EE475 Lab #4 Fall 2005

I/O Ports and Real-Time Interrupts

In this lab you will develop several more I/O routines: reading the toggle switch positions, setting up the real-time interrupts, and generating output signals on the Port T pins.

Preliminaries

1. Make a temporary local folder for your work:
   `c:\EEClasses\EE475\tempxxx` .

2. Launch CodeWarrior and create a new project using the New Project Wizard (see Lab #2 if you don't recall the procedures).

3. Replace the `main.c` file with your "blinking LED" program from last week (including the `XIRQ` interrupt handler function).

Exercise #1: Enable and use the IRQ interrupt

First verify that your "blinking LED" program still works, and that the XIRQ interrupt occurs when you press SW1 on the daughterboard. Recall that XIRQ is level sensitive, so it keeps getting triggered as long as you hold the button down.

→ Now modify your blink program so that it has another interrupt service routine installed to handle the IRQ interrupt in an *edge sensitive* mode rather than level sensitive.

To implement this program you will need to:

(a) Note that the IRQ signal is pin 2 of the daughterboard's edge connector. Run a jumper wire from pin 2 on the header adjacent to the daughterboard down to the "PB5" (push button 5) position on the USER I/O header of the SLK main board.

(b) Figure out which interrupt vector number corresponds to the IRQ signal, and install the interrupt handler function as you did for XIRQ last week.

(c) Write a test program to verify that the default IRQ mode is level sensitive. For example, have your IRQ interrupt handler function increment a variable and then observe the variable using the debugger.

(d) Determine the control bit in the 9S12 processor that makes IRQ *edge sensitive* rather than level sensitive, and put instructions in your program to enable this feature (refer to the Cady book or to the Freescale on-line documentation).

(e) Use the debugger to verify that your new IRQ interrupt is occurring in an edge-sensitive mode and that your XIRQ interrupt is still enabled and active.

Finally, modify your program so that the IRQ signal (push button 5) toggles the blinking: stop blinking on the first PB5 press, then start on the next press, etc.

→ Show the instructor your IRQ-controlled blinking LED program.

Exercise #2: Enable and use the real time interrupt

→ Now modify your program to install an interrupt service routine for the `Real Time Interrupt` vector and activate the HC12's Real Time Interrupt timer hardware. *Refer to the handout sheets and chip documentation to verify any details.*

To enable the real time hardware timer your program needs to:

(a) Set up the Real Time Interrupt divisor for roughly a 1 millisecond interval between interrupts (see the RTICTL scaling bits table).

(b) Reset the Real Time Interrupt flag bit in the Clock Flag Register (CRGFLG, 0x0037) by writing a 1 to bit 7 of that register.

(c) Set up the CRG (clocks and reset generator) Interrupt Enable Register (CRGINT, 0x0038) to enable real time interrupts (bit 7).

(d) In your interrupt service routine, arrange to toggle the state of bit 1 (CAREFUL: bit 1 is the *second* bit: bit 0 is the least significant bit) of Port T (PTT, 0x00AE). In other words, the output should change 0 to 1 during one interrupt, then 1 to 0 during the next interrupt, and so forth. *Don't forget to make bit 1 of Port T an output using the corresponding data direction register!* Bit 1 of Port T is connected to pin 15 of the daughterboard's edge connector. Observe pin 15 on the daughterboard connector with the oscilloscope.

(e) Arrange to set the RTI flag bit *inside* the interrupt service routine: you must set bit 7 (RTIF) of the flag register (CRGFLG, 0x0037) to 1 for your program to work properly because the flag is managed by the timer interrupt circuitry.

→ Start your program and observe the output waveform. Is the waveform period as expected? Record your measurement for your report. Verify that the LEDs can still be switched on and off with the IRQ signal while the waveform is generated.

→ As a final step for this exercise, *change* the Port T signal generated by your program from bit 1 (pin 15) to bit 0 (pin 13). Pin 13 on the daughterboard is connected to the buzzer on the SLK board, so you should hear the buzzer ringing when the program runs!

Exercise #3: Use the RT interrupt to control the LED flash rate

Modify your program once again: figure out a way to use the roughly 1 millisecond real time interrupt to make your LED flash occur as close to once per second as you can.

You might create a static variable that counts down the right number of "ticks" in the interrupt service routine and then sets a global flag variable to tell the main loop to toggle the LEDs. Another method would be to put the toggle code into the interrupt service routine itself. Since the real time interrupt is not exactly 1 ms, you will need to be careful about how you get a 1 second interval. Explain this in your report.

→ Demonstrate your precise 1 second LED flash program for the instructor.

Instructor Verification Sheet

Lab #4 Fall 2005

Student Name: _____

	Instructor Signature	**Date**
LED flash toggle with IRQ press (exercise 1)		
1 second LED flash (exercise 3)		

Lab Report

The lab report is to be written up in the Memo format. Be sure to put the *lab number* in the Memo header along with your name and date. For each exercise, answer the given questions and demonstrate your understanding of the exercise. Include **commented** file excerpts and this instructor verification sheet to get credit for the lab.

→ This lab report is due the beginning of the lab period in one week.