

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

Clock Test

Document revision history

Date	Author	Version	Notes
11/2015	Jozef Sedlak	0.1	Initial version
11/2015	Jozef Sedlak	1.0	Version for certification
11/2016	Jozef Sedlak	1.1	NXP
11/2018	Jozef Sedlak	3.0	Release with new compilers support. MIMXRT added. 2 nd test version removed.

1	CLOCK TEST ARCHITECTURE.....	4
2	CLOCK TEST IN COMPLIANCE WITH IEC/UL STANDARDS.....	5
3	CLOCK TEST IMPLEMENTATION	5
3.1	IEC60730B_CM4_CM7_CLK_SYNC_INIT()	6
3.2	IEC60730B_CM4_CM7_CLK_SYNC_LPTMR_ISR()	7
3.3	IEC60730B_CM4_CM7_CLK_SYNC_GPT_ISR()	7
3.4	IEC60730B_CM4_CM7_CLK_CHECK()	8
4	CLOCK TEST MODULE TEST CONCEPT	9
5	CLOCK TEST VALIDATION	9

1 Clock Test Architecture

The Clock test procedure tests the clock frequency of the processor for wrong frequency fault. The clock test is performed during application runtime.

The identification of a safety error is ensured by the specific FAIL return in case of clock fault. The application developer needs to assess the return value of the test function, and if it is equal to the 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 safety state.

The clock test principle is based on comparison of two independent clock sources. If the test routine detects a change in the frequency ratio between the clocks sources, a fail error code is returned. The test routine uses one timer and one periodical event in the application. The periodical event could be also an interrupt from different timer as the one that is already involved. Low Power Timer (LPTMR) or General Purpose Timer (GPT) are used.

The block diagram of test is shown in the following figure:

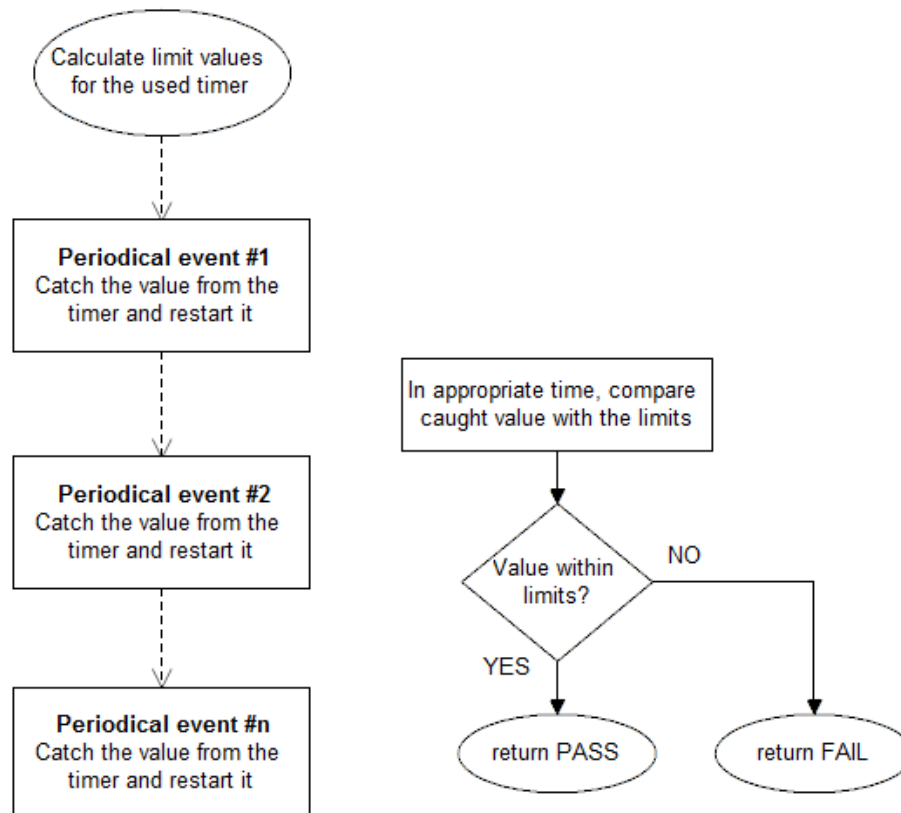


Fig. 1. Block diagram for Clock test

2 Clock 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. Clock Test in Compliance with the IEC and UL Standards

Test	Component	Fault/Error	SW / HW Class	Acceptable Measures
Clock test	3.Clock	Wrong Frequency	B / R.1	Frequency monitoring

3 Clock Test Implementation

Test functions for the Clock test are placed in IEC60730_B_CM4_CM7_clock.c and are written as c functions. The header file with the function prototypes is IEC60730_B_CM4_CM7_clock.h. The common library header file is IEC60730_B_CM4_CM7.h.

The following functions are called to test the Clock frequency:

- IEC60730B_CM4_CM7_CLK_SYNC_Init()
- IEC60730B_CM4_CM7_CLK_SYNC_LPTMR_Isr()
- IEC60730B_CM4_CM7_CLK_SYNC_GPT_Isr()
- IEC60730B_CM4_CM7_CLK_Check()

User must configure LPTMR/GPT , choose an appropriate periodical event and calculate limit values. The 32-bit global variable, for storing the content of the timer counter register must then be declared. Clock source of the chosen timer must differ from clock source of the periodical event. The IEC60730B_CM4_CM7_CLK_SYNC_Init() function is called once, normally before while() loop. The IEC60730B_CM4_CM7_CLK_SYNC_LPTMR_Isr() function is then called within periodic event. The function for evaluating - IEC60730B_CM4_CM7_CLK_Check() can be called at any chosen time. When test is in initialization phase, the check function returns "in progress" value. If the captured value from LPTMR/RTC is within the preset limits, the check function returns a pass value. If not, a defined fail value is returned.

The example of test implementation is as follows:

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

```
#define PERIODE_MSEC    (12.5)
```

```
#define DEVIATION_PERCENT (10)
```

```
#define TMR_CLK_HZ      (4e06)
```

```
#define LIMIT    ((PERIODE_MSEC) * (TMR_CLK_HZ/1000))
```

```
#define LIMIT_HI (((LIMIT) * (100 + DEVIATION_PERCENT)) / 100)
```

```
#define LIMIT_LO (((LIMIT) * (100 - DEVIATION_PERCENT)) / 100)
```

```
unsigned long clockTestContext;
```

```
unsigned long IEC60730B_clock_test_result ;
```

```

SysTick->VAL = 0;
SysTick->LOAD = 1000000; /* 12.5ms with 80MHz frequency */
SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk | SysTick_CTRL_ENABLE_Msk |
SysTick_CTRL_TICKINT_Msk;

SIM->SCGC5 |= SIM_SCGC5_LPTMR_MASK;
LPTMR0->CSR = 0;
LPTMR0->CSR = LPTMR_CSR_TCF_MASK | LPTMR_CSR_TFC_MASK;
LPTMR0->PSR |= LPTMR_PSR_PBYP_MASK;
LPTMR0->PSR &= (~LPTMR_PSR_PCS_MASK);
LPTMR0->CMR = 0;
LPTMR0->CSR |= LPTMR_CSR_TEN_MASK;
IEC60730B_CM4_CM7_CLK_SYNC_Init(&clockTestContext);

while(1)
{
    IEC60730B_clock_test_result = IEC60730B_CM4_CM7_CLK_Check(clockTestContext, LIMIT_LO,
LIMIT_HI);
}

void SYSTICK_Isr(void)
{
    IEC60730B_CM4_CM7_CLK_SYNC_LPTMR_Isr((unsigned long*)LPTMR0, &clockTestContext);
}

```

3.1 IEC60730B_CM4_CM7_CLK_SYNC_Init()

This function initializes one instance of the clock sync test. It sets the TestContext value to “in progress” state.

Function prototype:

```
void IEC60730B_CM4_CM7_CLK_SYNC_Init (unsigned long* pTestContext);
```

Function inputs:

pTestContext - Pointer to the context value of the clock test.

Function output:

Void. Function does not return any value.

Function performance:

The function duration is approximately 15 cycles (0.19 μ s)¹. Function size is 8 bytes.²

3.2 IEC60730B_CM4_CM7_CLK_SYNC_LPTMR_Isr()

This function is used only with the LPTMR module. The function reads the counter value from the timer and saves it into the TestContext variable. After that, the function starts the LPTMR again.

Function prototype:

```
void IEC60730B_CM4_CM7_CLK_SYNC_LPTMR_Isr(unsigned long* pSafetyTmr, unsigned long* pTestContext);
```

Function inputs:

pSafetyTmr - pointer to base address of the LPTMR module.
pTestContext - pointer to Context value of the clock test.

Function output:

Void. Function does not return any value.

Function performance:

Function duration is approximately 132 cycles (1.65 μ s)¹. Function size is 26 bytes.²

3.3 IEC60730B_CM4_CM7_CLK_SYNC_GPT_Isr()

This function is used only with the GPT module. This function reads the counter value from the timer and saves it into the TestContext variable. After that, it starts the GPT again.

Function prototype:

```
void IEC60730B_CM4_CM7_CLK_SYNC_GPT_Isr(unsigned long* pSafetyTmr, unsigned long* pTestContext);
```

Function inputs:

pSafetyTmr - pointer to base address of the GPT module.
pTestContext - pointer to Context value of the clock test.

Function output:

Void. Function does not return any value.

Function performance:

Function duration is approximately 52 cycles (2.17 μ s)³. Function size is 12 bytes.²

3.4 IEC60730B_CM4_CM7_CLK_Check()

This function handles the Clock test. It evaluates the caught value stored in TestContext variable with predefined limits. Until the first execution of the respective Isr function, the check function returns IEC60730B_ST_CLK_PROGRESS.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_CLK_Check(unsigned long testContext , unsigned long  
low_limit, unsigned long high_limit);
```

Function inputs:

pTestContext - Context value of the clock test. The type of the context variable is unsigned long.
low_limit - Low limit for the caught value from the timers counter.
high_limit - High limit for the caught value from the timers counter.

Function output:

This function returns the value of the unsigned long data type. It can have the following values:

- IEC60730B_ST_CLK_FAIL (0x00000601)
- IEC60730B_ST_CLK_PASS (0x00000000)
- IEC60730B_ST_CLK_PROGRESS (0x00000602)

Function performance:

The function duration is approximately 37 cycles (0.46 μ s)¹. Function size is 44 bytes.²

-
- 1- The number of cycles and the execution time were measured with the use of MKV31 / 80 MHz CPU clock / 20 MHz flash clock
 - 2- Function compiled by IAR v8.22.2
 - 3- The number of cycles and the execution time were measured with the use of MIMXRT1050/600MHz CPU clock

4 Clock Test Module test concept

Content moved to TestReport_IEC60730B_Clock_rev3_0 document

5 Clock Test Validation

Content moved to TestReport_IEC60730B_Clock_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.

