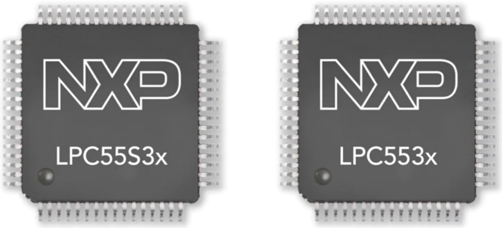
Manuals+ — User Manuals Simplified.



NXP AN13823 IEC 60730 Class B Software for LPC553x MCUs User Guide

AN13823 IEC 60730 Class B Software for LPC553x MCUs User Guide

Contents

AN13823 IEC 60730 Class B Software for LPC553x MCUs

Rev. 0 — 4 January 2023
Application note

Document information

Informatio n	Content
Keywords	LPC553x, AN13823, IEC 60730, LPC5536-EVK, IEC60730B
Abstract	The main purpose of this application note is to accelerate customer software development and c ertification processes for products based on LPC553x MCUs.

Introduction

The IEC 60730 safety standard defines the test and diagnostic methods that ensure the safe operation of embedded control hardware and software for household appliances.
To achieve functional safety, it is necessary to remove all risk of hazards that system malfunction may cause.
The IEC 60730 standard classifies the applicable equipment into three categories:

- Class A: Not intended to be relied upon for the safety of the equipment
- Class B: To prevent unsafe operation of the controlled equipment
- Class C: To prevent special hazards

NXP provides IEC 60730 safety Class B library to help manufacturers of automatic controls in the large appliance market meet the IEC 60730 class B regulation. The library supports the IAR, Keil, and MCUXpresso IDEs.
You can integrate NXP safety library binary into your application software. For easier development of the IEC60730B application, the library also provides an example project. This example is distributed through the **IEC 60730 Safety Standard for Household Appliances** on **nxp.com** website.

Library	Supported Devices	Documents	Example Projects
Filter by...	Filter by...	Filter by...	Filter by...
IEC60730B Safety library for CM33 version 4.2	LPC55Sxx LPC55xx	CM33 Safety Library user guide v4.2 Test report VDE Common part Test report VDE Core part Release notes	LPCxpresso55S36

The main purpose of this application note is to accelerate customer software development and certification processes for products based on LPC553x MCUs.

NXP IEC 60730 Class B library overview

The safety library includes core-dependent part and peripheral-dependent part self-tests as listed below:

- Core-dependent part
  - CPU registers test
  - CPU program counter test
  - Variable memory test
  - Invariable memory test
  - Stack test
- Peripheral-dependent part
  - Clock test
  - Digital input/output test
  - Analog input/output test
  - Watchdog test

Table 1. Compliance with IEC 60730 Class B standards

NXP IEC 60730 Class B Library		IEC 60730	
component	**Method**	**Items**	**Applied**
CPU registe rs	The CPU register test procedure tests all of the CM33 C PU registers for the stuck-at condition.	1.1 Register	H.2.16.6
Program co unter	The CPU program counter test procedure tests the CPU program counter register for the stuck-at condition. The program counter register test can be performed once af ter the MCU reset and also during runtime. Force the CPU (program flow) to access the corresponding address that is testing the pattern to verif y the program counter functionality.	1.3 Program counter	H.2.16.6
Clock	The clock test procedure tests the oscillators of the proc essor for the wrong frequency. The clock test principle i s based on the comparison of two independent clock so urces. If the test routine detects a change in the frequen cy ratio between the clock sources, a failure error code i s returned.	3.Clock	NA
Invariable m emory	The invariable memory test is to check whether there is a change in the memory content (on-chip Flash) during the application execution. Several checksum methods (for example, CRC16) can be used for this purpose.	4.1 Invariable memory	H.2.19.3.1
Variable me mory test	Checks the on-chip RAM for DC faults. The March C an d March X schemes are used as control mechanisms.	4.2 Variable memory	H.2.19.6
Digital input/ output test	The DIO test functions are designed to check the digital input and output functionality and short circuit condition s between the tested pin and the supply voltage, ground, or optional adjacent pin.	7.1 Digital I/O	H.2.18.13
Analog Inpu t/ Output (I/ 0) test	The test checks the analog input interface and three ref erence values: reference high, reference low, and band gap voltage. The analog input test is based on a conver sion of three analog inputs with known voltage values a nd it checks if the converted values fit into the specified limits. Normally, the limits should be roughly 10 % aroun d the desired reference values.	7.2 Analog I/O	H.2.18.13

NXP IEC 60730 Class B library example project

For easier development of the IEC60730B application, the library provides an example project framework, built upon a dedicated LPC553x evaluation board [Sign in to NXP.com | NXP Semiconductors](#) (LPC5536-EVK). You must configure the correct library settings for the actual project.

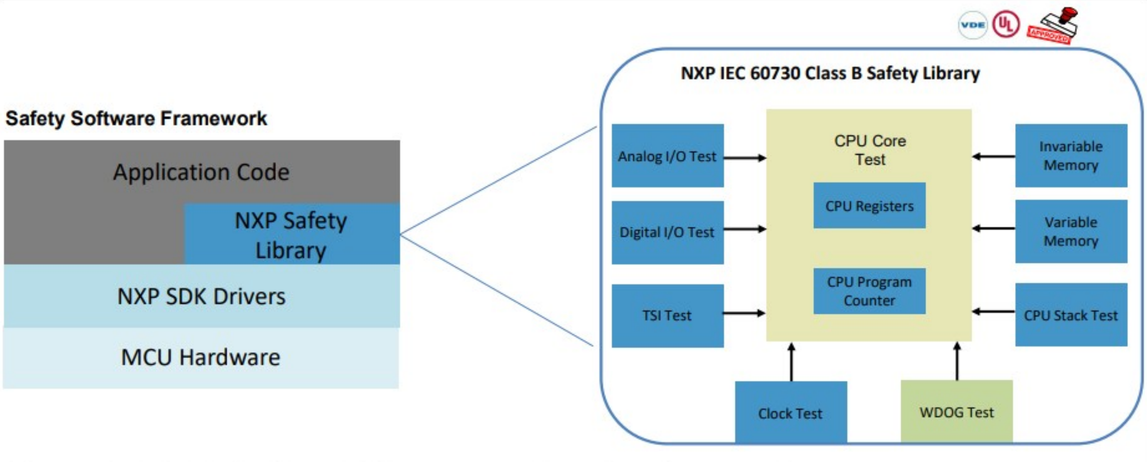


Figure 1. IEC 60730 Class B library example project framework

3.1 Integration of the safety library into the user application

The safety example project routines are divided into two main processes: pre-run one time safety test and runtime periodical safety test.

The following figure shows the safety test processes.

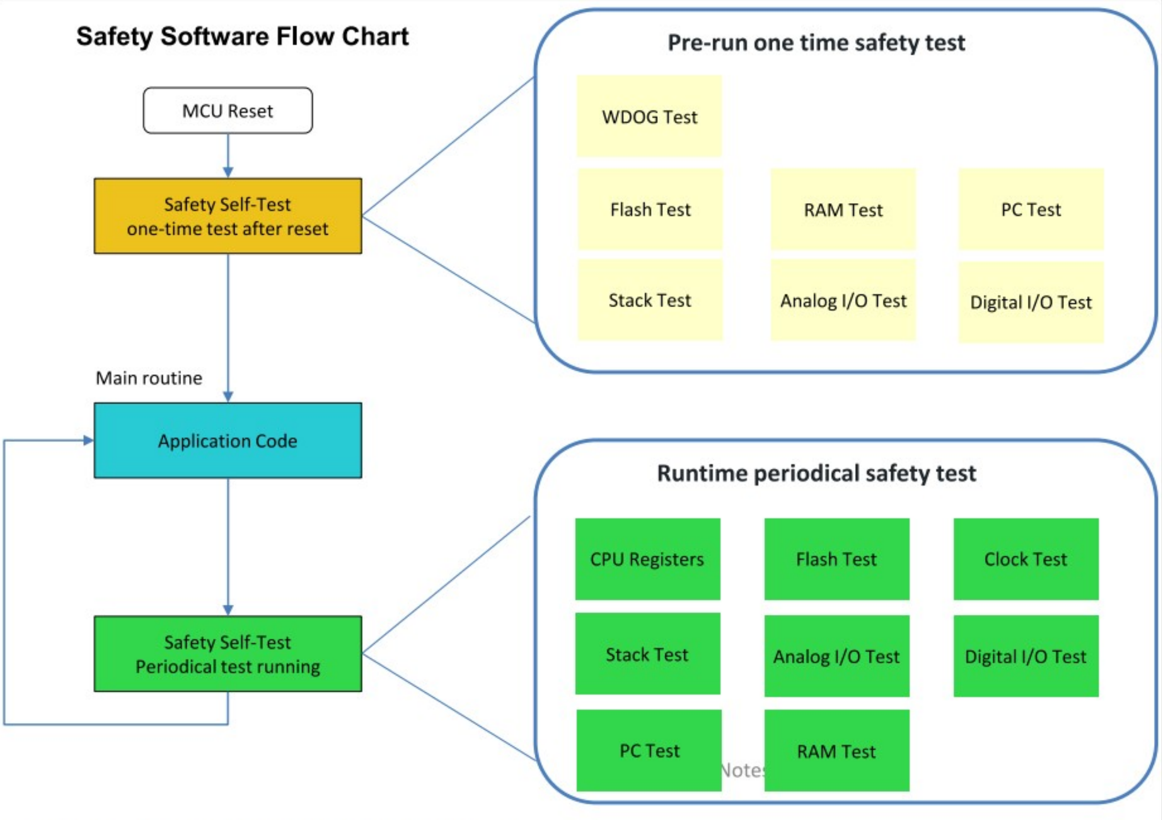


Figure 2. NXP safety software flowchart

To integrate NXP safety library, perform the following steps:

1. Download the safety example project from nxp.com
2. Hardware setting considering the peripherals used for the safety self-test
3. Configure the safety library according to the actual hardware design
4. Turn on the safety test functions one by one in safety_config.h
   - For debugging, it is better to turn the flash test and watchdog OFF first
   - Take care of the interrupts, as some of the safety tests cannot be interrupted
5. Develop the application code based on the safety example project framework

LPC553x safety library example project in practice

4.1 Hardware block diagram

The following modules are used for safety self-test by default as shown in the figure below:

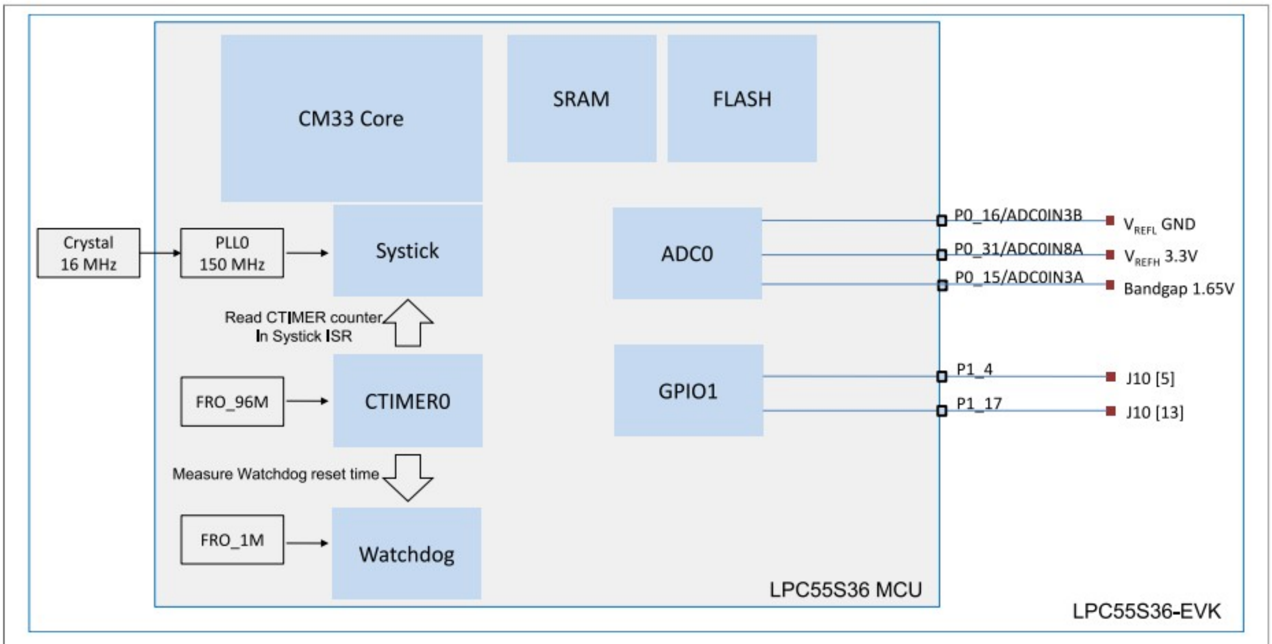


Figure 3. LPC553x safety library hardware block diagram

Table 2. MCU module for safety self-test

Safety library test item	MCU module
CPU test	LPC5536 CM33 Core
Clock test	Systick CTIMER0
Watchdog test	Watchdog CTIMER0
Variable memory test	SRAM
Invariable memory test	Flash
Digital I/O test	GPIO1
Analog I/O test	ADC0

4.2 CPU test

4.2.1 CPU registers test description

The CPU register test procedure tests all of the CM33 CPU registers for the stuckat condition (except for the program counter register). The program counter test is implemented as a standalone safety routine. This set of tests includes the test of the following registers:

- General-purpose registers:
  - R0-R12
- Stack pointer registers:
  - MSP + MSPLIM (secure / non-secure)
  - PSP + PSPLIM (secure / non-secure)
- Special registers:
  - APSR
  - CONTROL (secure / non-secure)

  - PRIMASK (secure / non-secure)
  - FAULTMASK (secure / non-secure)
  - BASEPRI (secure / non-secure)
- Link register:
  - LR
- FPU registers:
  - FPSCR
  - S0 – S31

There is a set of tests that are performed once after the MCU is reset and also during runtime. You can reuse the default settings of LPC553x safety library example project, however, you must pay attention to the interrupt as some of CPU register tests cannot be interrupted.

- Pre-run one time safety test
  - SafetyCpuAfterResetTest /* Interrupts must be disabled for a while */
  - FS_CM33_CPU_Register
  - FS_CM33_CPU_NonStackedRegister
  - FS_CM33_CPU_SPmain_S
  - FS_CM33_CPU_SPmain_Limit_S
  - FS_CM33_CPU_SPprocess_S
  - FS_CM33_CPU_SPprocess_Limit_S
  - FS_CM33_CPU_Primask_S
  - FS_FAIL_CPU_PRIMASK
  - FS_CM33_CPU_Special8PriorityLevels_S
  - FS_CM33_CPU_Control
  - FS_CM33_CPU_Float1
  - FS_CM33_CPU_Float2
- Runtime periodical safety test
  - SafetyCpuBackgroundTest /* Interruptible CPU registers test */
  - FS_CM33_CPU_Register
  - FS_CM33_CPU_NonStackedRegister
  - FS_CM33_CPU_Control /* Interrupts must be disabled for a while */
  - FS_CM33_CPU_SPprocess_S /* Interrupts must be disabled for a while */

4.3 CPU program counter test

4.3.1 CPU program counter test description

The CPU program counter register test procedure tests the CPU program counter register for the stuck-at condition. Contrary to the other CPU registers, the program counter cannot be simply filled with a test pattern. It is necessary to force the CPU (program flow) to access the corresponding address that is testing the pattern to verify the program counter functionality.
Note that the program counter test cannot be interrupted.

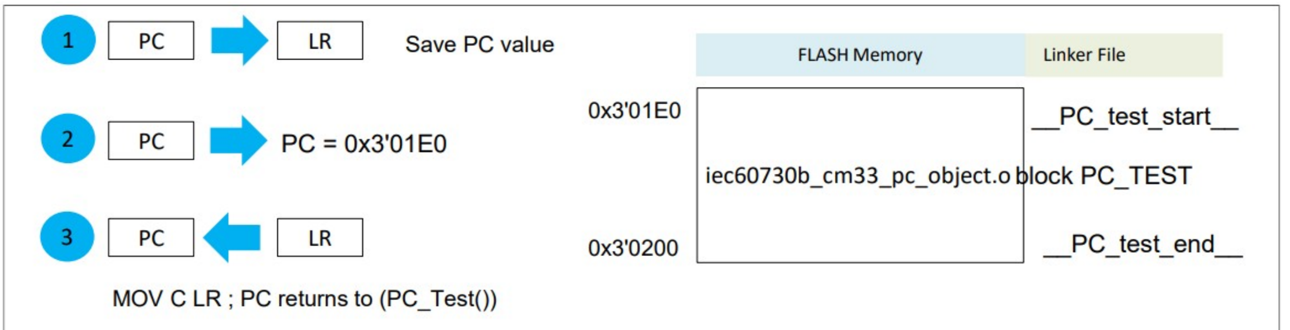


Figure 4. Example of CPU program counter test

The program counter register test can be performed once after the MCU is reset and also during runtime.

- Pre-run one time safety test
  - SafetyPcTest
  - FS_CM33_PC_Test
- Runtime periodical safety test
  - SafetyIsrFunction > SafetyPcTest
  - FS_CM33_PC_Test

4.4 Variable memory test

4.4.1 Variable memory test description

The variable memory test for supported devices checks the on-chip RAM for DC faults.

The application stack area can also be tested. The March C and March X schemes are used as control mechanisms.

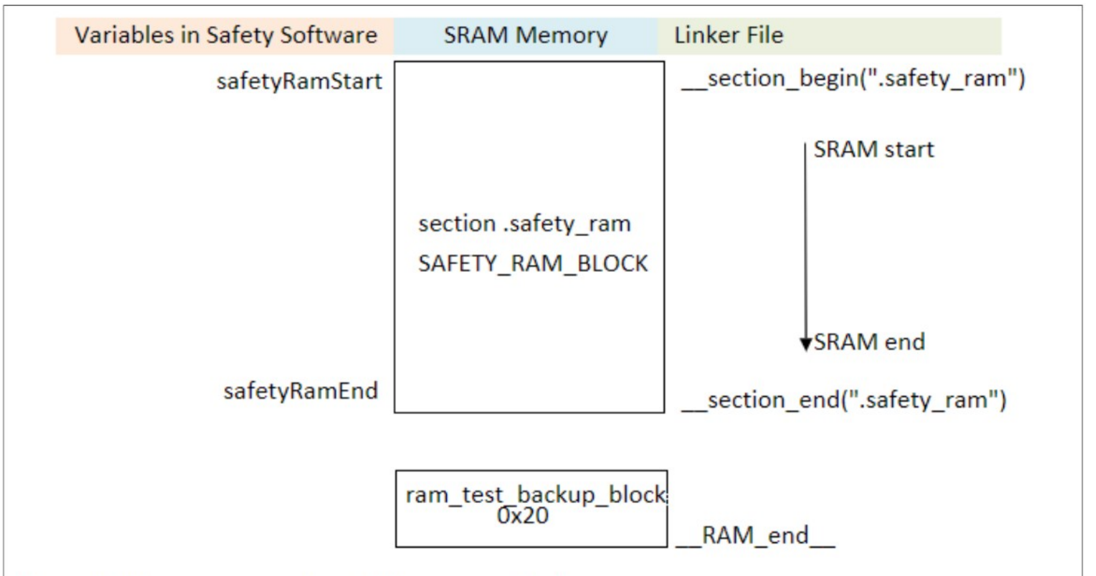


Figure 5. Memory map of variable memory test

The handling functions are different for the after-reset test and for the runtime test.

The after-reset test is done by the FS_CM33_RAM_AfterReset () function. This function is called once after the reset, when the execution time is not critical. Reserve free memory space for the backup area. The block size parameter cannot be larger than the size of the backup area. The function first checks the backup area, then the loop begins. Blocks of memory are copied to the backup area and their locations are checked by the respective March test. The data is copied back to the original memory area and the actual address with the block size is updated. This is repeated until the last block of memory is tested. If a DC fault is detected, the function returns a failure pattern.

The runtime test is done by the FS_CM33_RAM_Runtime () function. To save time, it only tests one segment (defined by RAM_TEST_BLOCK_SIZE) of SRAM on time. While the after-reset test checks the whole block of safety-related RAM space. In LPC553x safety library example project, RAM_TEST_BLOCK_SIZE is configured to 0x4, it means that 32 bytes of RAM will be tested in one runtime RAM test routine.

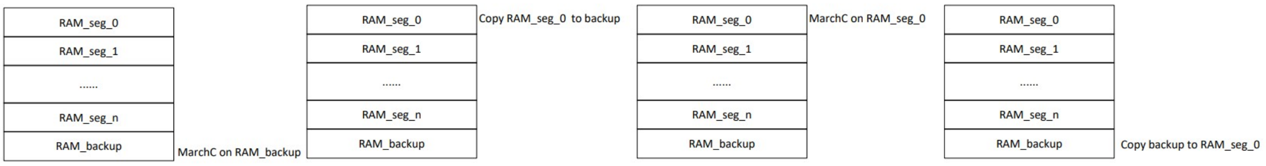


Figure 6. Example of variable memory test

- Pre-run one time safety test

  – SafetyRamAfterResetTest /* Test the whole RAM space of the section ".safety_ram" before running the main routine. */

  – FS_CM33_RAM_AfterReset
- Runtime periodical safety test

  – SafetyIsrFunction(&g_sSafetyCommon, &g_sSafetyRamTest, &g_sSafetyRamStackTest) /* executed in Systick ISR, cannot be interrupted */

  – FS_CM33_RAM_Runtime

4.4.2 Variable memory test configuration

The configuration of the variable memory test in <Safety_config.h>:

```
#define RAM_TEST_BLOCK_SIZE 0x4 /* size of block for runtime testing */

#if defined(__IAR_SYSTEMS_ICC__) || (defined(__GNUC__) && (__ARMCC_VERSION >= 6010050)) /* IAR + KEIL */
#define RAM_TEST_BACKUP_SIZE  0x20 /* must fit with the setup from linker configuration file */
```

The configuration of safety RAM block is in <lpcxpresso55s36_safety.icf>:
define block SAFETY_RAM_BLOCK with alignment = 8
{section .safety_ram };
place in RAM_region {block SAFETY_RAM_BLOCK};
Note that only the .safety_ram is covered by the variable memory test. Add the variables into the .safety_ram section manually, as shown below in main.c.

```
#if defined(__IAR_SYSTEMS_ICC__) /* IAR */
    #pragma section =  ".safety_ram"
    #pragma section = ".pctest"

    safety_common_t g_sSafetyCommon @ ".safety_ram";
    wd_test_t g_sSafetyWdTest @ ".safety_ram";
    fs_flash_runtime_test_parameters_t g_sFlashCrc @ ".safety_ram";
    fs_flash_configuration_parameters_t g_sFlashConfig @ ".safety_ram";
    fs_ram_test_t g_sSafetyRamTest @ ".safety_ram";
    fs_ram_test_t g_sSafetyRamStackTest @ ".safety_ram";
    fs_clock_test_t g_sSafetyClockTest @ ".safety_ram";
```

4.5 Invariable memory test

4.5.1 Invariable memory test description

The invariable memory on the LPC5536 MCU is the on-chip flash. The principle of the invariable memory test is to check whether there is a change in the memory content during the application execution. Several checksum methods can be used for this purpose. The checksum is an algorithm that calculates a signature of the data placed in the tested memory. The signature of this memory block is then periodically calculated and compared with the original signature.

The signature for the assigned memory is calculated in the linking phase of an application. The signature must be saved into the invariable memory, but in a different area than the one that the checksum is calculated for. In runtime and after the reset, the same algorithm must be implemented in the application to calculate the checksum. The results are compared. If they are not equal, a safety error state occurs.

When implemented after the reset or when there is no restriction on the execution time, the function call can be as follows.

• Pre-run one time safety test

– SafetyFlashAfterResetTest
– FS_FLASH_C_HW16_K /* calculate CRC of the whole Flash */
In the application runtime and with limited time for execution, the CRC is computed in a sequence. It means that the input parameters have different meanings in comparison with the calling after reset. The implementation example is as follows:
• Runtime periodical safety test
– SafetyFlashRuntimeTest
– FS_FLASH_C_HW16_K /* calculate CRC block by block */
– SafetyFlashTestHandling /* compare CRC when all Flash blocks are calculated. */

4.5.2 Invariable memory test configuration

In LPC553x safety library example project, the flash allocation is shown below as specified in the Linker file <lpcxpresso55s36_safety.icf>. The object files <main.o> and <safety_cm33_lpc.o> are placed in the safety flash block which is checked by the invariable memory test. You can put more object files into SAFETY_FLASH_BLOCK Flash area by modifying the Linker file accordingly.

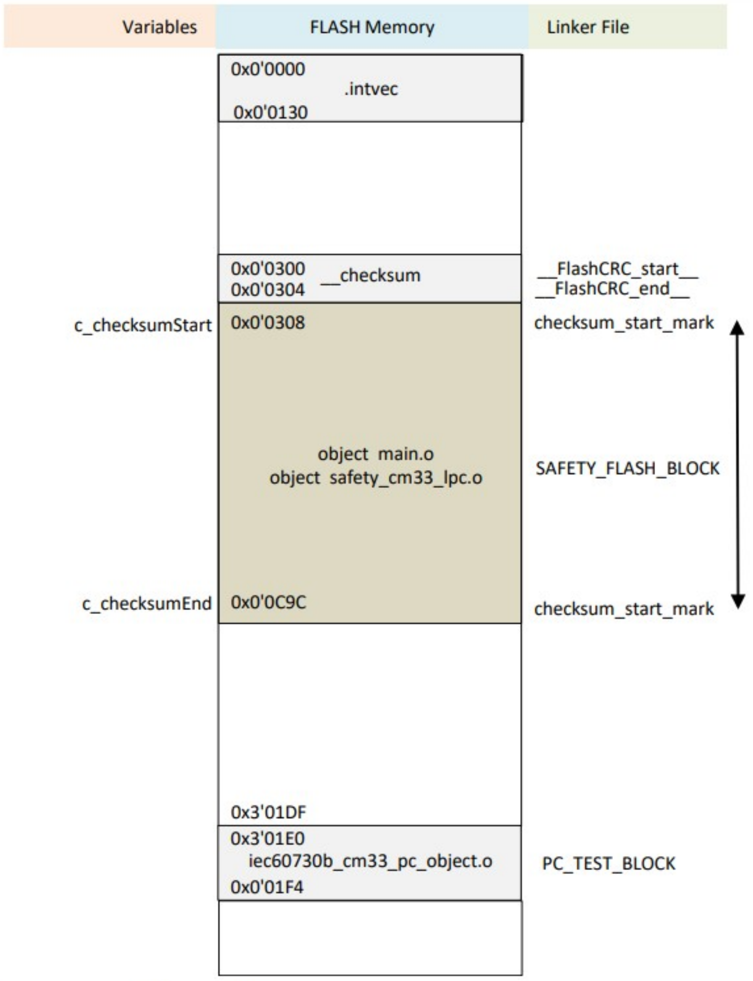


Figure 7. Memory map of invariable memory test

There are two checksums to be compared during the MCU runtime to verify whether the contents of the given flash space have been modified:

- Checksum calculated by Linker at Compiling/Linking
- Checksum calculated by MCU at runtime

Definition of the location to place the checksum result (pre-calculated by the linker tools) is in <lpcxpresso55s36_safety.icf>:
define symbol __FlashCRC_start__ = 0x0300; /* for placing a checksum */
define symbol __FlashCRC_end__ = 0x030F; /* for placing a checksum */
define region CRC_region = mem: [from __FlashCRC_start__ to __FlashCRC_end__];
define block CHECKSUM with alignment = 8 {section. checksum}; place in CRC_region { block CHECKSUM};
Take IAR IDE, for example, in the project option setting > Build Actions > Post-build command line.

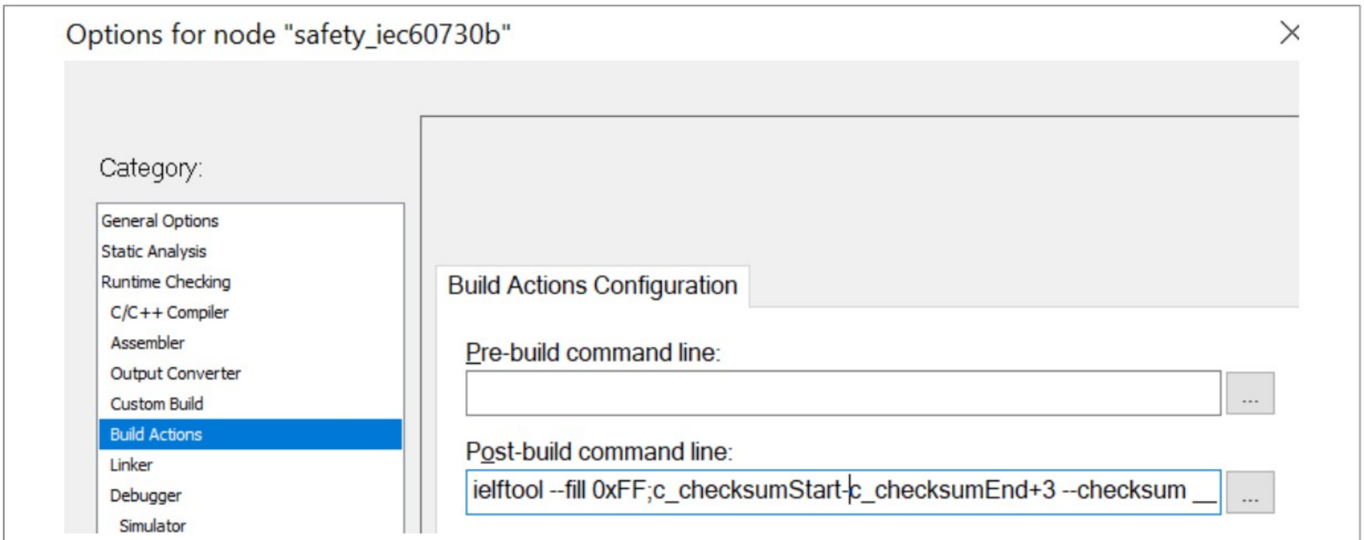


Figure 8. Example of checksum calculation by IAR

Command line:

ielftool –fill 0xFF;c_checksumStart-c_checksumEnd+3 –checksum __checksum:2,crc16,0x0;c_checksumStart-c_checksumEnd+3 –verbose “$TARGET_PATH$” “$TARGET_PATH$”

The linker calculates the original checksum of the flash addressing from _checksumStart to c_checksumEnd, then places the checksum result into _checksum, which is in block CHECKSUM defined by the Linker file.

Definition of the specified flash space to be checked is in **<lpcxpresso55s36_safety.icf>:**

define block SAFETY_FLASH_BLOCK with alignment = 8, fixed order { readonly section checksum_start_mark, section .text object main.o, section .text object safety_cm33_lpc.o, section .rodata object safety_cm33_lpc.o, readonly section checksum_end_mark };

place in ROM_region { block SAFETY_FLASH_BLOCK};

4.6 Stack test

4.6.1 Stack test description

The stack test is an additional test, not directly specified in the IEC60730 annex H table.

This test routine is used to test the overflow and underflow conditions of the application stack. The testing of the stuck-at faults in the memory area occupied by the stack is covered by the variable memory test. The overflow or underflow of the stack can occur if the stack is incorrectly controlled or by defining the “too-low” stack area for the given application.

The principle of the test is to fill the area below and above the stack with a known pattern. These areas must be defined in the linker configuration file, together with the stack. The initialization function then fills these areas with your pattern. The pattern must have a value that does not appear elsewhere in the application. The purpose is to check if the exact pattern is still written in these areas. If it is not, it is a sign of incorrect stack behavior. If this occurs, then the FAIL return value from the test function must be processed as a safety error.

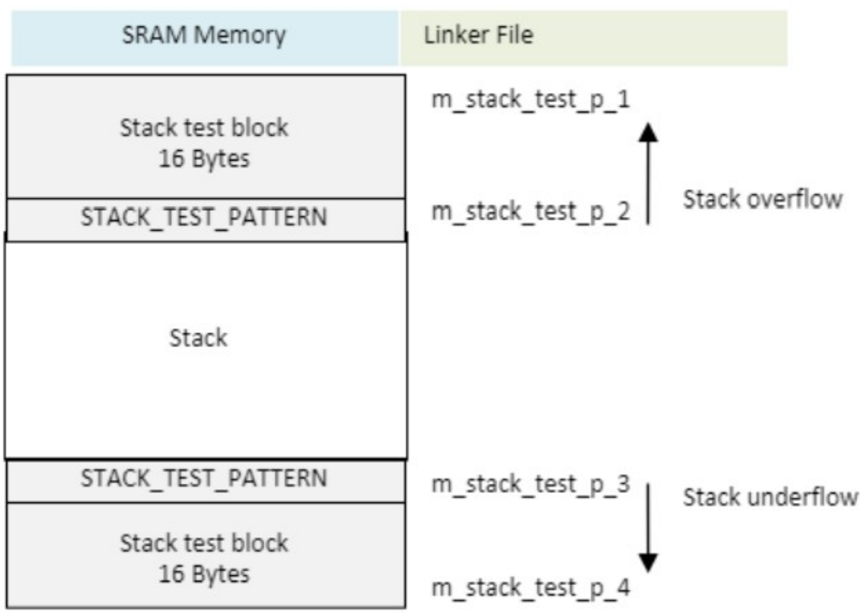


Figure 9. Memory map of stack test

The test is performed after the reset and during the application runtime in the same way.

- Pre-run one time safety test
  – SafetyStackTestInit
  – FS_CM33_STACK_Init /* write STACK_TEST_PATTERN (0x77777777) to STACK_TEST_BLOCK */
  – SafetyStackTest
  – FS_CM33_STACK_Test /* check the contents of STACK_TEST_BLOCK, failed if the value is not equal to STACK_TEST_PATTERN (0x77777777).
- Runtime periodical safety test
  – SafetyStackTest
  – FS_CM33_STACK_Init /* write STACK_TEST_PATTERN (0x77777777) to STACK_TEST_BLOCK */
  – SafetyStackTest
  – FS_CM33_STACK_Test /* check the contents of STACK_TEST_BLOCK, fails if the value is not equal to STACK_TEST_PATTERN (0x77777777)

4.6.2 Stack test configuration

The configuration of the stack test is in <Safety_config.h> and the linker file <lpcxpresso55s36_safety.icf>

```
#if defined(__IAR_SYSTEMS_ICC__) || (defined(__GNUC__) && (__ARMCC_VERSION >= 6010050)) /* IAR + KEIL */
#define RAM_TEST_BACKUP_SIZE  0x20 /* must fit with the setup from linker configuration file */
#define STACK_TEST_BLOCK_SIZE 0x10 /* must fit with the setup from linker configuration file */
#endif

#define STACK_TEST_PATTERN 0x77777777
```

```
define exported symbol m_stack_test_p_4     = m_safety_error_code - 0x4;
define exported symbol m_stack_test_p_3     = m_stack_test_p_4 - stack_test_block_size + 0x4;
define exported symbol __BOOT_STACK_ADDRESS = m_stack_test_p_3 - 0x4;
define exported symbol m_stack_test_p_2     = __BOOT_STACK_ADDRESS - __size_cstack__;
define exported symbol m_stack_test_p_1     = m_stack_test_p_2 - stack_test_block_size + 0x4;
```

4.7 Clock test

4.7.1 Clock test description

The clock test principle is based on the comparison of two independent clock sources.
In LPC553x safety library example project, CTIMER0 and Systick on MCU LPC5536 are used as two independent clocks for the safety clock test, they do not depend on the LPC5536-EVK hardware board.
The clock test routine is executed in the runtime periodical safety test only.

- Pre-run one time safety test
  – No clock test
- Runtime periodical safety test
  – SafetyClockTestCheck
  – SafetyClockTestIsr

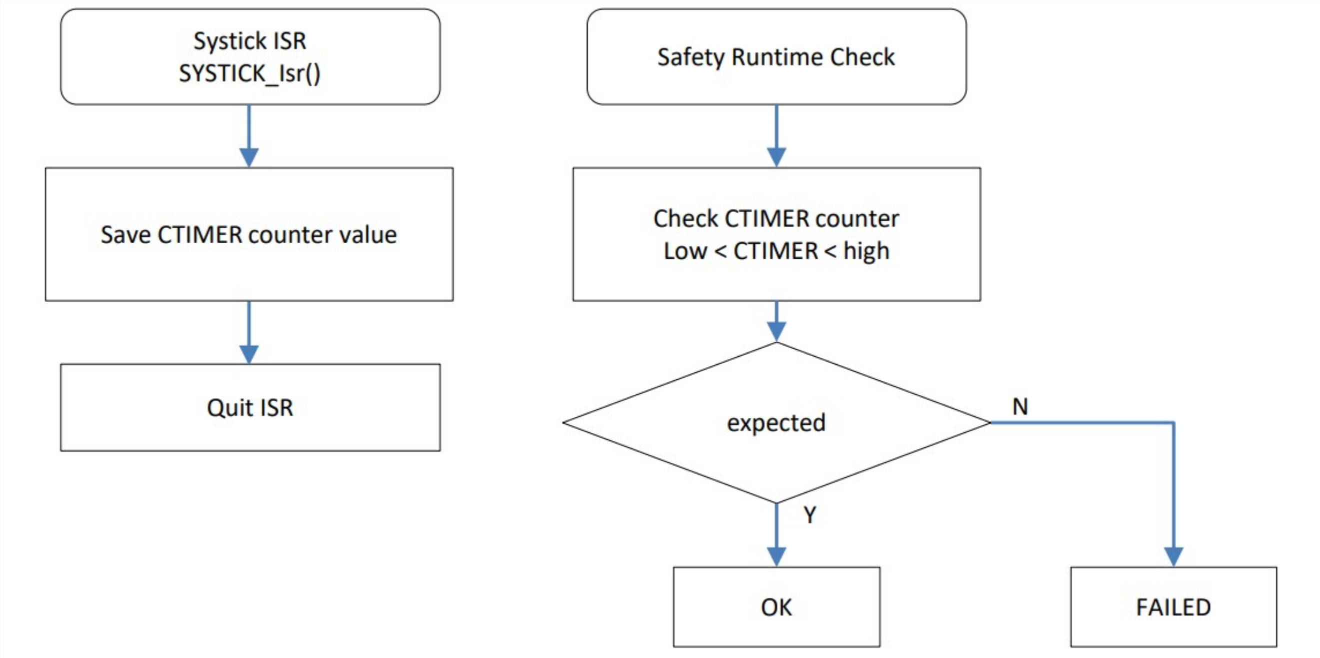


Figure 10. Example of clock test

4.7.2 Clock test configuration

As two independent clocks are required for the clock test in LPC553x safety library example project:

- SYSTICK timer is sourced from PLL0 150 M (sourced from the external 16 MHz crystal)
- CTIMER0 timer is sourced from the internal FRO_96M

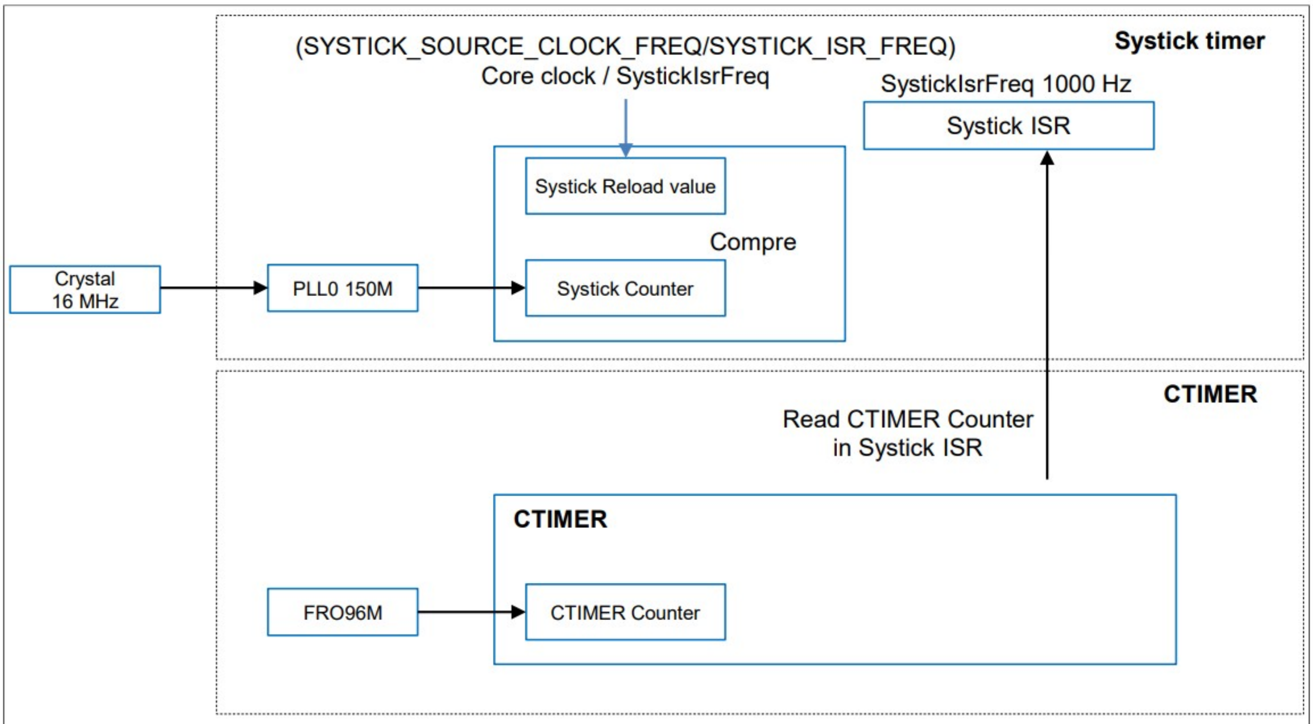


Figure 11. Configuration of clock test

The detailed configurations of the Systick and CTIMER0 are shown below:

- Systick config: SystickISR_Freq = 1000 Hz, by setting 150,000 reload value under 150 MHz core clock
- CTIMER config: CTIMER_Freq = 96 MHz, sourced from 96 MHz FRO_96M clock
- Expected CTIMER counter should be CTIMER _Freq/SystickISR_Freq = 96 MHz / 1000 = 96,000
- In each Systick interrupt ISR, save the CTIMER counter value

- In runtime while (1) loop, check: (96,000 – 20 %) < CTIMER expect counter < (96,000 + 20 %)

The configuration of the clock test is in Safety_config.h.
According to the actual application, you can change the CTIMER instance for the safety clock test by configuring REF_TIMER_USED macro. Also, you must configure REF_TIMER_CLOCK_FREQUENCY according to the actual clock frequency.

```
/* CLOCK test */
#define USE_WKT                     0 /*USE CTIMER = 0 USE WKT = 1 */
#define CLOCK_ERROR_HANDLING        1
#define REF_TIMER_USED              CTIMER0
#define REF_TIMER_CLOCK_FREQUENCY   96e06
#define SYSTICK_RELOAD_VALUE        150000
#define ISR_FREQUENCY               1000 /* Hz */
#define CLOCK_TEST_TOLERANCE        20   /* % */
```

4.8 Digital I/O test

4.8.1 Digital I/O test description

In LPC553x safety library example project, GPIO P1_4 and P1_17 on LPC5536-EVK are selected for the safety digital I/O test, these two pins are connected to J10 header on LPC553x EVK board.
The digital I/O test routines are divided into two main processes: pre-run one time safety test and runtime periodical safety test

- Pre-run one time safety test
  – SafetyDigitalOutputTest
  – SafetyDigitalInputOutput_ShortSupplyTest
  – SafetyDigitalInputOutput_ShortAdjTest
- Runtime periodical safety test
  – SafetyDigitalOutputTest
  – SafetyDigitalInputOutput_ShortSupplyTest

4.8.2 Digital I/O test configuration

The configuration of the digital I/O test is in safety_test_items.c.

```
fs_dio_test_lpc_t dio_safety_test_item_0 =                /* P1_4 */
    {.iocon_mode_shift = IOCON_PIO_MODE_SHIFT,            /*Device depend*/
     .pPort_byte       = (uint8_t *)&(GPIO->B[1][4]),     /*adress of byte register in GPIO*/
     .pPort_dir        = (uint32_t *)&(GPIO->DIR[1]),     /* asress of dir1 register*/
     .pPort_Iocon      = (uint32_t *)&(IOCON->PIO[1][4]), /* Adress of concrete IOCON register*/
     .pinNum           = 4,                               /*Position in DIR registor*/
     .gpio_clkc_shift  = SYSCON_AHBCLKCTRL0_GPIO1_SHIFT};

fs_dio_test_lpc_t dio_safety_test_item_1 =                /* P1_17 */
    {.iocon_mode_shift = IOCON_PIO_MODE_SHIFT,            /* Device depend */
     .pPort_byte       = (uint8_t *)&(GPIO->B[1][17]),     /*adress of byte register in GPIO*/
     .pPort_dir        = (uint32_t *)&(GPIO->DIR[1]),     /* asress of dir1 register*/
     .pPort_Iocon      = (uint32_t *)&(IOCON->PIO[1][17]), /* Adress of concrete IOCON register*/

     .pinNum          = 17, /*Position in DIR registor*/
     .gpio_clkc_shift = SYSCON_AHBCLKCTRL0_GPIO1_SHIFT};

/* NULL terminated array of pointers to dio_test_t items for safety DIO test */
fs_dio_test_lpc_t *dio_safety_test_items[] = {&dio_safety_test_item_0, &dio_safety_test_item_1, NULL};
```

The execution of the digital I/O tests must be adapted to the final application. Be careful with the hardware connections and design. You can change the GPIO for the safety
digital I/O test by configuring dio_safety_test_items[] in safety_test_items.c. In most cases, the tested (and sometimes also auxiliary) pin must be reconfigured during the application run. It is recommended to use the unused pins for the digital I/O test.

4.9 Analog I/O test

4.9.1 Analog I/O test description

In LPC553x safety library example project, P0_16/ADC0IN3B, P0_31/ADC0IN8A, and P0_15/ADC0IN3A on LPC5536-EVK are selected for the safety analog I/O test, because the ADC module on MCU LPC5536 does not allow to connect the VREFH, VREFL internally to the ADC input. It is necessary for the user to connect these signals (for the analog I/O test) with flying wires as shown below.

- GND connected to P0_16/ADC0IN3B (J9-5) for ADC VREFL Test
- 3.3 V connected to P0_31/ADC0IN8A (J9-31) for ADC VREFH Test
- 1.65 V connected to P0_15/ADC0IN3A (J9-1) for ADC Bandgap Test

The analog I/O test routines are divided into two main processes:

- Pre-run one time safety test
  – SafetyAnalogTest
- Runtime periodical safety test
  – SafetyAnalogTest

4.9.2 Analog I/O test configuration

The execution of the analog I/O tests must be adapted to the final application. Be careful with the hardware connections and design. You can change the ADC channels for the safety analog I/O test by configuring FS_CFG_AIO_CHANNELS_INIT and
FS_CFG_AIO_CHANNELS_SIDE_INIT in safety_config.h.

- FS_CFG_AIO_CHANNELS_INIT indicates ADC channel number.
- FS_CFG_AIO_CHANNELS_SIDE_INIT indicates ADC channel side.

```
#define TESTED_ADC      ADC0 //  ADC0 /*which ADC is use for AIO test*/

/* 0V, 3.3V,  1.65V */
#define FS_CFG_AIO_CHANNELS_INIT \
    {                            \
        3, 8, 3                 \
    } /* ADC trigger ID for respective channel (set in Adc0Setup()) */

#define FS_CFG_AIO_CHANNELS_SIDE_INIT {1, 0, 0}  /* sides associated with input channels,  0 = A, 1 = B */
```

As shown in the above figure:

- First element corresponds to ADC VREFL test
- Second element corresponds to ADC VREFH test
- Third element corresponds to ADC Bandgap test

For example, “3” in FS_CFG_AIO_CHANNELS_INIT and “1” in
FS_CFG_AIO_CHANNELS_SIDE_INIT indicates that ADC0 channel 3 side B is selected for ADC VREFL test.

4.10 Watchdog test

4.10.1 Watchdog test description

The watchdog test is not directly specified in the IEC60730 – annex H table, however, it partially fulfills the safety requirements according to IEC 60730-1, IEC 60335, UL 60730, and UL 1998 standards.
The watchdog test provides the testing of the watchdog timer functionality. The test is run only once after the reset. The test causes the WDOG reset and compares the preset time for the WDOG reset to the real time.

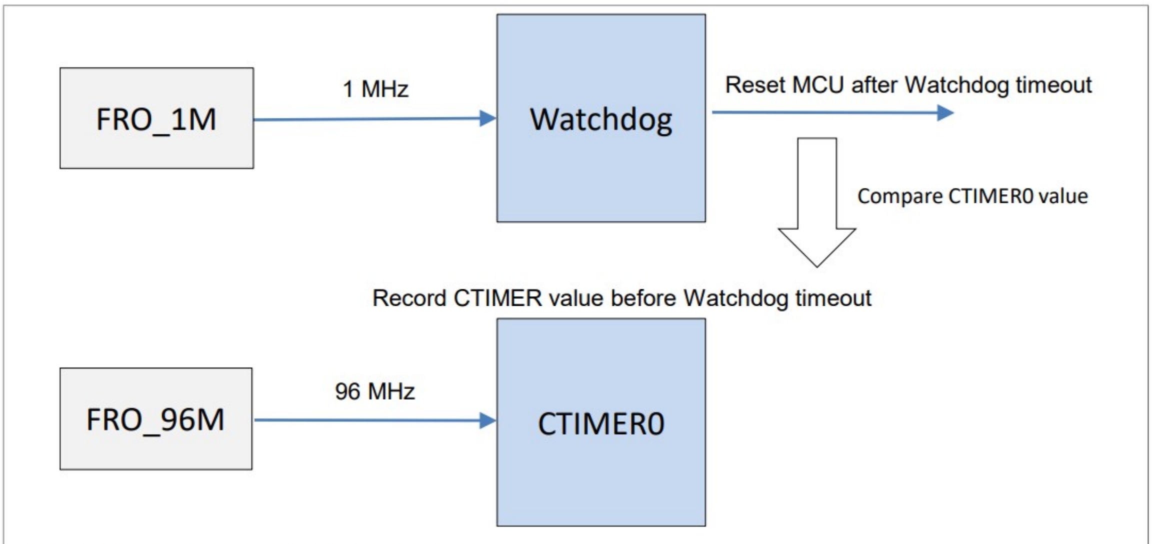


Figure 12. Example of watchdog test

In LPC553x safety library example project, the watchdog is tested using the following steps:

1. After reset, enable watchdog and stop refreshing on purpose to trigger watchdog reset MCU.
2. Enable CTIMER0 to measure how long it takes for the watchdog timeout and reset.
3. After watchdog reset, confirm that this reset is caused by watchdog by checking PMC->AOREG1 register.
4. Read CTIMER0 to get the exact time of watchdog timeout and reset.

Revision history

The table below summarizes the revisions to this document.

Table 3. Revision history

Revision number	Date	Substantive changes
0	4-Jan-23	Initial public release

Legal information

6.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including – without limitation lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice.

This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks

associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at **http://www.nxp.com/profile/terms**, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at **PSIRT@nxp.com**) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

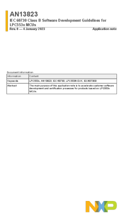
6.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective

owners.
NXP — wordmark and logo are trademarks of NXP B.V.
AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μVision, Versatile — are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.
Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2023 NXP B.V.
For more information, please visit: http://www.nxp.com
All rights reserved.
Date of release: 4 January 2023
Document identifier: AN13823

Documents / Resources

	**NXP AN13823 IEC 60730 Class B Software for LPC553x MCUs** [pdf] User Guide AN13823 IEC 60730 Class B Software for LPC553x MCUs, AN13823, IEC 60730 Class B Software for LPC553x MCUs, AN13823 IEC 60730 Class B Software

References

- NXP® Semiconductors Official Site | NXP Semiconductors
- NXP® Semiconductors Official Site | NXP Semiconductors
- Our Terms And Conditions Of Commercial Sale | NXP Semiconductors

Manuals+,