Interrupt Vector Redirection Method in Kinetis

Interrupt technology is an important method to improve the efficiency of real-time CPU. In Kinetis, interrupt vector table is allocated in lowest ROM address of whole memory.

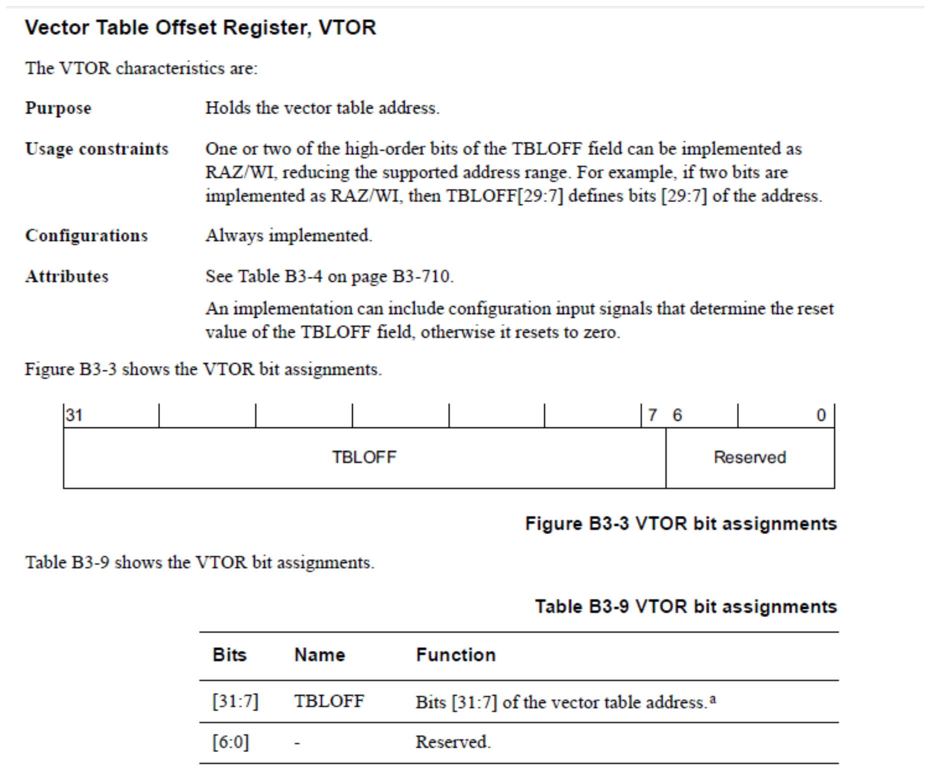
For instance,

in KL25Z128, interrupt vector table is 0x00000000 - 0x000000BF

in K40DX256, interrupt vector table is 0x00000000 - 0x000001DF

However in some application, we need move vector table to other address. A typical usage is developing boot loader code. The boot loader code occupies the first region of the FLASH memory (the lowest memory address space starting from 0x0000000).
This placement moves at the beginning of the available memory space and it is necessary to shift this address in the user application.

The CM0+ and CM4 core adds support for a programmable Vector Table Offset Register (VTOR) to relocate the exception vector table, which is not existing in Freescale 8/16bit MCU. This device supports booting from internal flash and RAM.
Kinetis supports booting from internal flash with the reset vectors located at addresses 0x0 (initial SP_main), 0x4 (initial PC), and RAM with relocating the exception vector table to RAM.
This is the description of VTOR register:



Vector Table Offset Register, VTOR

The VTOR characteristics are:

Purpose	Holds the vector table address.
Usage constraints	One or two of the high-order bits of the TBLOFF field can be implemented as RAZ/WI, reducing the supported address range. For example, if two bits are implemented as RAZ/WI, then TBLOFF[29:7] defines bits [29:7] of the address.
Configurations	Always implemented.
Attributes	See Table B3-4 on page B3-710. An implementation can include configuration input signals that determine the reset value of the TBLOFF field, otherwise it resets to zero.

Figure B3-3 shows the VTOR bit assignments.

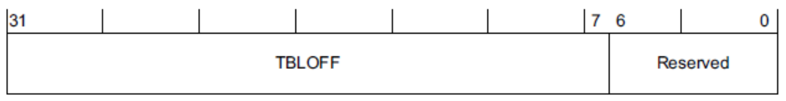


Figure B3-3 VTOR bit assignments

Table B3-9 shows the VTOR bit assignments.

Table B3-9 VTOR bit assignments

Bits	Name	Function
[31:7]	TBLOFF	Bits [31:7] of the vector table address.[a]
[6:0]	-	Reserved.

This is the sample of implementing KL25Z128 vector redirection under CW10.6 to allocate vector table from 0x00000000 to 0x00001000

1. Modify linker file *.ld
   Original ld file:

```
MEMORY
{
  m_interrupts  (rx) : ORIGIN = 0x00000000, LENGTH = 0xC0
  m_cfmprotrom  (rx) : ORIGIN = 0x00000400, LENGTH = 0x10
  m_text        (rx) : ORIGIN = 0x00000800, LENGTH = 128K - 0x800
  m_data       (rwx) : ORIGIN = 0x1FFFF000, LENGTH = 16K        /* SRAM */
}
```

   Modified linker file *.ld after vector relocation:

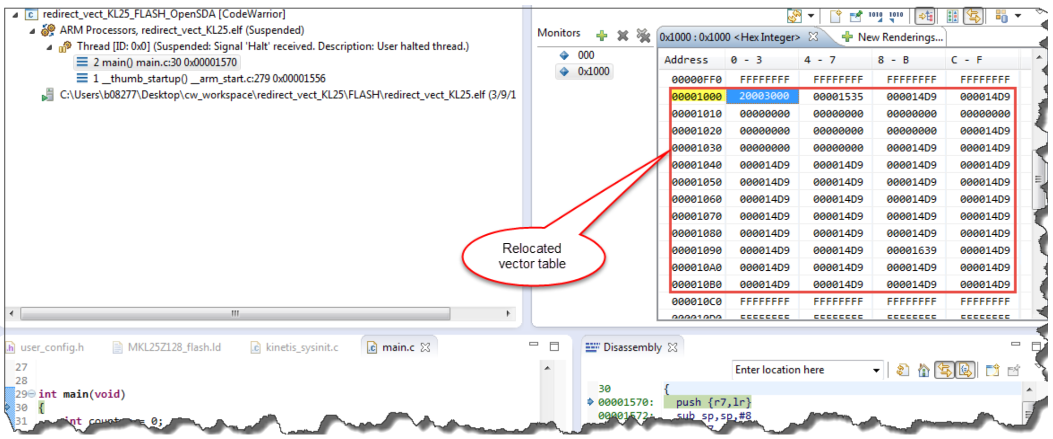
```
MEMORY
{
  m_interrupts  (rx) : ORIGIN = 0x00001000, LENGTH = 0xC0
  m_cfmprotrom  (rx) : ORIGIN = 0x00001400, LENGTH = 0x10
  m_text        (rx) : ORIGIN = 0x00001410, LENGTH = 128K - 0x1410
  m_data       (rwx) : ORIGIN = 0x1FFFF000, LENGTH = 16K        /* SRAM */
}
```

2. Set SCB_VTOR resigter.
   In CW10.6, If the project is created with project wizard, setting SCB_VTOR register is done in default generated kinetis_sysinit.c, __init_hardware() function.

```
void __init_hardware()
{
    SCB_VTOR = (uint32_t)__vector_table; /* Set the interrupt vector table position */
    // Disable the Watchdog because it may reset the core before entering main().
    SIM_COPC = KINETIS_WDOG_DISABLED_CTRL;
}
```

Here __vector_table is interrupt vector address which is defined in linker file *.ld memory allocation section.

I also attach the demo code for vector relocation. It includes all I am talking above. Below is the testing result in debugger: vector table is relocated to 0x1000. Interrupt can work well after vector relocation.



```
redirect_vect_KL25_FLASH_OpenSDA [CodeWarrior]
  ARM Processors, redirect_vect_KL25.elf (Suspended)
    Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)
      2 main() main.c:30 0x00001570
      1 __thumb_startup() __arm_start.c:279 0x00001556
  C:\Users\b08277\Desktop\cw_workspace\redirect_vect_KL25\FLASH\redirect_vect_KL25.elf (3/9/1
```

```
Address   0 - 3     4 - 7     8 - B     C - F
00000FF0  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
00001000  20003000  00001535  000014D9  000014D9
00001010  00000000  00000000  00000000  00000000
00001020  00000000  00000000  00000000  000014D9
00001030  00000000  00000000  000014D9  000014D9
00001040  000014D9  000014D9  000014D9  000014D9
00001050  000014D9  000014D9  000014D9  000014D9
00001060  000014D9  000014D9  000014D9  000014D9
00001070  000014D9  000014D9  000014D9  000014D9
00001080  000014D9  000014D9  000014D9  000014D9
00001090  000014D9  000014D9  00001639  000014D9
000010A0  000014D9  000014D9  000014D9  000014D9
000010B0  000014D9  000014D9  000014D9  000014D9
000010C0  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF
```

```
int main(void)
{
    int count = 0;
```

```
30          {
00001570:     push {r7,lr}
00001572:     sub sp,sp,#8
```