

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

Stack Test

Document revision history

Date	Author	Version	Notes
11/2015	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	STACK TEST ARCHITECTURE	4
2	STACK TEST IN COMPLIANCE WITH IEC/UL STANDARDS	4
3	STACK TEST IMPLEMENTATION	4
3.1	LINKER SETUP	5
3.2	TEST INITIALIZATION	6
3.3	STACK TEST	7
4	STACK TEST MODULE TEST CONCEPT	8
5	STACK TEST VALIDATION	8

1 Stack Test Architecture

This test routine is used for testing overflow and underflow conditions of the application stack. Testing of stuck-at faults in the memory area occupied by the stack is covered by the Variable Memory Test. Overflow or underflow of the stack can occur if the stack is incorrectly controlled, or by defining too low a stack area for the given application.

The principle of the test is to fill an area below and above the stack with a known pattern. These areas need to be defined in the linker configuration file, as the stack is also. The initialization function then fills these areas with a pattern defined by the application developer. The pattern must have a value that does not appear elsewhere in the application. The test is performed after reset and during the application runtime in the same way. The purpose is to check if the exact pattern is still written in these areas. If it is not, it is a sign that there is incorrect stack behavior. If this occurs then the fail return value from the test function must be processed as a safety error.

2 Stack Test in compliance with IEC/UL standards

Stack test is an additional test, not directly specified in IEC60730 annex H table.

3 Stack Test Implementation

The test function for the stack and the initialization function are placed in the *IEC60730_B_CM4_CM7_stack.S* and are written as assembler functions. The header file with the return values and the function prototypes is *IEC60730_B_CM4_CM7_stack.h*.

IEC60730_B_CM4_CM7.h and *asm_mac_common.h* are files that are included in *IEC60730_B_CM4_CM7_stack.S* and therefore need to also be placed in the application. In the following chapters, the example for linker setup, the process of initialization and implementation is shown.

3.1 Linker setup

The size and placing of the application stack is generally defined in the linker configuration file. Therefore, the user must define areas below and under the stack here as well. There are other methods to achieve this, only one example is shown here. The size of the areas must be a multiple of 0x4. The minimal size is 0x4.

```

define symbol __ICFEDIT_region_RAM_start__      = 0x1FFFC410;
define symbol __ICFEDIT_region_RAM_end__        = 0x1FFFFFF7;
define symbol __region_RAM2_start__             = 0x20000014;
define symbol __region_RAM2_end__               = 0x20003FDB;
define symbol __ICFEDIT_size_cstack__ = 512;
define exported symbol STACK_TEST_BLOCK_SIZE = 0x10;
define exported symbol STACK_TEST_P_4          = __region_RAM2_end__ - 0x3;
define exported symbol STACK_TEST_P_3          = STACK_TEST_P_4 - STACK_TEST_BLOCK_SIZE + 0x4;
define exported symbol __BOOT_STACK_ADDRESS = STACK_TEST_P_3 - 0x4;
define exported symbol STACK_TEST_P_2          = __BOOT_STACK_ADDRESS - __ICFEDIT_size_cstack__ -
0x4;
define exported symbol STACK_TEST_P_1          = STACK_TEST_P_2 - STACK_TEST_BLOCK_SIZE;
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to __region_RAM2_end__] -
mem:[from STACK_TEST_P_1 size 0x10] - mem:[from STACK_TEST_P_3 size 0x10];

//
// | _____ | --> STACK_TEST_P_1  ....ADR
// | _____ |                      ....ADR + 0x4
// | _____ |                      ....ADR + 0x8
// | _____ | --> STACK_TEST_P_2  ....ADR + 0xC
// | _____ |
// | _____ |
// | _____ |
// |  STACK  |
// | _____ |
// | _____ |
// | _____ |
// | _____ |
// | _____ | --> __BOOT_STACK_ADDRESS
// | _____ | --> STACK_TEST_P_3
// | _____ |
// | _____ |
// | _____ | --> STACK_TEST_P_4

```

In the example, the size is set to 0x10. The symbols STACK_TEST_P_2 and STACK_TEST_P_3 define the first addresses under and above the stack and are defined as exported symbols. This means that they are also visible in the application. Areas are not included in the RAM region, so the compiler cannot reserve this place for some variables or other parameters.

3.2 Test initialization

The purpose of the initialization is to fill defined areas with a given pattern. Firstly you must set values from the linker configuration file into variables. Then define the rest of the parameters needed for the initialization function.

Example of initialization:

```
#include "IEC60730_B_CM4_CM7.h"
```

```
extern unsigned long STACK_TEST_P_2;
extern unsigned long STACK_TEST_P_3;
const unsigned long stack_test_first_address = (unsigned long) &STACK_TEST_P_2;
const unsigned long stack_test_second_address = (unsigned long) &STACK_TEST_P_3;
const unsigned long stack_test_pattern = 0x77777777;
const unsigned long stack_test_block_size = 0x10;
```

Function prototype:

```
void IEC60730B_CM4_CM7_Stack_Init(unsigned long stack_test_pattern, \
unsigned long first_address, unsigned long second_address, unsigned long block_size);
```

Function inputs:

stack_test_pattern - pattern to be written into the areas
first_address - the first address under the stack area
second_address - the first address above the stack area
block_size - size of areas under and above the stack.

Function output:

Void. The function does not return any value.

Function performance:

The function duration for block size 0x10 is approximately 72 cycles (0.9 μ s)¹
The function size is 26 bytes.²

Calling restrictions:

None.

3.3 Stack test

The testing procedure is the same after reset and during runtime. The function checks if the areas are not rewritten with different content, as the defined pattern is. Inputs for testing functions must be the same as for the initialization function.

Function prototype:

IEC60730B_RESULT IEC60730B_CM4_CM7_Stack_Test (unsigned long stack_test_pattern, unsigned long first_address, unsigned long second_address, unsigned long block_size);

Function inputs:

stack_test_pattern - the pattern to be written into areas
first_address - the first address under the stack area
second_address - the first address above the stack area
block_size - size of areas under and above the stack

Function output:

typedef unsigned long IEC60730B_RESULT;

This can have two values:

IEC60730B_ST_STACK_FAIL (0x00000501)

IEC60730B_ST_STACK_PASS (0)

Function performance:

The function duration for block size 0x10 is approximately 100 cycles (1.25 μ s)¹

The function size is 38 bytes.²

Calling restrictions:

None.

1 – The number of cycles and the execution time were measured with the use of MKV31 / 80 MHz CPU clock / 20 MHz flash clock. The execution time depends on the defined size of the areas.

2- Function compiled by IAR v8.22.2

4 Stack Test Module test concept

Content moved to TestReport_IEC60730B_Stack_rev3_0 document

5 Stack Test Validation

Content moved to TestReport_IEC60730B_Stack_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	Jozef Sedlak	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.

