EE475 Lab #7 Fall 2005

Investigating Task Priority

This week in Lab #7 you will use the prior framework of Lab #6 to evaluate the effects of task priority in the simple non-preemptive multitask system.

Preliminaries

1. Start with the code you developed for Lab #6.
2. Make a temporary local folder for your work: `c:\EEClasses\EE475\tempxxx` .
3. Launch Code Warrior and make a new project file for this week.
4. Replace the `main.c` file with your program from Lab #6. Run the code to make sure it still functions as you expect.

Recall from last week that there is an 8-bit variable called `interrupt_pattern`:

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
unused	unused	unused	unused	Enable F3	IRQ	RTI 1	RTI 0

a) `RTI_handler()` sets bits _0_ and _1_ of `interrupt_pattern` on a periodic basis.

b) `IRQ_handler()`sets bit _2_ of `interrupt_pattern` whenever the IRQ button is pressed.

c) `func0()`flips the state of pin 1 in `PORTT`.
`func1()`flips the state of pin 2 in `PORTT`.
`func2()`flips the state of LED 1 (CSM)
and sets bit _3_ of `interrupt_pattern` *on every other call.*
`func3()` flips the state of LED 2 (CSM).
`idle()`increments global variable `idle_count` to track how many times it runs.

d) Finally, recall that the task loop in `main()` requires the DIP switches to be set properly and that the lower task numbers be satisfied before allowing the higher task numbers to run (simple priority scheme that always checks task 0 first).

Exercise #1: More complicated tasks

So far we have been using some very short and trivial tasks. In this exercise you will simulate what happens when a more time-consuming task or set of tasks is to be used.

Modify the code for `func0()` so that in addition to modifying pin 1 of `PORTT` it also reads each address from $00 through $FF into a `static char` array and calculates the sum of the 256 characters.

Modify the code for `func1()` so that in addition to modifying pin 2 of `PORTT` it also reads each address from $100 through $1FF into a `static char` array and calculates the sum of the 256 characters.

Finally, modify the code for `func3()` so that in addition flipping LED 1 to it also reads each address from $200 through $2FF into a `static char` array and calculates the sum of the 256 characters.

This additional processor activity will increase the amount of time spent in `func0()`, `func1()`, and `func3()`. <u>If the total takes so long that the system completely malfunctions, reduce the delay modifications to a smaller amount than 256</u>.

→ Experiment with the following:

(a) Figure out a way to compare how quickly `idle_count` increases when `func0()` is enabled and disabled. Does the debugger provide any means to do this accurately?
(b) If all the functions are enabled, does the increased computation cause any noticeable change in the system behavior (e.g., `PORTT` waveforms)?
(c) Is there any noticeable effect on the `PORTT` waveforms when the IRQ button is pressed? Do the LEDs illuminate as expected?

Exercise #2: Priority changes

Edit your `main()` program to alter the task priorities: load `func_table[]` so that `func3` is in [0], `func2` is in [1], `func0` is in [2], and `func1` is in [3]. This makes `func3` the highest priority and `func1` the lowest priority. <u>You will also need to switch the bits in `interrupt_pattern` to match the new ordering! Find the places that `interrupt_pattern` is set and tested and make sure the modifications do what you need to do.</u>

→ Experiment with the following:

(a) Compare how quickly `idle_count` increases when `func0()` is enabled and disabled. How do the results compare to what you found in Exercise #1?
(b) Is there any noticeable effect on the `PORTT` waveforms when the IRQ button is pressed?
(c) Predict what would happen if `func3()` was altered to take twice as much time to execute. Then try it and see!
(d) Finally, exchange `func1` and the altered `func3` in the task table, making `func1` the highest priority. Observe the system behavior in this condition and comment on the results.

→ ***Demonstrate and explain the altered priority exercise for the instructor.***

Instructor Verification Sheet
Lab #7   Fall 2005

Student Name: ______________________

	Instructor Signature	**Date**
Ex. #2 Task loading and priority demonstration		

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 will be due at the start of lab on Tuesday, November 8, 2005 (instructor out of town next week).