EE475 Lab #5 Fall 2004

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 program from Lab #4 or some other previous lab example.

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 while leaving all the other LEDs alone (PORTP).

→ 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: Task Loop With Timing

Next, determine a way that you can use the `UserRTI` interrupts so that your `main()` program calls each task according to a specific schedule.

Your interrupt service routine and task loop must use a global `int` 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 ≤ n ≤ 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()`	4 s
`task_5()`	8 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 `UserRTI` intervals. Choose the slowest `UserRTI` frequency that will still provide better than 1 ms accuracy for each time interval, and include your precision calculations in your memo report.

→ Show the instructor your time-controlled LEDs.

Exercise #3: Task Loop With Timing and Enabling

Finally, add a UserIRQ interrupt service routine so that each time you press the IRQ button your ISR will read the position of the toggle switches and save the information in a global variable so that the `main()` task loop will only call the `task_n()` if the corresponding toggle switch was 'on' when the IRQ button was pressed. NOTE that the toggle switch positions are only to be checked when IRQ is pressed. 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 next week.

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

RTICTL (Real-Time Interrupt Control Register located at 0x0014)

(You will need to set bit7 and bits2:0)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RTIE	RSWAI	RSBCK	0	RTBYP	RTR2	RTR1	RTR0

RTIE – Real-Time Interrupt Enable
0 = Interrupt requests from RTI are disabled (default)
1 = Enable RTI interrupts

RSWAI = 0; RSBCK = 0; RTBYP = 0;

RTR2:RTR0 – Real-time Interrupt Rate Select

RTR2	RTR1	RTR0	Divide M by:	Interrupt Time: (M=8.0 MHz)
0	0	0	Off	Off
0	0	1	2^{13}	1.024 ms
0	1	0	2^{14}	2.048 ms
0	1	1	2^{15}	4.096 ms
1	0	0	2^{16}	8.192 ms
1	0	1	2^{17}	16.384 ms
1	1	0	2^{18}	32.768 ms
1	1	1	2^{19}	65.536 ms

RTIFLG, Real-Time Interrupt Register (located at 0x0015)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RTIF	0	0	0	0	0	0	0

NOTE: you must set RTIF=1 inside the interrupt service routine for your program to work properly because the flag is changed by the interrupt circuitry.

PORTP (Port P located at 0x0056)

You will write data to this port to turn the LEDs on and off. Note that LEDs are connected active low.

PORTT (Port T located at 0x00AE)

You will write data to bit 0 (I_OC0) of this port in order to change the output from '0' to '1' in a periodic fashion (when each interrupt occurs).

PORTAD (Port AD located at 0x006F)

This port is connected to the toggle switches via a tri-state buffer. If the buffer is enabled, the switch positions can be determined by reading this port. The switches read active low.

PORTCAN (Port located at 0x013E)

Set bit 6 (PCAN6)to '0' and bit 5 (PCAN5) to '1' in this register to enable the tri-state buffers for the LEDs and switches.

DDRP (Port P data direction register located at 0x0057)

You need to set each bit of this register to determine the direction of the data on the corresponding pin. To make a bit in port P an output bit, you set the bit in DDRP to '1'. In this case turn them ALL into outputs.

DDRT (Port T data direction register located at 0x00AF)

To make a bit in port T an output bit, you set the corresponding bit in DDRT to '1'.

DDRCAN (data direction register located at 0x013F)

Set bits 5 and 6 (DDRCAN5, DDRCAN6) of this register to '1' so that the corresponding bits in PORTCAN are outputs.

Instructor Verification Sheet
Lab #5 Fall 2004

Student Name: _____

	Instructor Signature	Date
Ex #2 Time-controlled tasks		
Ex #3 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 the beginning of the lab period in one week.