Hal/6220/build/hal_vectors_gnu.asm (entry)

```
    @ENTRY

    .global ExceptionTable
ExceptionTable:
    LDR PC, ResetAddr
    LDR PC, UndefinedAddr
    LDR PC, SwiAddr
    LDR PC, PrefetchAddr
    LDR PC, AbortAddr
    NOP
    LDR PC, IrqAddr
@   LDR PC, FiqAddr

    SUB r14,r14,#4          @// Adjust lr to proper address.
    STMFD   r13!,{r0-r7,r12,r14}        @// Save registers to stack.

    LDR r12,=Hal_Fsr
    MOV r14,pc              @// Store the return address (=pc+4) in r14 (=lr
).
    BX  r12             @// Branch to address pointed to by r12.

    LDMFD   r13!,{r0-r7,r12,pc}^        @// Restore registers and return to
code that was interrupted.
```

显然ExceptionTable必须位于96M memory的边界处，因为r4 core在"reset"后被设置成从这儿开始取指运行。

```
Disassembly of section .reset:

06000000 <ExceptionTable>:

    @ENTRY

    .global ExceptionTable
ExceptionTable:
    LDR PC, ResetAddr
 6000000:   e59ff02c    ldr pc, [pc, #44]   ; 6000034 <ResetAddr>
    LDR PC, UndefinedAddr
 6000004:   e59ff02c    ldr pc, [pc, #44]   ; 6000038 <UndefinedAddr>
    LDR PC, SwiAddr
 6000008:   e59ff02c    ldr pc, [pc, #44]   ; 600003c <SwiAddr>
    LDR PC, PrefetchAddr
 600000c:   e59ff02c    ldr pc, [pc, #44]   ; 6000040 <PrefetchAddr>
    LDR PC, AbortAddr
 6000010:   e59ff02c    ldr pc, [pc, #44]   ; 6000044 <AbortAddr>
    NOP
 6000014:   e320f000    nop {0}
    LDR PC, IrqAddr
 6000018:   e59ff02c    ldr pc, [pc, #44]   ; 600004c <IrqAddr>
@   LDR PC, FiqAddr

    SUB r14,r14,#4          @// Adjust lr to proper address.
 600001c:   e24ee004    sub lr, lr, #4
    STMFD   r13!,{r0-r7,r12,r14}        @// Save registers to stack.
 6000020:   e92d50ff    push    {r0, r1, r2, r3, r4, r5, r6, r7, ip, lr}

    LDR r12,=Hal_Fsr
 6000024:   e59fc048    ldr ip, [pc, #72]   ; 6000074 <FiqHandler>
    MOV r14,pc              @// Store the return address (=pc+4) in r14 (=lr
).
 6000028:   e1a0e00f    mov lr, pc
    BX  r12             @// Branch to address pointed to by r12.
 600002c:   e12fff1c    bx  ip

    LDMFD   r13!,{r0-r7,r12,pc}^        @// Restore registers and return to
code that was interrupted.
 6000030:   e8fd90ff    ldm sp!, {r0, r1, r2, r3, r4, r5, r6, r7, ip, pc}^

```

Hal/6220/build/hal_init_gnu.asm

Hal/6220/build/hal_2nd_init_gnu.asm

```
InvokeMain:
    MOV r1, #0 @ argv == NULL
    MOV r0, #0 @ argc == 0
    BL main
```

Hal/6220/build/hal_main.c

```
int main(void)
{
......

    uartPreInit(&_uart);

 static UART_CONFIG _config = { 115200,
              UART_DATA_BITS_8,
              UART_STOP_BITS_1,
              UART_PARITY_NONE };

    uartSetConfig(_uart, &_config);

    AppInit_Initialize();

    while ( 1 )
        ;
}
```

AppInit_Initialize() is in ApplicationMrvl/init/app_init.c

```
void AppInit_Initialize(void)
{
    SYS_INIT_CFG *cfg = { SYS_INIT_USECASE_SDK };


    const volatile uint8_t * hook = (const volatile uint8_t *)&set_speed[0];
    // chip speed is application specific - must be done before any other hw init or scheduler start
    if (*hook == 4) SysApiSystem_set_speed(2);      // fast
    if (*hook == 2) SysApiSystem_set_speed(1);
    if (*hook == 1) SysApiSystem_set_speed(0);
    // if none of bit[0:2] is set no speed change happens


    /*
    we have no tool to set the print mask before firmware start, or to modify it from outside after firmware start. driver?
    this is a hack to enable/disable output by patching the firmware binary - look for the "PrintBuf" string, etc..
    */
    uint32_t suppress_dbgcomp = 0;
    suppress_dbgcomp = APP_DBGCOMP_LAN | APP_DBGCOMP_LAN_RX | APP_DBGCOMP_LAN_TX | APP_DBGCOMP_LAN_ISR;     // this is a lot of ouput. you dont want to see it ususlly.
    hook = (const volatile uint8_t *)&initial_print_mask[0];
    if (*hook=='0') SysApiPrint_ConfigFlags(0xffffffff ^ suppress_dbgcomp, APP_DBGCAT_NO);  // switch off debug prints
    if (*hook=='1') SysApiPrint_ConfigFlags(0xffffffff ^ suppress_dbgcomp, APP_DBGCAT_ALL); // switch on most(?) of the debug prints
    if (*hook=='2') SysApiPrint_ConfigFlags(0xffffffff, APP_DBGCAT_ALL);
                    // switch on all debug prints
    // other values of initial_print_mask[0] will leave the debug print masks unchanged


    SysApiInit_Initialize(cfg);

}
```

in Sys/init/sys_init.c

```
void SysApiInit_Initialize(SYS_INIT_CFG* Cfg)
{

    HalApiMemory_RamInit();

    SysPrint_Init();


    if (HalApiReset_WasCold()) {
        /*
        sram memory has random content after power on (or clearing reset?).
        thats why uninitialized variables have also random values.
        in this case we do not print anything at this point.
        */
    } else {
        /* this is a warm start. we print using masks from previous run. */
        SysApiPrint_Write_varg("I", 0xffffffff, 0xffffffff, "%s/%d %s %s (%s
)\n", __func__, __LINE__, __DATE__, __TIME__, Version_Info);
    }

    const volatile uint8_t dump_nvm[] = "0=dump_nvm";
    if (dump_nvm[0]=='1')       HalApiMemory_dump_nvm();


#ifdef OS_threadx

    tx_kernel_enter();
#elif defined OS_freertos
    intInit();

    SysApi_InitApplModules(NULL);
    extern uint32_t pxCurrentTCB;
    SysApiPrint_Write_varg("I", 0xffffffff, 0xffffffff, "%s/%d pxCurrentTCB=
%08x\n", __func__, __LINE__, pxCurrentTCB);
    vTaskStartScheduler();

#endif
    while ( 1 )
        ;
}
```

> vTaskStartScheduler();

启动freertos embedded OS.