AN_1023
IEC60730-1 Class B

Abstract	IEC60730-1 is a safety standard for all home appliances sold in Europe. Nuvoton provides a sample code consists of low level software routines. These routines implements basic requirements specified in Annex H of the standard. User can add these codes into existing application to accelerate certification process. IEC60730-1 appendix H defines three classifications for automatic electronic controls. This sample code implements some test items required for Class B, which is intended to prevent unsafe operation of appliance. For example, thermal cut-offs and door locks for laundry equipment. For user interested in IEC60730-1 specification, please refer the document “Automatic electrical controls for household and similar use” published by International Electrotechnical Committee.
Apply to	NUC100 Series

Table of Contents-

1 GENERAL DESCRIPTION

IEC60730-1 is a safety standard for all home appliances sold in Europe. Nuvoton provides a sample code consists of low level software routines. These routines implements basic requirements specified in Annex H of the standard. User can add these codes into existing application to accelerate certification process.

IEC60730-1 appendix H defines three classifications for automatic electronic controls. This sample code implements some test items required for Class B, which is intended to prevent unsafe operation of appliance. For example, thermal cut-offs and door locks for laundry equipment. For user interested in IEC60730-1 specification, please refer the document “Automatic electrical controls for household and similar use” published by International Electrotechnical Committee.

2 TEST ITEMS SUMMARY

The sample code includes totally seven test items, CPU register test, program counter test, interrupt test, clock test, flash test and RAM test.

All the sources of the sample code is released. User can add them into existing project to cerate self-test program.

The following table lists the summary of test items.

	Test Items	Description
1	**1.1 CPU Register Test**	Test all CPU register.
2	**1.3 Program Counter Test**	Test Program Counter.
3	**2.0 Interrupt Test**	Interrupt count test using Timer and RTC.
4	**3.0 Clock Test**	Timer test using two timer channels.
5	**4.1 Flash Test**	Test a specific area of Flash and compare CRC.
6	**4.2 RAM Test - 1**	Test RAM using MarchX.
7	**4.2 RAM Test -2**	Test RAM using MarchC.

The following sections describe these test items.

3 GUIDE TO THE TEST ITEMS

3.1 CPU Registers Test

Read and write specific test patterns on CPU registers and check the result.

Format	void _**NUC1xx_CPU_Reg_Test** (void);
Arguments	None.
Global Value	*int CPUTestPass;* Test result will be stored in this value and main program will check its value to detect success or failure.
Return Values (Test Result)	0: FAIL 1: PASS
Related Files	cpureg_test.s

This test item uses the test pattern, 0xaaaaaaaa and 0x55555555, to verify the following registers:

- General purpose registers(R0 ~ R12)
- PRIMASK register
- CONTROL register
- SP register
- LR register
- APSR register

If any error occurs, this test will abort immediately and go for the next item.

3.2 Program Counter Test

Test whether PC can branch to the pre-defined address location or not.

Format	int **_NUC1xx_CPU_PC_Test** (void);
Arguments	None.
Global Value	None.
Return Values(Test Result)	0: FAIL 1: PASS
Related Files	programcounter_test.c iec60730.sct

This test requires a scatter file, “iec60730.sct”, to arrange the code layout.

In this file, there are two sections, pc_test_1 and pc_test_2, located in different address.

User can modify this address to fit the test environment.

```
LR_IROM1 0x00000000        {    ; load region
 ER_IROM1 0x00000000 0x00020000  {  ; load address = execution address
  *.o (RESET, +First)
  *(InRoot$$Sections)
 }

 ABS_ADDRESS1 0x160 FIXED 4 {
   programcounter_test.o (pc_test_1)
 }

 ABS_ADDRESS2 0x164 FIXED 8 {
   programcounter_test.o (pc_test_2)
 }

 ER_IROM2 +0 0x00020000  {  ; load address = execution address
  .ANY (+RO)
```

Pc_test_1 function returns a known value defined by user and pc_test_2 returns its function address. This test will check those two return values to make sure the PC branches correctly.

3.3 RAM Test

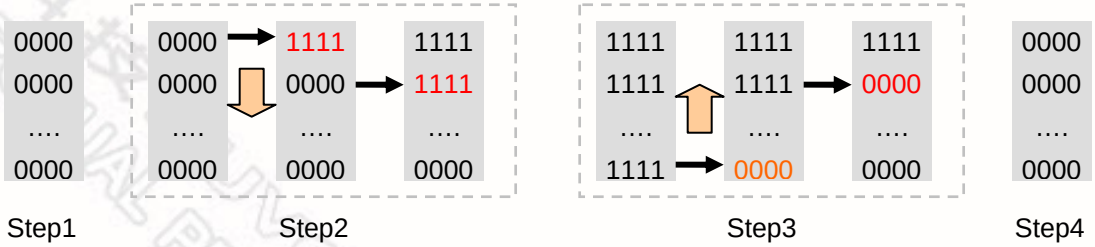
RAM memory is tested by March-X algorithm.

Format	int **_NUC1xx_RAM_MarchX_Test** (int nStartAddr, int nLength, char * cCopyToSafeArea);
Arguments	*nStartAddr* – The start address of RAM to be test. *nLength* – The length of RAM to be test. *cCopyToSafeArea* – If this argument isn't NULL, the tested area of RAM will be copied to this area that cCopyToSafeArea pointer specifics.
Global Value	None.
Return Values(Test Result)	0: FAIL 1: PASS
Related Files	ram_marchx_test.c

The data in the tested area will be lost after test. User can specify an address indicated by *cCopyToSafeArea* argument. Program will copy the data to this space and write back to the original area after test is finished.

The test procedure of March-X algorithm:

1. Clear all memory content to be zero.
2. Scanning memory content from low to high address. For each memory location, check the read bit is 0 or not. If not, the test is failed otherwise write 1 back to the memory location.
3. Scanning memory content from high to low address. For each memory location, check the read bit is 1 or not. If not, the test is failed otherwise write 0 back to the memory location
4. Check whether all bits are zero or not. The scanning direction can be either from high to low, or low to high address.



March-X Steps

3.4 RAM Test

RAM memory is tested by March-C algorithm.

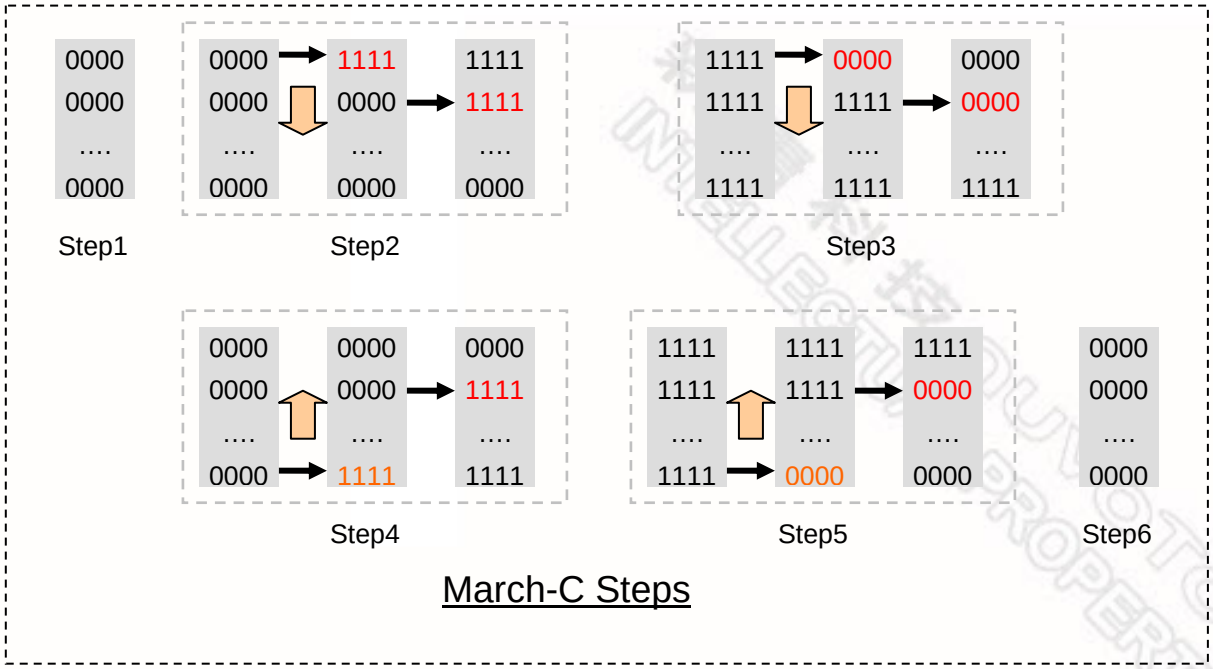
Format	int **_NUC1xx_RAM_MarchC_Test** (int nStartAddr, int nLength, char * cCopyToSafeArea);
Arguments	*nStartAddr* – The start address of RAM to be test. *nLength* – The length of RAM to be test. *cCopyToSafeArea* – If this argument isn't NULL, the tested area of RAM will be copied to this area that cCopyToSafeArea pointer specifics.
Global Value	None.
Return Values(Test Result)	0: FAIL 1: PASS
Related Files	ram_marchc_test.c

The data in the tested area will be lost after test. User can specify an address indicated by *cCopyToSafeArea* argument. Program will copy the data to this space and write back to the original area after test is finished.

March-X is a subset of March-C. User can select one of these two algorithms to test the RAM.

The test procedure of March-C algorithm:

1. Clear all memory content to be zero.
2. Scanning memory content from low to high address. For each memory location, check the read bit is 0 or not. If not, the test is failed otherwise write 1 back to the memory location.
3. Scanning memory content from low to high address. For each memory location, check the read bit is 1 or not. If not, the test is failed otherwise write 0 back to the memory location
4. Scanning memory content from high to low address. For each memory location, check the read bit is 0 or not. If not, the test is failed otherwise write 1 back to the memory location
5. Scanning memory content from high to low address. For each memory location, check the read bit is 1 or not. If not, the test is failed otherwise write 0 back to the memory location
6. Check whether all bits are zero or not. The scanning direction can be either from high to low, or low to high address.


0000
0000
....
0000
Step1
0000 → 1111
0000 → 0000
.... → ....
0000 → 0000
1111
1111
....
0000
Step2
1111 → 0000
1111 → 1111
.... → ....
1111 → 1111
0000
0000
....
1111
Step3
0000
0000
....
0000 → 1111
0000
0000
....
0000 → 1111
0000
1111
....
1111
Step4
1111
1111
....
1111 → 0000
1111
1111
....
0000 → 0000
1111
0000
....
0000
Step5
0000
0000
....
0000
Step6
March-C Steps

3.5 ROM Test

ROM memory is tested by checking CRC value.

Format	int **_NUC1xx_Flash_Test** (int nStartAddr, int nLength, int nCRCValue);
Arguments	*nStartAddr* – The start address of RAM to be test. *nLength* – a defined value that specific the length of flash to be test. *nCRCValue* – a pre-calculated CRC value used to compare with the CRC value generated at run time.
Global Value	char CRCTable[256] – the CRC table
Return Values(Test Result)	0: FAIL 1: PASS
Related Files	flash_test.c iec60730.map

The test function calculates the CRC value for the flash area specified by nStartAddr and nLength. Then it compares the CRC value with the nCRCValue to decide the test is successful or not.

A pre-calculated CRC value of the test flash area is required. To get a correct CRC value, user can run the getCRC() function (in flash_test.c) or check the output message of CRC value in advance. Then fill the correct value into *nCRCValue* argument when calling _NUC1xx_Flash_Test().

Please note that the flash test area must not include code area of main.o. User can check "iec60730.map" for the address of main.o and skip it by adjusting *nStartAddr* and *nLength*.

The following is extracted from the example iec60730.map.

```
0x00003604   0x00000260   Code   RO          1    i.main          main.o
```

Avoid this area (0x3604 ~ 0x3604+0x260)

3.6 Interrupt Test

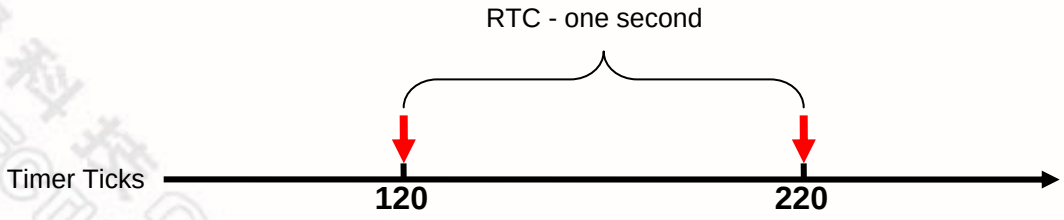
Compare timer interrupt times with RTC.

Format	int **_NUC1xx_Interrupt_Test** (void);
Arguments	None.
Global Value	int nRememberedTimerCnt – The first time tick recorded by RTC. int nTimerCnt – The second time tick recorded by RTC. Int bInterruptTestFinished – Flag to indicate whether test is finished or not. int bFirstIn – Flag to indicate the first time entering RTC ISR. Int bTestResult – The final test result.
Return Values(Test Result)	0: FAIL 1: PASS
Related Files	interrupt_test.c

The interrupt is tested by sampling timer interrupt count with RTC interrupt. The timer interrupt frequency is 100 Hz while RTC interrupt frequency is 1 Hz.

The RTC ISR records the timer tick value at 1st and 2nd entry. The real time interval is 1 second, and the timer tick count should be 100. If the difference of these timer tick values is larger than 101 or smaller than 99, the test is failed.

The following figure shows the test method.



3.7 Clock Test

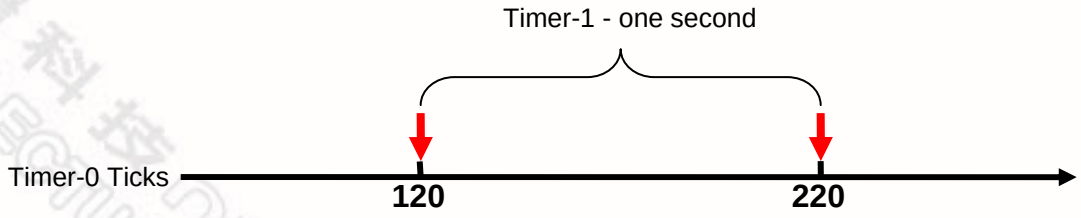
Compare different channel of timer by using different clock source.

Format	int **_NUC1xx_Clock_Test** (void);
Arguments	None.
Global Value	int nRememberedTimerCnt – The first time tick recorded by RTC. int nTimerCnt – The second time tick recorded by RTC. Int bInterruptTestFinished – Flag to indicate whether test is finished or not. int bFirstIn – Flag to indicate the first time entering RTC ISR. Int bTestResult – The final test result.
Return Values(Test Result)	0: FAIL 1: PASS
Related Files	clock_test.c

The clock test uses two timer channels having different clock sources. The Timer 0 interrupt frequency is 100 Hz while Timer 1 interrupt frequency is 1 Hz. The test result is decided by sampling Timer 0 interrupt count with Timer 1 interrupt.

The Timer-1 ISR records the timer tick value of Timer 0 at 1st and 2nd entry. The real time interval is 1 second, and the Timer 0 tick count should be 100. If the difference of these timer tick values is larger than 101 or smaller than 99, the test is failed.

The following figure shows the test method.



4 REVISION HISTORY

REV.	DATE	DESCRIPTION
1.00	Oct 15, 2010	Initially issued.

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Further more, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

Please note that all data and specifications are subject to change without notice. All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.