LPC Boot ROM checksum

In newer version of LPC Boot ROM , checksum is added in location 7 (offset 0x0000001C in vector table) of Boot ROM. For some LPC device, for example LPC8N04, older Boot ROM version 0.12 (equipped in Rev B board) doesn't contain checksum but newer version 0.14(equipped in Rev C board) adds it. Boot ROM checksum is a criteria for valid user code. Bootloader can jump to user code only when detects checksum value correct. Otherwise, it stays at boot code.

Here is a description of criteria for valid user code, I extracted from LPC8N04 user manual. We can find similar description in most of other LPC UM.

> The bootloader criterion for valid user code is checked by the Boot ROM (version 0.14) where the reserved Cortex-M0+ exception vector location 7 (offset 0x 0000 001C in the vector table) should contain the 2's complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The bootloader code checksums the first 8 locations in sector 0 of the flash. If the result is 0, then execution control is transferred to the user code.

Scenario:

User may have this annoying problem: the program runs well if download code to LPC flash with debugger. However power off and power on agian, the code won't run any more.

If you also experience same in field, you may consider the possibility of Boot ROM checksum failed.

In some of the old LPC demo code, checksum value isn't set on vector table location 7. Default value 0 is set but it is not we want.

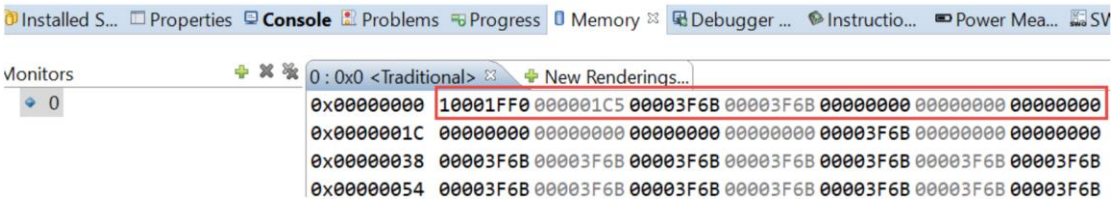
```
void (* const g_pfnVectors[])(void) = {
    &_vStackTop,          /* Handler for EXCEPTION0  @0x00000000 - The initial
    ResetISR,             /* Handler for EXCEPTION1  @0x00000004 - The reset ha
    NMI_Handler,          /* Handler for EXCEPTION2  @0x00000008 - The NMI hanc
    HardFault_Handler,    /* Handler for EXCEPTION3  @0x0000000C - The hard fau
    0,                    /* Handler for EXCEPTION4  @0x00000010 - Reserved */
    0,                    /* Handler for EXCEPTION5  @0x00000014 - Reserved */
    0,                    /* Handler for EXCEPTION6  @0x00000018 - Reserved */
    0,                    /* Handler for EXCEPTION7  @0x0000001C - Reserved */
    0,                    /* Handler for EXCEPTION8  @0x00000020 - Reserved */
    0,                    /* Handler for EXCEPTION9  @0x00000024 - Reserved */
    0,                    /* Handler for EXCEPTION0  @0x00000028 - Reserved */
    SVC_Handler,          /* Handler for EXCEPTION11 @0x0000002C - SVCall hand
    0,                    /* Handler for EXCEPTION12 @0x00000030 - Reserved */
```

Solution:

To get pass the BOOT ROOM checksum, checksum value at address 0x1C need to be filled at vector table. There are at least two methods to get the checksum value:

Method 1. calculate checksum value by hand

- launch debugger, add first 7 vector values together, then reverse:



0x10001FF0+0x00001C5+0x00003F6B+0x00003F6B+0+0+0=0x1000A08B

0-1000A08B = **0xEFFF5F75**

By this way, we get **0xEFFF5F75** which is the correct checksum value.

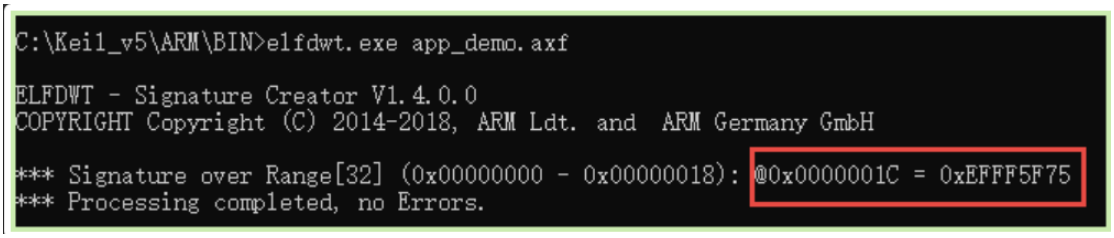
Method 2. Use Keil tool elfdwt.exe generate checksum value

You need install Keil IDE fist. the tool is under: C:/Keil_v5/ARM/BIN

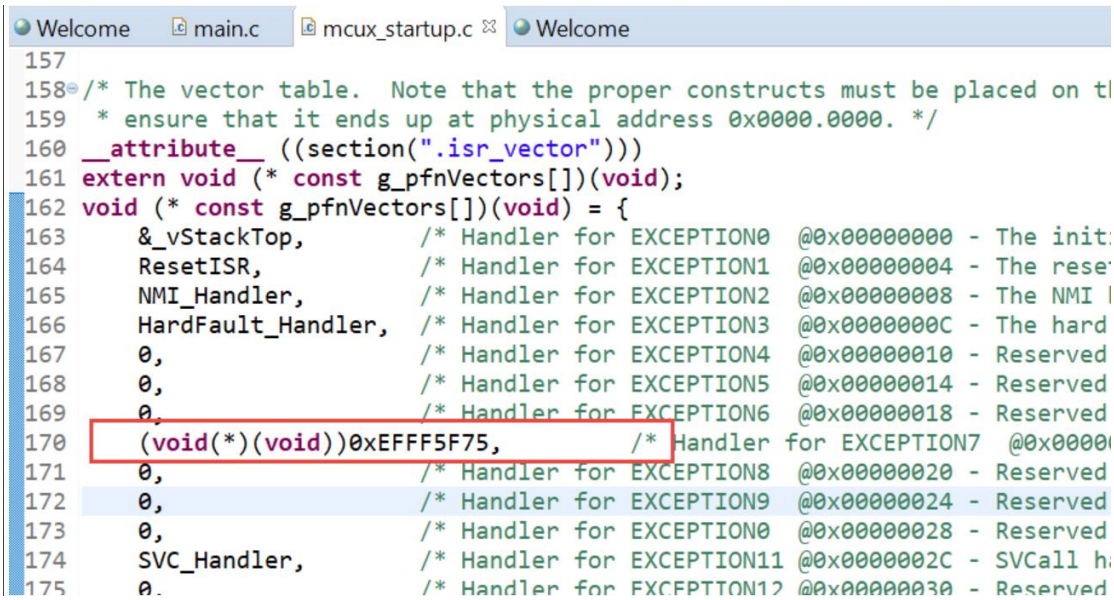
input command: >elfdwt.exe app_demo.axf

here, "app_demo.axf" is the mcuxpresso demo code generated executable file

After executing this command, checksum will be displayed as 0xEFFF5F75.



Next, fill the checksum value at address 0x1C in vector table:



build and launch debugger, This revision should fix the problem.

If user uses MDK Keil, we can add checksum value with elfdwt.exe via after build action: