingKe V3 series microprocessors.

8.1 CSR Register List

Table 8-1 List of Microprocessor CSR Registers

Type	Name	CSR Address	Access	Description
RISC-V Standard CSR	marchid	0xF12	MRO	Architecture number register
	mimpid	0xF13	MRO	Hardware implementation numbering register
	mstatus	0x300	MRW	Status register
	misa	0x301	MRW	Hardware instruction set register
	mtvec	0x305	MRW	Exception base address register
	mscratch	0x340	MRW	Machine mode staging register
	mepc	0x341	MRW	Exception program pointer register
	mcause	0x342	MRW	Exception cause register
	mtval	0x343	MRW	Exception value register
	pmpcfg<i>	0x3A0+i	MRW	PMP configuration register
	pmpaddr<i>	0x3B0+i	MRW	PMP address register
	tselect	0x7A0	MRW	Debug trigger selection register
	tdata1	0x7A1	MRW	Debug trigger data register 1
	tdata2	0x7A2	MRW	Debug trigger data register 2
	dcsr	0x7B0	DRW	Debug control and status registers
	dpc	0x7B1	DRW	Debug mode program pointer register
	dscratch0	0x7B2	DRW	Debug mode staging register 0
	dscratch1	0x7B3	DRW	Debug mode staging register 1
Vendor defined CSR	gintenr	0x800	URW	Global interrupt enable register
	intsyscr	0x804	URW	Interrupt system control register
	corecfgr	0xBC0	MRW	Microprocessor configuration register
	inestcr	0xBC1	MRW	Interrupt nested control register

8.2 RISC-V Standard CSR Registers

Architecture number register (marchid)

This register is a read-only register to indicate the current microprocessor hardware architecture number, which is mainly composed of vendor code, architecture code, series code, and version code. Each of them is defined as follows.

Table 8-2 marchid register definition

Bit	Name	Access	Description	Reset Value
31	Reserved	MRO	Reserved	1
[30:26]	Vender0	MRO	Manufacturer code 0 Fixed to the letter "W" code	0x17
[25:21]	Vender1	MRO	Manufacturer code1 Fixed to the letter "C" code	0x03
[20:16]	Vender2	MRO	Manufacturer code 2 Fixed to the letter "H" code	0x08
15	Reserved	MRO	Reserved	1
[14:10]	Arch	MRO	Architecture code	0x16

			RISC-V architecture is fixed to the letter "V" code	
[9:5]	Serial	MRO	Series code QingKe V3 series, fixed to the number "3"	0x03
[4:0]	Verision	MRO	Version code Can be the version "A", "B", "C" and other letters of the code	x

The manufacturer number and version number are alphabetic, and the series number is numeric. The coding table of letters is shown in the following table.

Table 8-3 Alphabetic Mapping Table

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Among them, QingKe V3A microprocessor, the register reads back to 0.

Hardware implementation numbering register (mimpid)

This register is mainly composed of vendor codes, each of which is defined as follows.

Table 8-4 mimpid register definition

Bit	Name	Access	Description	Reset Value
31	Reserved	MRO	Reserved	1
[30:26]	Vender0	MRO	Manufacturer code 0 Fixed to the letter "W" code	0x17
[25:21]	Vender1	MRO	Manufacturer code1 Fixed to the letter "C" code	0x03
[20:16]	Vender2	MRO	Manufacturer code 2 Fixed to the letter "H" code	0x08
15	Reserved	MRO	Reserved	1
[14:8]	Reserved	MRO	Reserved	0
[7:4]	Minor	MRO	Subversion number	0xX
[3:0]	Major	MR0	Major version number	0xX

This register is readable in any machine implementation, and in the QingKe V3A series processor, this register reads back to zero.

Machine mode status register (mstatus)

This register has been partially described in the previous section, and its folks are positioned as follows.

Table 8-5 mstatus register definition

Bit	Name	Access	Description	Reset Value
[31:13]	Reserved	MRO	Reserved	0
[12:11]	MPP	MRW	Privileged mode before entering break	0
[10:8]	Reserved	MRO	Reserved	0
7	MPIE	MRW	Interrupt enable state before entering interrupt	0
[6:4]	Reserved	MRO	Reserved	0
3	MIE	MRW	Machine mode interrupt enable	0
[2:0]	Reserved	MRO	Reserved	0

The MPP field is used to save the privileged mode before entering the exception or interrupt, and is used to restore the privileged mode after exiting the exception or interrupt. MIE is the global interrupt enable bit, and

when entering the exception or interrupt, the value of MPIE is updated to the value of MIE, and it should be noted that in the QingKe V3 series microprocessors, MIE will not be updated to 0 before the last level of nested interrupts to ensure that the interrupt nesting in Machine mode continues to be executed. When an exception or interrupt is exited, the microprocessor reverts to the Machine mode saved by MPP and the MIE is restored to the MPIE value.

QingKe V3 microprocessor supports Machine mode and User mode, if you need to make the microprocessor only work in Machine mode, you can set the MPP to 0x3 in the initialization of the boot file, that is, after returning, it will always remain in Machine mode.

Hardware instruction set register (misa)

This register is used to indicate the architecture of the microprocessor and the supported instruction set extensions, each of which is described as follows.

Table 8-6 misa register definition

Bit	Name	Access	Description	Reset Value
[31:30]	MXL	MRO	Machine word length 1:32 2:64 3:128	1
[29:26]	Reserved	MRO	Reserved	0
[25:0]	Extensions	MRO	Instruction set extensions	x

The MXL is used to indicate the word length of the microprocessor, QingKe V3 are 32-bit microprocessors, the domain is fixed to 1. Extensions are used to indicate that the microprocessor supports extended instruction set details, each indicates a class of extensions, its detailed description is shown in the following table.

Table 8-7 Instruction Set Extension Details

Bit	Name	Description
0	A	Atomic extension
1	B	Tentatively reserved for Bit-Manipulation extension
2	C	Compressed extension
3	D	Double-precision floating-point extension
4	E	RV32E base ISA
5	F	Single-precision floating-point extension
6	G	Additional standard extensions present
7	H	Hypervisor extension
8	I	RV32I/64I/128I base ISA
9	J	Tentatively reserved for Dynamically Translated Languages extension
10	K	Reserved
11	L	Tentatively reserved for Decimal Floating-Point extension
12	M	Integer Multiply/Divide extension
13	N	User-level interrupts supported
14	O	Reserved
15	P	Tentatively reserved for Packed-SIMD extension
16	Q	Quad-precision floating-point extension
17	R	Reserved
18	S	Supervisor mode implemented

19	T	Tentatively reserved for Transactional Memory extension
20	U	User mode implemented
21	V	Tentatively reserved for Vector extension
22	W	Reserved
23	X	Non-standard extensions present
24	Y	Reserved
25	Z	Reserved

For example, for QingKe V3A microprocessor, the register value is 0x401001105, which means that the supported instruction set architecture is RV32IMAC, and it has User mode implementation.

Machine mode exception base address register (mtvec)

This register is used to store the base address of the exception or interrupt handler and the lower two bits are used to configure the mode and identification method of the vector table as described in Section 3.2.

Machine mode staging register (mscratch)

Table 8-8 mscratch register definitions

Bit	Name	Access	Description	Reset Value
[31:0]	mscratch	MRW	Data storage	0

This register is a 32-bit readable and writable register in machine mode for temporary data storage. For example, when entering an exception or interrupt handler, the user stack pointer SP is stored in this register and the interrupt stack pointer is assigned to the SP register. After exiting the exception or interrupt, restore the value of user stack pointer SP from mscratch. That is, the interrupt stack and user stack can be isolated.

Machine mode exception program pointer register (mepc)

Table 8-9 mepc register definitions

Bit	Name	Access	Description	Reset Value
[31:0]	mepc	MRW	Exception procedure pointer	0

This register is used to save the program pointer when entering an exception or interrupt. It is used to save the instruction PC pointer before entering an exception when an exception or interrupt is generated, and mepc is used as the return address when the exception or interrupt is handled and used for exception or interrupt return. However, it is important to note that.

- When an exception occurs, mepc is updated to the PC value of the instruction currently generating the exception.
- When an interrupt occurs, mepc is updated to the PC value of the next instruction.

When you need to return an exception after processing the exception, you should pay attention to modifying the value of the mepc, and more details can be found in Chapter 2 Exceptions.

Machine mode exception cause register (mcause)

Table 8-10 mcause register definition

Bit	Name	Access	Description	Reset Value
31	Interrupt	MRW	Interrupt indication field 0: Exception 1: Interruption	0
[30:0]	Exception Code	MRW	Exception codes, see Table 2-1 for details	0

This register is mainly used to store the cause of the exception or the interrupt number of the interrupt. Its

highest bit is the Interrupt field, which is used to indicate whether the current occurrence is an exception or an interrupt. The lower bit is the exception code, which is used to indicate the specific cause. Its details can be found in Chapter 2 Exceptions.

Machine mode exception value register (mtval)

Table 8-11 mtval register definition

Bit	Name	Access	Description	Reset Value
[31:0]	mtval	MRW	Exception value	0

This register is used to hold the value that caused the exception when an exception occurs. For details such as the value and time of its storage, please refer to Chapter 2 Exceptions.

PMP configuration register (pmpcfg<i>)

This register is mainly used to configure the physical memory protection unit, and every 8 bits of this register are used to configure the protection of an area. Please refer to Chapter 4 for the detailed definition.

PMP address register (pmpaddr<i>)

This register is mainly used for the address configuration of the physical memory protection unit, which encodes the upper 32 bits of a 34-bit physical address. Please refer to Chapter 4 for the specific configuration method.

Debug mode program pointer register (dpc)

This register is used to store the address of the next instruction to be executed after the microprocessor enters Debug mode, and its value is updated with different rules depending on the reason for entering debug. Refer to Section 6.1 for detailed description.

Debug trigger select register (tselect)

It is only valid for microprocessors that support hardware breakpoints, and supports 4-channel breakpoints at most, and its lower 2 bits are valid. When configuring each channel breakpoint, you need to select the corresponding channel through this register before configuration.

Table 8-12 tselect register definition

Bit	Name	Access	Description	Reset Value
[31:2]	Reserved	MRO	Reserved	0
[1:0]	TSELECT	MRW	The breakpoint channel selection register is configured, that is, after the corresponding channel is selected, the tdata1 and tdata2 registers can be operated to configure breakpoint information.	X

Debug trigger data register 1(tdata1)

It is only valid for microprocessors that support hardware breakpoints. Microprocessors only support instruction address and data address breakpoints, where the bit TYPE of the tdata1 register is a fixed value of 2, and other bits conform to the definition of mcontrol in the debugging standard.

Table 8-13 tdata1 register definition

Bit	Name	Access	Description	Reset Value
[31:28]	TYPE	MRO	Breakpoint type definition, mcontrol type.	0x2

27	DMODE	MRO	0: The relevant registers of the flip-flop can be modified in both machine mode and debugging mode; 1: Only debug mode can modify the relevant registers of the flip-flop.	1
[26:21]	MASKMAX	MRO	When MATCH=1, the maximum exponential power range of matching is allowed, that is, the maximum allowable matching range is 2^{31} bytes.	0x1F
[20:13]	Reserved	MRO	Reserved	0
12	ACTION	MRW	Set the processing mode when triggering a breakpoint: 0: When triggering, enter the breakpoint and call back the interrupt; 1: Enter debugging mode when triggered.	0
[11:8]	Reserved	MRO	Reserved	0
7	MATCH	MRW	Matching policy configuration: 0: Match when the trigger value is equal to TDATA2; 1: The trigger value matches the high m bit of TDATA2, where m = 31–n, and n is the first 0 quote of TDATA2 (starting from the low bit).	0
6	M	MRW	Enable flip-flop in M mode: 0: Disable the trigger in M mode; 1: Enable the trigger in M mode.	0
[5:4]	Reserved	MRO	Reserved	0
3	U	MRW	Enable trigger in U mode: 0: Disable the trigger in U mode; 1: Enable the trigger in U mode.	0
2	EXECUTE	MRW	Instruction read address trigger enabled: 0: Disable; 1: Enable.	0
1	STORE	MRW	Data write address trigger enabled: 0: Disable; 1: Enable.	0
0	LOAD	MRW	Data read address trigger enabled: 0: Disable; 1: Enable.	0

Debug trigger data register 2(tdata2)

It is only valid for microprocessors that support hardware breakpoints, and is used to save the matching value of the trigger.

Table 8-14 tdata2 register definition

Bit	Name	Access	Description	Reset Value
[31:0]	TDATA2	MRW	Used to save matching values.	X

Debug control and status register (dcsr)

This register is used to control and record the running state of the debugging mode. Refer to Section 7.1 for details.

Debug mode program pointer (dpc)

This register is used to store the address of the next instruction to be executed after the microprocessor enters the debugging mode, and its value is different according to the reasons for entering the debugging mode, and the updating rules are also different. Refer to Section 7.1 for detailed description.

Debug mode staging register (dscratch0-1)

This group of registers is used for temporary storage of data in Debug mode.

Table 8-15 dscratch0-1 register definitions

Bit	Name	Access	Description	Reset Value
[31:0]	dscratch	DRW	Debug mode data staging value	0

8.3 User-defined CSR Register

User mode global interrupt enable register (gintenr)

This register is used to control the enable and mask of global interrupt. The enable and mask of global interrupt in machine mode can be controlled by the MIE and MPIE bits in mstatus, but this register cannot be operated in user mode. While the global interrupt enable register gintenr is the mapping of MIE and MPIE in mstatus. In user mode, gintenr can be used to set and clear MIE and MPIE, as described in Section 3.2 for details.

Note: Global interrupts do not include unmasked interrupts NMI and exceptions.

Interrupt system control register (intsyscr)

This register is mainly used to configure interrupt nesting depth, hardware stack pressing and other related functions, as described in Section 3.2 for details.

Microprocessor configuration register (corecfgr)

This register is used to control whether the NMI interrupt is allowed after the interrupt overflows, and whether the interrupt request is cleared when the fence instruction is executed. Please refer to Section 3.2 for the specific definition.

Interrupt nested control register (inestcr)

This register is used to indicate the interrupt nesting state and whether it overflows or not, and to control the maximum nesting level. Please refer to Section 3.2 for the specific definition.