Hi All,
I am during writing bootloader for STLUX processor (base on STM8). Both, bootloader and main program uses own interrupt vector table. The STM8 processor has fixed interrupt vector table at 0x8000 address.

I want to redirect from fixed vector table to vector table storing in RAM. I wrote the code to copy vector table from bootloader or main program to RAM and it is work fine,
but I have problem with redirection. I do not know how can I do this.

I use an IAR.

I did an initialization a fixed vector table like this:

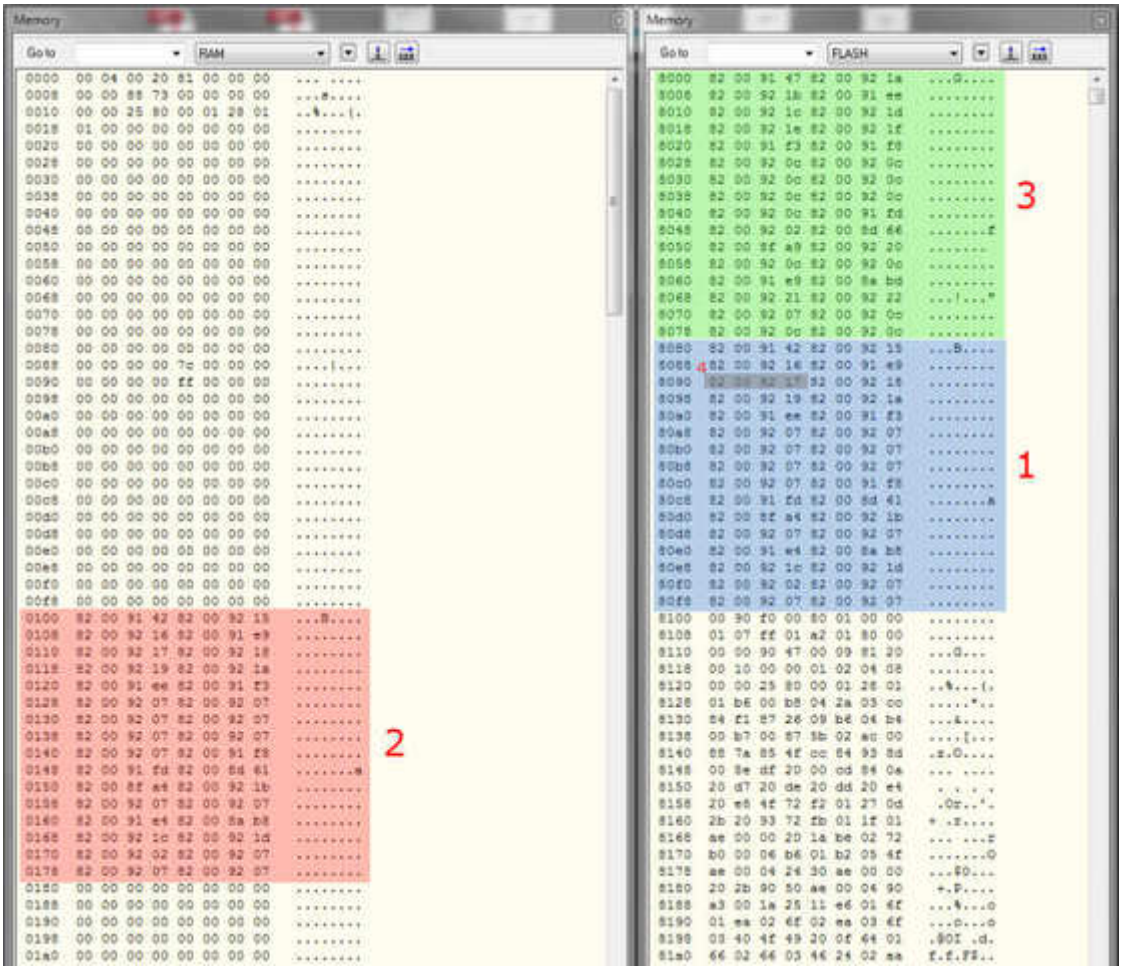
```
typedef void (__far_func * interrupt_handler_t)(void);

struct interrupt_vector {
    unsigned char interrupt_instruction;
    interrupt_handler_t interrupt_handler;
};

struct interrupt_vector RamISR_IRQ[32] @ 0x00100;

struct interrupt_vector const _vectab[] @ 0x8000 =   {
    {0x82, (interrupt_handler_t)(RamISR_IRQ)}, /* reset */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 1)}, /* trap  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 2)}, /* irq0  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 3)}, /* irq1  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 4)}, /* irq2  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 5)}, /* irq3  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 6)}, /* irq4  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 7)}, /* irq5  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 8)}, /* irq6  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+ 9)}, /* irq7  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+10)}, /* irq8  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+11)}, /* irq9  */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+12)}, /* irq10 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+13)}, /* irq11 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+14)}, /* irq12 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+15)}, /* irq13 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+16)}, /* irq14 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+17)}, /* irq15 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+18)}, /* irq16 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+19)}, /* irq17 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+20)}, /* irq18 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+21)}, /* irq19 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+22)}, /* irq20 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+23)}, /* irq21 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+24)}, /* irq22 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+25)}, /* irq23 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+26)}, /* irq24 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+27)}, /* irq25 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+28)}, /* irq26 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+29)}, /* irq27 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+30)}, /* irq28 */
    {0x82, (interrupt_handler_t)(RamISR_IRQ+31)}, /* irq29 */
};
```

The result:



1. Vector table from bootloader
2. Vector table in RAM
3. STM8 fixed vector table.

As You see, The vector table from bootloader and ram is identical, so it is ok, but the values in fixed vector table are incorrect. I do not know why :-(

Any idea?

Update:
When I manually set fixed vector table (0x8000 to 0x8080) to 0x00 (clear it, green color in above image) I see it is not programming by IDE (all bytes in section are still zero). Of course, I still uses a code attached above.

*icf file:

```
//////////////////////////////////////////////////////////////////
//      Example ILINK command file for
//      STM8 IAR C/C++ Compiler and Assembler.
//
//      Copyright 2014 IAR Systems AB.
//
//////////////////////////////////////////////////////////////////

define memory with size = 16M;

define region TinyData = [from 0x00 to 0xFF];

//define region RamVector = [from 0x00100 to 0x017F];
```

```
define region NearData = [from 0x00100 to 0x07FE];

define region Eeprom = [from 0x4000 to 0x43FF];

define region BootROM = [from 0x6000 to 0x67FF];

define region RedirectVector = [from 0x8000 to 0x8080];

define region BootloaderVector = [from 0x8080 to 0x8100];

define region NearFuncCode = [from 0x8080 to 0xFFFE];

define region FarFuncCode = [from 0x8080 to 0xFFFE];

define region HugeFuncCode = [from 0x8080 to 0xFFFE];


////////////////////////////////////////////////////////////////

define block CSTACK with size = _CSTACK_SIZE  {};

define block HEAP  with size = _HEAP_SIZE {};

define block BOOTLOADERINTVEC with size = 0x80 { ro section .intvec };

define block REDIRECTINTVEC with size = 0x80 { ro section .rivector };

// Initialization
initialize by copy { rw section .far.bss,
                     rw section .far.data,
                     rw section .far_func.textrw,
                     rw section .huge.bss,
                     rw section .huge.data,
                     rw section .huge_func.textrw,
                     rw section .iar.dynexit,
                     rw section .near.bss,
                     rw section .near.data,
                     rw section .near_func.textrw,
                     rw section .tiny.bss,
                     rw section .tiny.data,
                     ro section .tiny.rodata };

initialize by copy with packing = none {section __DLIB_PERTHREAD };

do not initialize  { rw section .eeprom.noinit,
                     rw section .far.noinit,
                     rw section .huge.noinit,
                     rw section .near.noinit,
                     rw section .tiny.noinit,
                     rw section .vregs };

// Placement
place at start of TinyData      { rw section .vregs };
place in TinyData               { rw section .tiny.bss,
                                  rw section .tiny.data,
                                  rw section .tiny.noinit,
                                  rw section .tiny.rodata };

place at end of NearData        { block CSTACK };
place in NearData               { block HEAP,
                                  rw section __DLIB_PERTHREAD,
                                  rw section .far.bss,
                                  rw section .far.data,
                                  rw section .far.noinit,
                                  rw section .far_func.textrw,
                                  rw section .huge.bss,
```

```
                                  rw section .huge.data,
                                  rw section .huge.noinit,
                                  rw section .huge_func.textrw,
                                  rw section .iar.dynexit,
                                  rw section .near.bss,
                                  rw section .near.data,
                                  rw section .near.noinit,
                                  rw section .near_func.textrw };

place in BootloaderVector        { block BOOTLOADERINTVEC };

place in RedirectVector          { block REDIRECTINTVEC };

place in NearFuncCode           { ro section __DLIB_PERTHREAD_init,
                                  ro section .far.data_init,
                                  ro section .far_func.textrw_init,
                                  ro section .huge.data_init,
                                  ro section .huge_func.textrw_init,
                                  ro section .iar.init_table,
                                  ro section .init_array,
                                  ro section .near.data_init,
                                  ro section .near.rodata,
                                  ro section .near_func.text,
                                  ro section .near_func.textrw_init,
                                  ro section .tiny.data_init,
                                  ro section .tiny.rodata_init };

place in FarFuncCode            { ro section .far.rodata,
                                  ro section .far_func.text };

place in HugeFuncCode           { ro section .huge.rodata,
                                  ro section .huge_func.text };

place in Eeprom                 {    section .eeprom.noinit };

place in Eeprom                 {    section .eeprom.data };

place in Eeprom                 {    section .eeprom.rodata };

//////////////////////////////////////////////////////////////////
```

Thanks for any help.