CMPE 443
Principles of Embedded System Design
fall’11

Problem Session
**Overview**

- **Interrupts**
  - Initializing Interrupt Vector Locations
  - Enabling & Initializing Interrupts
  - The Interrupt Service Routine (ISR)
Initializing Interrupt Vector Locations

- To place the interrupt service routines address in the correct vector location, the linker parameter file is used
  - Two ways to do this
    - Use address of the vector location
      - VECTOR ADDRESS 0xFFF6 SWI_ISR
    - Use priority number of the vector
      - VECTOR 4 SWI_ISR

- Alternatively, include the interrupt vector number in the function definition
  - void interrupt 4 SWI_ISR(void) {...}

*Reference:* Table 12-1 Interrupt Vector Assignments
Sample PRM file

NAMES END /* CodeWarrior will pass all the needed files to the linker by command line. But here you may add your own files too. */

SEGMENTS /* Here all RAM/ROM areas of the device are listed. Used in PLACEMENT below. */
  RAM = READ_WRITE 0x0400 TO 0xFFF;
  ROM_4000 = READ_ONLY 0x4000 TO 0x7FFF;
  ROM_C000 = READ_ONLY 0xC000 TO 0xFEFF;
  PAGE_38 = READ_ONLY 0x388000 TO 0x38BFFF;
  PAGE_39 = READ_ONLY 0x398000 TO 0x39BFFF;
  PAGE_3A = READ_ONLY 0x3A8000 TO 0x3ABFFF;
  PAGE_3B = READ_ONLY 0x3B8000 TO 0x3BBFFF;
  PAGE_3C = READ_ONLY 0x3C8000 TO 0x3CBFFF;
  PAGE_3D = READ_ONLY 0x3D8000 TO 0x3DBFFF;
END

PLACEMENT /* here all predefined and user segments are placed into the SEGMENTS defined above. */
  _PRESTART, /* Used in HIWARE format: jump to _Startup at the code start */
  STARTUP, /* startup data structures */
  ROM_VAR, /* constant variables */
  STRINGS, /* string literals */
  VIRTUAL_TABLE_SEGMENT, /* C++ virtual table segment */
  NON_BANKED, /* runtime routines which must not be banked */
  COPY /* copy down information: how to initialize variables */
  INTO ROM_C000/*, ROM_4000*/;
  DEFAULT_ROM INTO PAGE_38, PAGE_39, PAGE_3A, PAGE_3B, PAGE_3C, PAGE_3D;
  SSTACK, /* allocate stack first to avoid overwriting variables on overflow */
  DEFAULT_RAM INTO RAM;
END

STACKSIZE 0x100

VECTOR 0 _Startup /* reset vector: this is the default entry point for a C/C++ application. */
VECTOR 4 SWI_ISR

Reference: Table 12-1 Interrupt Vector Assignments
Global mask bit, called the I bit in the CCR, is used to mask (disallow) or unmask (allow) all interrupts.

Each interrupting subsystem may also have a local enable bit used to enable (allow) or disable (disallow) that device from interrupting

- Ex: the IRQEN bit of the INITCR register is set to enable IRQ_L interrupt.

Initialize the registers that control the interrupt parameters

- Ex: IRQE bit of the INITCR register selects edge sensitivity for IRQ_L interrupt
  - IRQE=0 : IRQ_L configured for low level recognition (default)
  - IRQE=1 : IRQ_L configured for falling edges

Reference: Chapter 12-7 External interrupt sources
The Interrupt Service Routine (ISR)

- Requires,
  - The vector has been initialized properly
  - Interrupts have been unmasked and enabled

- Important Hints
  - Clear the interrupt generating flags
    - Prevent generating interrupts continuously
  - The CPU registers are pushed onto the stack before ISR
    - At the end of the ISR, the content of the registers will be automatically restored
  - Keep it short
    - Reduce the latency in servicing other interrupts
The wait instruction (WAI) waits for an interrupt service
  ◦ Pushes all the registers onto the stack in preparation for a subsequent interrupt
    • Reduced delay in executing the ISR
  ◦ Places the into WAIT mode (but the system clock signals continue to run)
    • Reduced power consumption