Interrupts

Additional Details and Examples
Other types of interrupts

- **/XIRQ**
  - The /XIRQ pin can cause an interrupt (similar to the /IRQ pin)
  - It can be enabled by clearing the X bit in the condition code register (bit 6)
  - It’s called “nonmaskable” because once enabled, you can’t disable it

- **Unimplemented opcode trap**
  - If the CPU tries to execute an invalid machine code opcode, it causes an interrupt

- **SWI (software interrupt)**
  - This instruction, SWI, causes an interrupt
Resets

• Resets are similar to interrupts
  – They also cause control to be transferred to a location specified in the vector table
  – However, it doesn’t bother saving registers (because control won’t ever be transferred back)

• Resets can be caused by
  – Power on
  – The reset pin being asserted
  – COP (computer operating properly) failure
  – Clock monitor reset
Special issue – Serial monitor

- There is a special problem in initializing the vector table when using the serial monitor program
  - The serial monitor is the program on the HCS12 that you talk to when you load your program using the debugger, or step through your program

- The serial monitor is in ROM, in locations $F800..$FFFF
  - However, the vector table is predefined to be at $FF80..$FFFF
  - You can’t modify those locations

- Instead, the serial monitor redirects all attempts to load values into $FF80..$FFFF to a duplicate vector table located just below the monitor ($F780–$F7FF)
  - Then, when an interrupt occurs, the monitor jumps to the location that you loaded into the duplicate vector table

- All of this is transparent to you
  - Just go ahead and load your vector into the normal (non-monitor) vector locations ($FF80–$FFFF)
Example – time multiplexing 7-segment displays

- One port outputs the code for a 7-segment display
- It drives each character sequentially in time
- Another port is connected to cathodes; enables each display in turn
- Do it fast, user sees all displays illuminated simultaneously

Figure 4.18 Port B and Port P together drive six seven-segment displays (MC9S12DG256)
Strategy

• Main program
  – Set up ports B, P for output
  – Set up RTI rate
  – Initialize RTI vector
  – Enable RTI interrupts (RTIE)
  – Enable interrupts (CLI)
  – Loop forever

• RTIISR
  – Clear RTIF
  – Get character to be displayed
  – Get corresponding 7-segment pattern
  – Write pattern to Port B
  – Increment display count
  – RTI

• Assume we have six 7-segment displays
• Port B outputs the code for each display in turn
• Port P drives the cathodes of each display – it outputs a low to the bit corresponding to the display we want to light
Example in C

```c
int num=0;            // Number of character last displayed (0,1,2,3,4,5)

// These 6 hex digits will be displayed; each is 0..F.  Just initialize to some
// random values for now.
char dispDigits[] = { 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F };

char segm_ptrn[] = {  // 7-segment pattern of LEDs, for hex digits 0..F
    0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};
//  0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  A,  b,  C,  d,  E,  F

void main(void) {
    DDRB = 0xff;       // Set up PortB for output
    DDRP = 0xff;       // Set up PortP for output

    RTICTL = 0x13;     // period is 4096 ticks of 8Mhz clock, or about 2 ms
    CRGINT = 0x80;     // RTIE=1: enable rti interrupts

    EnableInterrupts;  // Turn on interrupt system (clears I bit)

    for(;;) {
        _FEED_COP();    /* feeds the dog */
    } /* loop forever */
}
```
Example in C

```c
void interrupt VectorNumber_Vrti rti_isr() {
    char d, c;

    CRGFLG = 0x80;        // acknowledge, clear RTIF flag

d = dispDigits[num];  // Get digit to be displayed (0..F)
c = segm_ptrn[d];     // Get corresponding 7-segment pattern

    PTP = 0xfe<<num;      // Output 0 at bit position “num” to turn on that display
    PTB = c;              // Output the 7-segment code

    num = (num+1)%6;      // Increment display number
}
```