

**Core self-test library compliant with IEC 60730, IEC 60335 UL 60730,
UL 1998 documentation**

CPU Registers Test

Document revision history

Date	Author	Version	Notes
10/2015	Jozef Sedlak	0.1	Initial release
11/2015	Jozef Sedlak	1.0	Version for certification
10/2016	Jozef Sedlak	1.1	NXP
11/2018	Jozef Sedlak	3.0	Release with new compilers support

1	CPU REGISTER TEST ARCHITECTURE.....	4
2	CPU REGISTER TEST IN COMPLIANCE WITH IEC/UL STANDARDS.....	10
3	CPU REGISTER TEST IMPLEMENTATION.....	11
3.1	IEC60730B_CM4_CM7_CPU_REGISTERTEST().....	12
3.2	IEC60730B_CM4_CM7_CPU_NONSTACKEDREGISTERTEST().....	13
3.3	IEC60730B_CM4_CM7_CPU_PRIMASKTEST().....	14
3.4	IEC60730B_CM4_CM7_CPU_SPMaintTest().....	15
3.5	IEC60730B_CM4_CM7_CPU_SPPROCESSTEST().....	16
3.6	IEC60730B_CM4_CM7_CPU_SPECIALTEST().....	17
3.7	IEC60730B_CM4_CM7_CPU_SPECIALTEST_8PRIORITYLEVELS().....	18
3.8	IEC60730B_CM4_CM7_CPU_CONTROLTEST().....	19
3.9	IEC60730B_CM4_CM7_CPU_CONTROLTEST_FPU().....	20
3.10	IEC60730B_CM4_CM7_CPU_FLOATTEST1().....	21
3.11	IEC60730B_CM4_CM7_CPU_FLOATTEST2().....	22
4	CPU REGISTERS TEST MODULE TEST CONCEPT.....	23
5	CPU REGISTERS TEST VALIDATION.....	23

1 CPU Register Test Architecture

The CPU registers tests procedure tests all of the CM4/CM7 CPU registers, except the Program Counter register for stuck at condition. The Program counter test is implemented as a standalone safety routine. There is a set of tests performed once after microcontroller reset and also during runtime.

CPU registers test functions cover tests of the following registers:

General purpose registers:

- R0 – R12

Stack pointer registers:

- SP_main
- SP_process

Link register:

- LR

Special registers:

- PRIMASK
- FAULTMASK
- BASEPRI
- CONTROL
- APSR

FPU registers:

- FPSCR
- S0 – S31

The identification of safety errors is ensured by the specific FAIL return in case of some register being stuck at fault. The application developer must assess the return value of every function, and if it is equal to FAIL return, then the jump into the safety error handling function should occur. The safety error handling function may be specific according to the application and is not a part of our library. The main purpose of this function is to put the application into a safe state.

In some special cases, error is not reported by fail return, as it would require the action of a corrupted register. In that case, function waits in endless loop for reset.

The principle of stuck at error test of the CPU registers is to write and compare two test patterns into every register. The content of the register is compared with the constant, or with the value written into another register which has been tested before. Most of the time, R0, R1, and R2 are used as auxiliary registers. Patterns are defined to check logical one and logical zero values in all of the registers bits. For tests of PRIMASK, FAULTMASK, BASEPRI and CONTROL, the original contents need to be spared. For tests of SP_main and SP_process, also CONTROL register content needs to be spared. In case of FPU registers test, the content of FPSCR is spared. The system register CPACR contains one bit for enabling FPU. Its content is spared as well. The block diagrams for the respective registers are shown in the following figures:

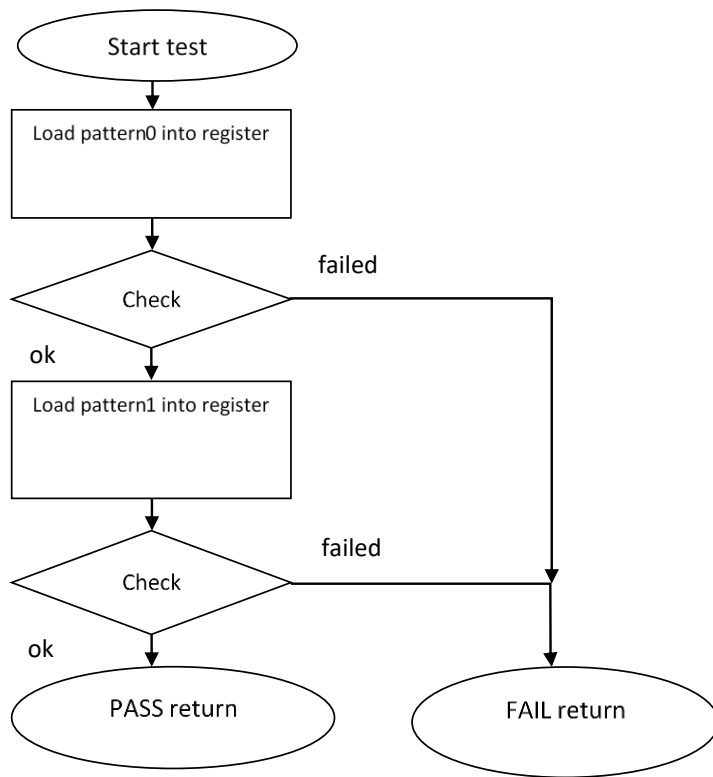


Figure 1. Block diagram for R2 – R12 registers test

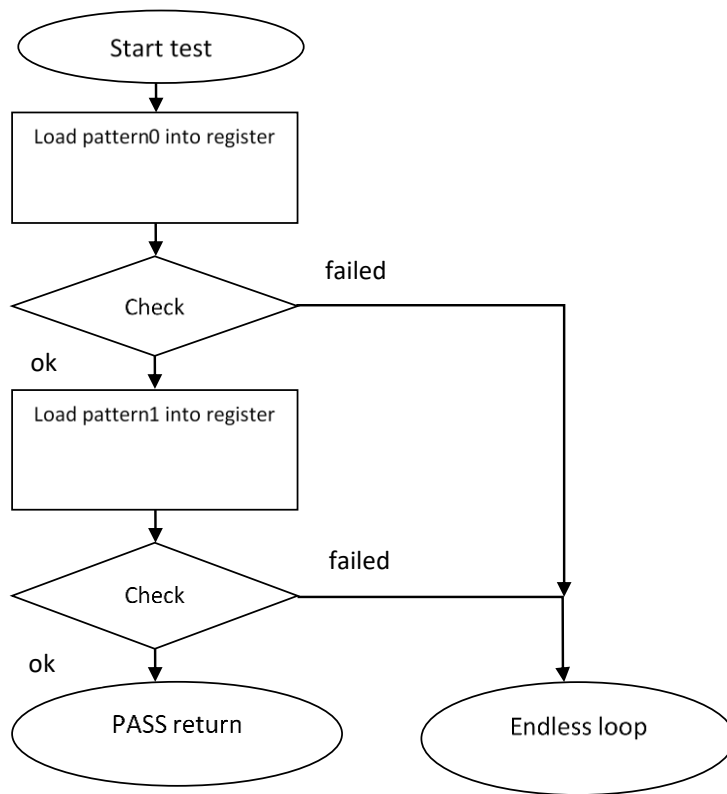


Figure 2. Block diagram for R0, R1, LR, APSR registers test

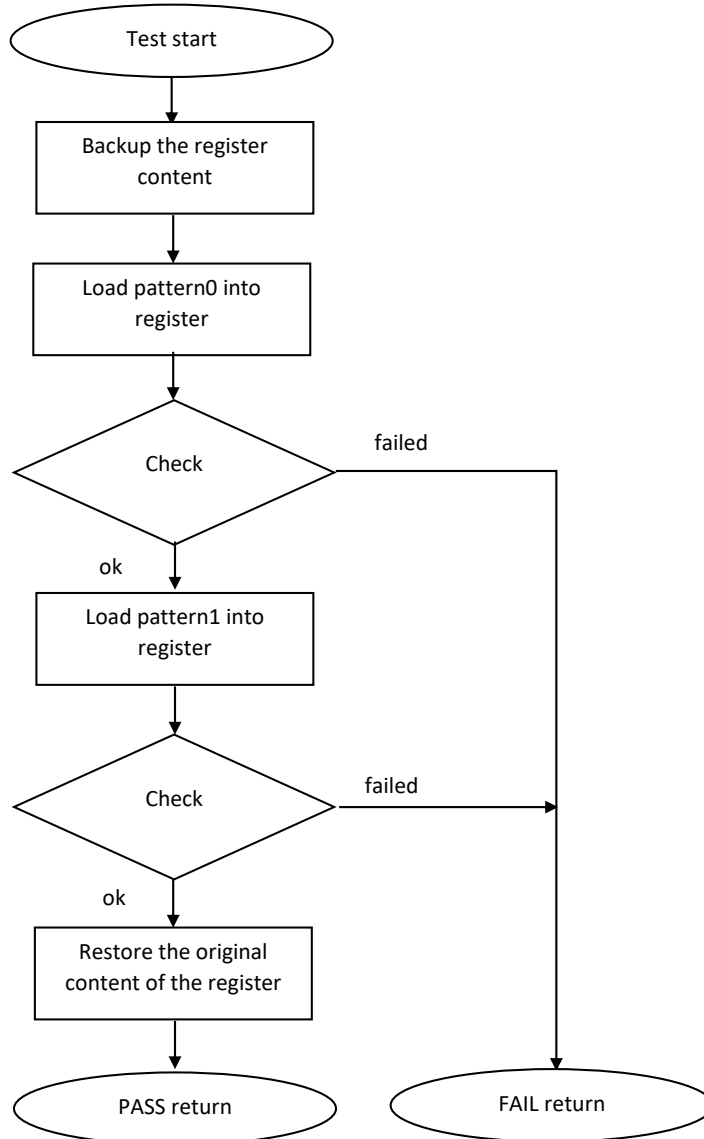


Figure 3. Block diagram for PRIMASK, FAULTMASK, BASEPRI and CONTROL registers test

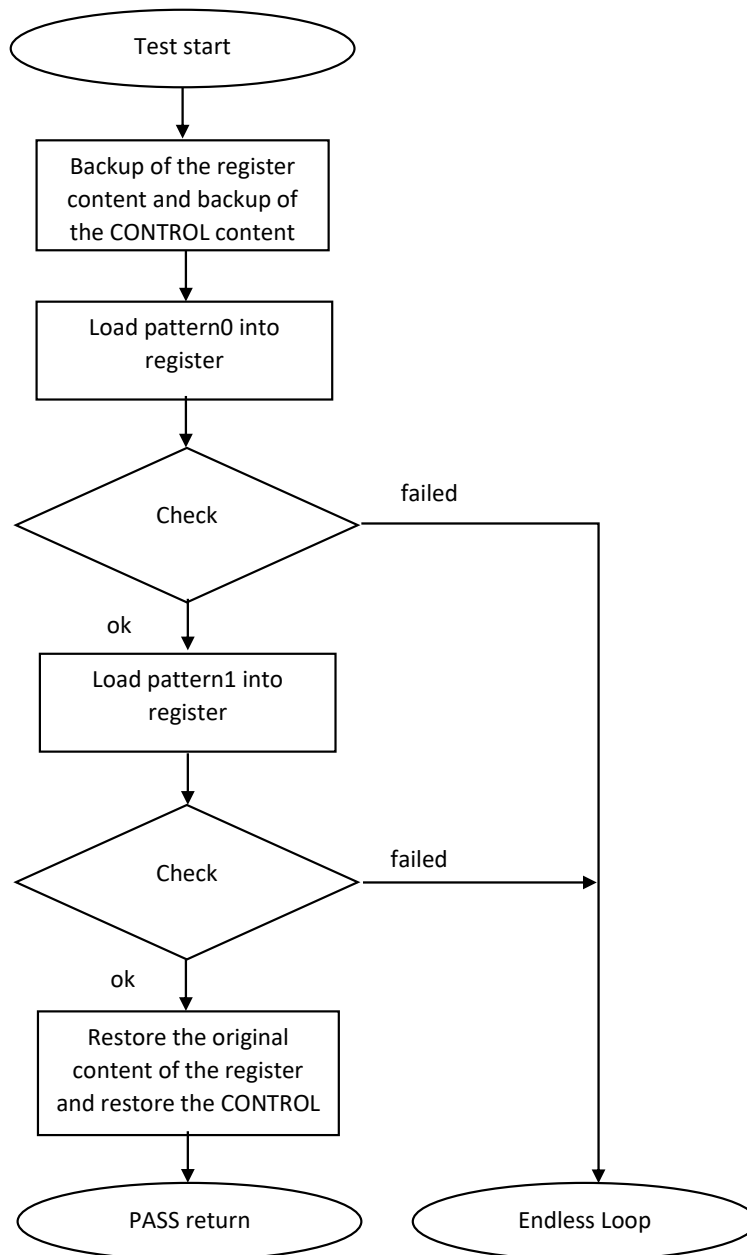


Figure 4. Block diagram for SP_main and SP_process registers test

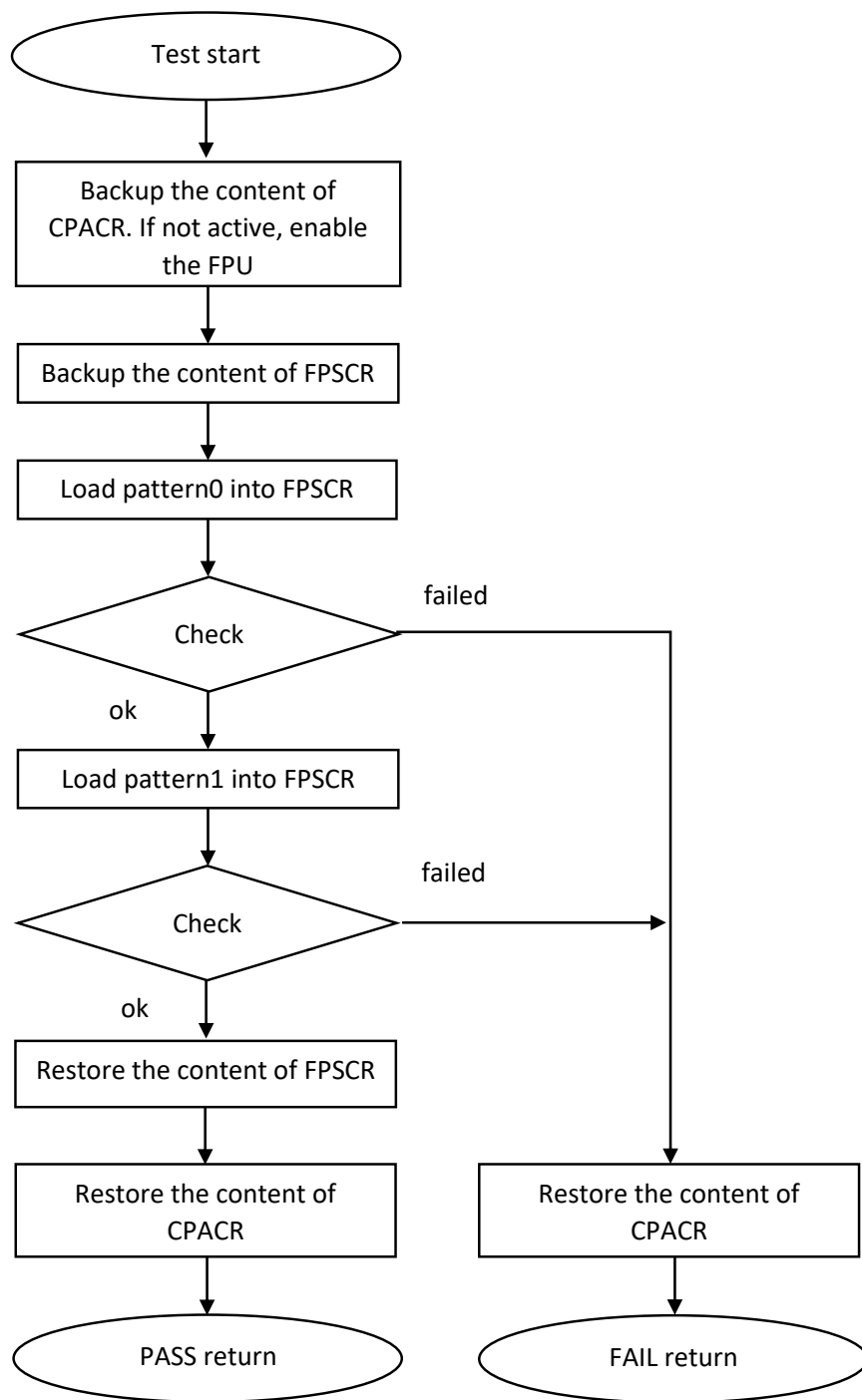


Figure 5. Block diagram for FPSCR register test

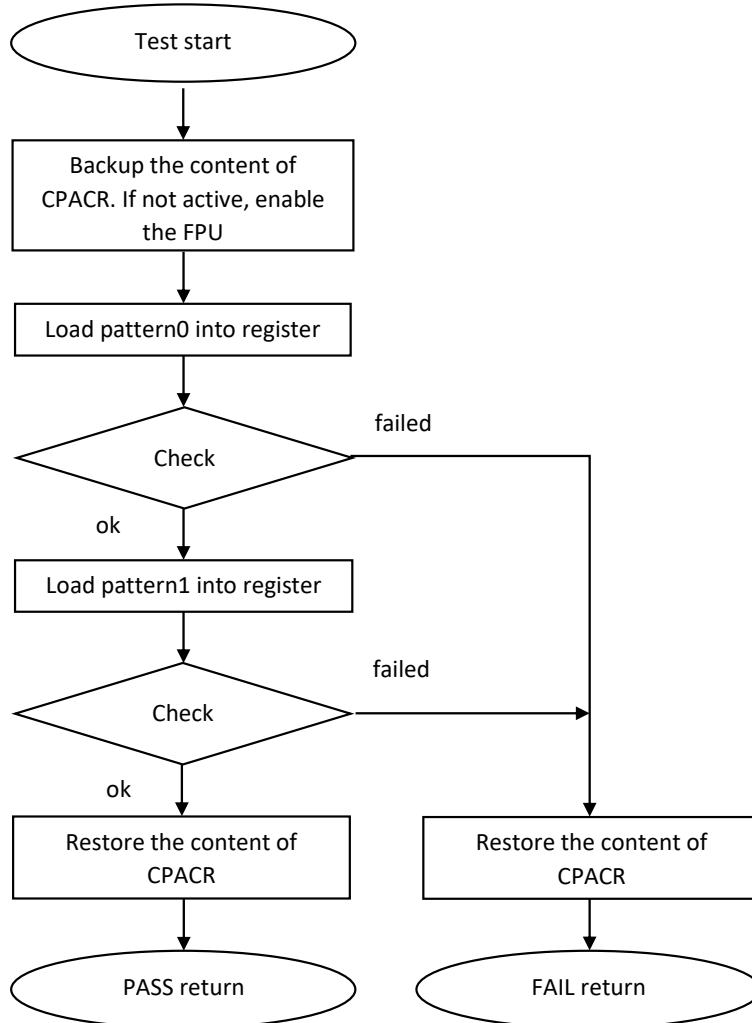


Figure 6. Block diagram for S0 – S31 registers test

2 CPU Register Test in compliance with IEC/UL standards

The performed overload test fulfils safety requirements according to IEC 60730-1, IEC 60335, UL 60730, and UL 1998 standards as described in the following table:

Table 1. CPU Registers Test in Compliance with IEC and UL Standards

Test	Component	Fault / error	SW / HW class	Acceptable measures
CPU registers test	CPU (1.1 – Registers)	Stuck at	B/R.1	Periodic self-test

3 CPU Register Test Implementation

The test functions for the CPU registers are placed in IEC60730_B_CM4_CM7_reg.S and are written as assembler functions. The header file with return values and function prototypes is

IEC60730_B_CM4_CM7_reg.h.

IEC60730_B_CM4_CM7.h, and asm_mac_common.h are files that are included in

IEC60730_B_CM4_CM7_reg.S and therefore need to also be placed in the application. For devices containing the FPU, IEC60730_B_CM4_CM7_reg_fpu.S is an additional file with tests of FPU-related registers.

The following functions are called to test the corresponding registers:

- IEC60730B_CM4_CM7_CPU_RegisterTest ()
- IEC60730B_CM4_CM7_CPU_NonStackedRegisterTest ()
- IEC60730B_CM4_CM7_CPU_PrimaskTest ()
- IEC60730B_CM4_CM7_CPU_SPmainTest ()
- IEC60730B_CM4_CM7_CPU_SPprocessTest ()
- IEC60730B_CM4_CM7_CPU_ControlTest ()
- IEC60730B_CM4_CM7_CPU_SpecialTest()

When device has FPU (the following functions are placed in IEC60730_B_CM4_CM7_reg_fpu.S):

- IEC60730B_CM4_CM7_CPU_ControlTest_fpu ()
- IEC60730B_CM4_CM7_CPU_FloatTest1 ()
- IEC60730B_CM4_CM7_CPU_FloatTest2 ()

Error detection is recognized by the specific return value as described in the following chapters. There are several exceptions. If some of the following registers are corrupted (R0, R1, LR, APSR, and SP), an application will be in an endless loop instead of returning an error value. This is because if some of these registers are corrupted, the application is not able to make standard operations to identify the safety error (to compare something, to move out from the function, or to return a value).

The use of functions is the same after reset and during runtime. The developer must be aware in case of functions use during runtime as described in the following chapters.

The following is an example of a function calling:

```
#include "IEC60730_B_CM4_CM7.h"
if(IEC60730B_ST_CPU_REGISTER_FAIL== IEC60730B_CM4_CM7_CPU_RegisterTest())
    SafetyError();
```

3.1 IEC60730B_CM4_CM7_CPU_RegisterTest()

This function checks R0-R7, R12, LR and APSR in a sequence. Each register is tested according to the block diagrams in above or above.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_RegisterTest(void)
```

Test patterns for respective registers:

R0 – R7, R12, LR: 0x55555555, 0xAAAAAAAA

APSR: 0x50050000, 0xA80A0000

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT;

Function can have two values:

IEC60730B_ST_CPU_REGISTER_FAIL (0x00000101)

IEC60730B_ST_CPU_PASS (0)

In case that R0, R1, LR, or APSR are corrupted, the function will be in an endless loop with interrupts disabled. This state needs to be observed by another safety mechanism (for instance Watchdog).

Function performance:

Function duration is approximately 166 cycles (2.08 μ s) ¹

Function size is 184 bytes.²

Calling restrictions:

None

3.2 IEC60730B_CM4_CM7_CPU_NonStackedRegisterTest()

This function checks R8 – R11 in a sequence. Each register is tested according to the block diagram in above.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_NonStackedRegisterTest(void);
```

Test patterns for respective registers:

R8 – R11: 0x55555555, 0xAAAAAAAA

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT;

Can have two values:

IEC60730B_ST_CPU_NONSTACKED_REGISTER_FAIL (0x00000102)

IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 60 cycles (0.75 μ s) ¹

Function size is 68 bytes.²

Calling restrictions:

None

3.3 IEC60730B_CM4_CM7_CPU_PrimaskTest()

This function checks PRIMASK according to the block diagram in above.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_PrimaskTest(void);
```

Test pattern:

PRIMASK: 0x00000001, 0x00000000

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT

Function can have two values:

- IEC60730B_ST_CPU_PRIMASK_FAIL (0x00000103)
- IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 29 cycles (0.37 μ s) ¹

Function size is 44 bytes.²

Calling restrictions:

This function cannot be interrupted by an interrupt where the global interrupts will be disabled.

3.4 IEC60730B_CM4_CM7_CPU_SPmainTest()

This function checks SP_main according to the block diagram in Figure 4.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_SPmainTest(void);
```

Test pattern:

SP_main: 0x55555554, 0xAAAAAAAA8

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT;

When passes:

IEC60730B_PASS(0)

In case that SP_main is corrupted, the function will be in an endless loop with interrupts disabled. This state needs to be observed by another safety mechanism (for instance Watchdog).

Function performance:

Function duration is approximately 49 cycles (0.61 μ s) ¹

Function size is 58 bytes.²

Calling restrictions:

This function cannot be interrupted.

3.5 IEC60730B_CM4_CM7_CPU_SPprocessTest()

This function checks SP_process according to the block diagram in Figure 4.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_SPprocessTest(void);
```

Test pattern:

SP_process: 0x55555554, 0xAAAAAA8

Function inputs:

Void. No inputs are passed to the function.

Function output:

typedef unsigned long IEC60730B_RESULT;

When function passes:

IEC60730B_PASS(0)

In case that the SP_process is corrupted, the function will be in an endless loop with interrupts disabled. This state needs to be observed by another safety mechanism (for instance the Watchdog).

Function performance:

Function duration is approximately 40 cycles (0.5 μ s) ¹

Function size is 58 bytes.²

Calling restrictions:

This function cannot be interrupted.

3.6 IEC60730B_CM4_CM7_CPU_SpecialTest()

This function checks FAULTMASK and BASEPRI registers according to the block diagram in Figure 3.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_SpecialTest(void);
```

Test pattern:

FAULTMASK: 0x00000001, 0x00000000

BASEPRI: 0x000000A0, 0x00000050

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT

Function can have two values:

- IEC60730B_ST_CPU_SPECIAL_FAIL (0x00000107)
- IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 54 cycles (0.68 μ s) ¹

Function size is 84 bytes. ²

Calling restrictions:

None

3.7 IEC60730B_CM4_CM7_CPU_SpecialTest_8PriorityLevels ()

This function checks FAULTMASK and BASEPRI registers for devices with 8 interrupt priority levels according to the block diagram in Figure 3.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_SpecialTest_8PriorityLevels(void);
```

Test pattern:

FAULTMASK: 0x00000001, 0x00000000

BASEPRI: 0x000000A0, 0x00000040

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT

Function can have two values:

- IEC60730B_ST_CPU_SPECIAL_FAIL (0x00000107)
- IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 53 cycles (1.1 μ s) ¹

Function size is 84 bytes.²

Calling restrictions:

None

3.8 IEC60730B_CM4_CM7_CPU_ControlTest()

This function checks CONTROL according to the block diagram in Figure 3.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_ControlTest(void);
```

Test pattern:

CONTROL: 0x00000002, 0x00000000

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT

Function can have two values:

- IEC60730B_ST_CPU_CONTROL_FAIL (0x00000106)
- IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 30 cycles (0.38 μ s)³

Function size is 48 bytes.²

Calling restrictions:

This function cannot be interrupted.

3.9 IEC60730B_CM4_CM7_CPU_ControlTest_fpu()

This function checks CONTROL according to the block diagram in Figure 3.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_ControlTest_fpu(void);
```

Test pattern:

CONTROL: 0x00000000, 0x00000002, 0x00000004

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT

Function can have two values:

- IEC60730B_ST_CPU_CONTROL_FAIL (0x00000106)
- IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 50 cycles (0.625 μ s) ¹

Function size is 62 bytes.²

Calling restrictions:

This function cannot be interrupted. Should be used for devices with FPU, as a replace of the IEC60730B_CM4_CM7_CPU_ControlTest() function.

3.10 IEC60730B_CM4_CM7_CPU_FloatTest1()

The function checks FPSCR and S0 – S15 registers according to block diagrams in Figure 5 and Figure 6. Within the function, FPU is enabled in the CPACR register. At the end of the function, the original content of CPACR is adjusted.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_FloatTest1(void);
```

Test pattern:

FPSCR: 0x55400015, 0xA280008A
S0 – S15: 0x55555555, 0xAAAAAAAA

Function inputs:

Void. No inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT

Function can have two values:

- IEC60730B_ST_CPU_FLOAT_1_FAIL (0x00000108)
- IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 285 cycles (3.56 μ s) ¹

Function size is 472 bytes.²

Calling restrictions:

Can be used only for devices with FPU.

3.11 IEC60730B_CM4_CM7_CPU_FloatTest2()

The function checks S16 – S31 registers according to the block diagram in Figure 6. Within the function, FPU is enabled in CPACR register. At the end of the function, original content of CPACR is adjusted.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CPU_FloatTest2(void);
```

Test pattern:

S16 – S31: 0x55555555, 0xAAAAAAAA

Function inputs:

Void, no inputs are passed into the function.

Function output:

typedef unsigned long IEC60730B_RESULT

Function can have two values:

- IEC60730B_ST_CPU_FLOAT_2_FAIL (0x00000109)
- IEC60730B_ST_CPU_PASS (0)

Function performance:

Function duration is approximately 280 cycles (3.5 μ s) ¹

Function size is 446 bytes.²

Calling restrictions:

Can be used only for devices with FPU.

-
- 1- The number of cycles and execution time were measured by MKV31 / 80 MHz CPU clock / 20 MHz flash clock.
 - 2- Function compiled by IAR v8.22.2
 - 3- The number of cycles and execution time were measured by MK32W / 48 MHz CPU clock / 24 MHz flash clock.

4 CPU Registers Test Module test concept

Content moved to TestReport_IEC60730B_Register_rev3_0 document

5 CPU Registers Test Validation

Content moved to TestReport_IEC60730B_Register_rev3_0 document

Table V. Validation

Date	Validated by	Validated Revision of Document	Validated Version of Source Code	Validation Result
11/2015	Jaroslav Lepka	1.0	1.0	P
11/2016	Pavel Sustek	1.1	1.0	P
11/2018	Jaroslav Lepka	3.0	3.0	P

Validation result options:

P – Passed

F – Failed

N/A – Not applicable

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all **liability, including without limitation consequential or incidental damages.** "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating **parameters, including "typicals," must be validated for each customer application by customer's technical experts.** NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:
nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, and the Freescale logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners.

ARM, the ARM logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2016 NXP B.V.

