EE475 Lab #5 Fall 2005

Basic Multitasking

This lab introduces some simple non-preemptive multitasking concepts and issues. The way in which the processor is shared among multiple process threads, or *tasks*, is important when considering system performance, latency, and expandability.

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 own code. You may want to use portions of your previous lab exercises as a starting point.

Exercise #1: Simple Background Loop

For the first exercise this week you will be creating a simple background task loop.

Create a set of 8 independent functions named `task_0()`, `task_1()`, … `task_7()`. When run, the function must toggle the state of the LED corresponding to the task number, e.g., `task_4()` must toggle LED number 4 on the SLK main board while leaving all the other LEDs alone.

→ The first four LEDs (**LED1** – **LED4**) on the SLK main board are connected by default to **PAD04/AN04** – **PAD07/AN07**, respectively (no jumper wires required).For the other four LEDs carefully use jumper wires to connect **PM0** – **PM3** to LEDs **LED5** – **LED8** on the SLK main board.

RECALL that you will have to enable the proper port pins as outputs prior to driving the LEDs on and off, and also properly assert the SLK LED enable (Port T bit 4).

→ Write a `main()` program that tests your 8 task functions by running them one after the other in a loop. Put in a delay so that you can see each LED blink as expected.

Exercise #2: Running the Core Faster

The 9S12C32 processor on the CSM daughterboard uses a 16MHz crystal. The processor core (bus clock) runs normally at the crystal frequency divided by 2, or 8MHz. The processor core is able to function correctly at up to 24MHz by using the on-chip phase-lock loop (PLL) to generate a frequency multiplied clock. Refer to the PLL section of the 9S12 data sheet for more information. The following instructions can be used to program the PLL to make the bus clock 24MHz instead of 8MHz:

```
/*
 * Set up oscillator phase lock loop (PLL) for 24MHz bus speed.
 * (Procedure taken from HCS12 serial monitor code).
 */
 CLKSEL_PLLSEL  = 0;     // Disengage PLL
 PLLCTL_PLLON   = 1;     // Turn on PLL

 SYNR  = 0x02;  // Set loop multiplier to 3 (SYNR = 2, mult=SYNR + 1)
 REFDV = 0x00;     // Set input osc divider to 1 (REFDV=0, div=REFDV+1)

 _asm("nop"); _asm("nop");   // Delay for stabilization

 while( CRGFLG_LOCK != 1){}; // Wait until clock circuit locks on

 CLKSEL_PLLSEL  = 1;                            // Engage PLL and go!
```

→ Add these instructions to the start of your main() program and observe the effect on the LED flash rate. Be sure to explain this behavior in your lab report—including use of the PLL registers.

Exercise #3: Task Loop With Timing

Next, experiment with a way that you can use the `Real Time Interrupt` so that your `main()` program calls each task according to a specific schedule.

Your interrupt service routine and task loop must use a global `unsigned char` variable called `taskbits`. The task loop in `main()` must test one after the other each of the least significant 8 bits in the variable `taskbits`. If a bit is one, your main routine should clear it and call the corresponding function `task_n()`, and so forth, like the following pseudocode:

```
loop forever:
        for $0 \le n \le 7$ :
                if bit $n$ in taskbits is '1':  set that bit to zero and call task_$n$();
                 else continue
        end for $n$
end forever
```

Your interrupt routine must keep track of the call schedule for each task, according to the schedule table given below. When the appropriate number of interrupt ticks have occurred, your ISR must set the proper bits in `taskbits` so that your background task loop in `main()` will trigger the proper tasks.

Task Name	**Call every:**
`task_0()`	250 ms
`task_1()`	500 ms
`task_2()`	1 s
`task_3()`	2 s
`task_4()`	3 s
`task_5()`	4 s
`task_6()`	16 s
`task_7()`	32 s

NOTE that you will not be able to obtain the precise durations due to the coarseness of the available `Real Time Interrupt` intervals. Choose the slowest `RTI` frequency that will still provide better than 1 ms accuracy for each time interval, and include your precision calculations in your memo report.

→ Also, try your program both with and without the PLL timing change. Does the Real Time Interrupt rate change according to the bus clock?

Exercise #4: Task Loop With Timing and Disabling

→ Finally, add an IRQ interrupt service routine and a jumper wire from one of the SLK pushbuttons to the IRQ* (pin 2) on the daughterboard header. Create another global variable so that each time you press the button your ISR will cause `task_`*`n`*`()` not to be called by preventing its enable bit from ever getting set in the `taskbits` variable. Start with $n = 0$ and then increment n after each button press. In other words, after 8 presses none of the tasks should be called. With the 9th and subsequent presses your program should re-enable the 8 tasks one at a time.

Also, be sure that `IRQE` is set for edge-triggered operation.

As you implement your program, consider if you might be able to use a *task table* consisting of an array of pointers to each function. This could reduce the size and complexity of your program, and make it easier to expand. We'll work on this more in future labs.

→ Show the instructor your time-controlled AND IRQ-controlled LEDs in operation.

Lab #5 Fall 2005

Student Name: ______________________

	Instructor Signature	**Date**
Ex #4 Time and switch controlled tasks		

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 a week from today by 5PM.