BootROM, miniloader and threadx are all running on R4 core.

But BootROM and miniloader use high vector address (0xFFFF,0000) to save exception vector table.

in pcsgit/miniloader/source/main/main_entry.c

```
1.  typedef void (*ROM_restart_entry_func_t) (void);
2.  ROM_restart_entry_func_t ROM_restart = (ROM_restart_entry_func_t)0xFFFF0000;
3.
4.  minPrintf("\nRe-Entering eROM Code\n");
5.  ROM_restart();
```

miniloader reboot R4 core by jumping to 0xFFFF,0000. The first entry is reset exception in exception vector table.

But because threadx is running in the memory region from 96M to 100M. The other memory space is occupied by Linux kernel. The R4's exception vector table need to be remapped to the new memory space between 96M and 100M.

in pcsgit/r4/oem/marvell/88pa6220_r4/init/src/main.c

```
1.  int main( void )
2.  {
3.      unsigned long nptrs = (unsigned long) __CTOR_LIST__[0];
4.      unsigned i;
5.
6.      /* Call C++ static/global constructors */
7.      for (i = nptrs; i >= 1; i--)
8.      {
9.          __CTOR_LIST__[i]();
10.     }
11.
12.     /* this must run before cpu_init because some features inited here are used i
    n cpu */
13.     hwConfigInit();         ①
14.
15.     cpu_init();
16.
17.     /* This routine does not return */
18.     tx_kernel_enter();
19.
20.     /* assert if it ever does! */
21.     ASSERT(0);
22.
23.     return -1;
24. }
```

①

```
1.  void hwConfigInit(void)
2.  {
3.  //#ifdef HAVE_SERENITY
4.      if (hwIsFPGA())
5.      {
6.          FPGAA_FREQ_REGS_t   *fpga1_freq = (FPGAA_FREQ_REGS_t*)FPGAA_FREQ_BASE;
7.          hw_config_table.proc_speed_mhz = fpga1_freq->Freq_clkdiv2;
8.          hw_config_table.bus_speed_mhz = fpga1_freq->Freq_clk;
9.          hw_config_table.platform_mem_size = (uint32_t)&__ram_size__;
10.         hw_config_table.scan_clock_mhz = 64;
11.
12.         // map 64K of logical 0 into our physical space
13.         ((FPGAC_CONFIG_REGS_t*)FPGAC_CONFIG_BASE)->BCMlowvec=((uint32_t)&__start_
    ram__ | 1);
14.     }
15.     else
16.     {
17.     //#else
18.         hw_config_table.program_table       = 0;  // TBD
19.         hw_config_table.proc_speed_mhz      = 400;  // TBD
20.         hw_config_table.platform_mem_size   = (uint32_t)&__ram_size__;
21.         hw_config_table.bus_speed_mhz       = 400;  // TBD --- AP busclk or IPS b
    usclk
22.         hw_config_table.scan_clock_mhz      = 250;  // TBD
23.
24.         // map 64K of logical 0 into our physical space
25.         ((CIU_REGS_t*)AP_CIU_BASE)->R4_Vector_Remap = CIU_R4_VECTOR_REMAP_R4_VECT
    OR_REMAP_ADDRESS_REPLACE_VAL(((CIU_REGS_t*)AP_CIU_BASE)->R4_Vector_Remap,(((uint3
    2_t)&__start_ram__)>>16));                ②
26.         ((CIU_REGS_t*)AP_CIU_BASE)->R4_Vector_Remap = CIU_R4_VECTOR_REMAP_R4_VECT
    OR_REMAP_ENABLE_REPLACE_VAL(((CIU_REGS_t*)AP_CIU_BASE)->R4_Vector_Remap,1);
                                                                                ③
27.     }
28.
29.     hwUartInit();
30. //#endif // HAVE_SERENITY
31.
32.     /* initialize the timebase configuration with the bus frequency */
33.     hwTimebaseInit();
34.
35.     // wtm module needs info....
36.     // APB_CONFIG_REGS_t changed, items removed.  Fuse_Bank shifted to lower addr
    ess (previous ASIC, so all ASICs will set this up)
37.     wtm_asic_reg_fuse_bank0_r8 = (uint32_t)&(((APB_CONFIG_REGS_t*)AP_AP_APB_APB_C
    ONFIG_BASE)->Fuse_Bank0_R8);
38.     // default pwr level for WTM can configured after WTM lib is built, so use va
    riable to store it.
39.     wtm_asic_default_pwr_level = PWRMGR_WTM_LEVEL;
40.
41. }
```

②

set R4's new vector table address to __start_ram symbol address.

__start_ram symbol is defined in pcsgit/r4/oem/marvell/88pa6220_r4/buildtools/memory_map.ld

```
MEMORY
{
    /* NOTE - the ram line here is rewritten by the makefile before the linker
              script is used.  If you want to move the code, specify
              RTOS_MEM_START=... and RTOS_MEM_SIZE=... on the makeline */
    ram (rwx): ORIGIN = 0x0, LENGTH = 0x0
    squ0 (rw ) : ORIGIN = 0xD1000000, LENGTH = 32K  /* SQU0 32K - TBD*/
    squ1 (rw ) : ORIGIN = 0xD1008000, LENGTH = 32K  /* SQU1 32K - TBD*/
    squ2 (rw ) : ORIGIN = 0xD1010000, LENGTH = 32K  /* SQU2 32K - TBD*/
}

OUTPUT_ARCH(arm)
ENTRY(reset)
EXTERN(reset)         /* make sure this gets linked in */

SECTIONS
{
    __start_ram__     = ORIGIN(ram);            /* all starts with our anchor */
    __ram_size__      = LENGTH(ram);            /* all starts with our anchor */
    __vectors__       = __start_ram__;          /* vectorsall starts with our anc
hor */

    /* Place vector table. */
    vectors __vectors__ :
    {
        *(.text.vectors)

    } > ram

......
```

__start_ram__ is the head of "ram" memory region, and vector table is placed on "ram".

But "ram" is not defined staticly in memory_map.ld.

We could get __start_ram__ symbol address from the elf file.

walterzh$ arm-linux-gnueabi-nm tx.elf | grep __start_ram__

06000000 T __start_ram__

walterzh$ arm-linux-gnueabi-nm tx.elf | grep __ram_size__

00400000 A __ram_size__

So in memory_map.ld

ram (rwx): ORIGIN = 0x0, LENGTH = 0x0

==>

ram (rwx): ORIGIN = 0x06000000, LENGTH = 0x00400000

0x06000000 = 96M

0x00400000 = 4M

③

enable R4's vector table remap functionality.