

**EE 371 LAB 8 - F05**  
**Interrupts I**

Name \_\_\_\_\_

Partner \_\_\_\_\_

Demo (1) \_\_\_\_\_

*There is always one more imbecile than you counted on.*

Schedule:     Nov 2,3: Do the lab  
              Nov 9,10: Last time to demo lab.

1. You are to design and implement a program that has a foreground job and a background job that runs under interrupt control.

Foreground Task:

The foreground task is to be a simple typing program. Using getchar and putchar from the serial I/O routines, you should be able to continuously type characters on the keyboard and display them on the screen. Continue doing this forever. The only exception to simply echoing the character returned by getchar is that if a carriage return or a line feed character is received, your program should output a carriage return/line feed pair. This allows the user to hit the enter key and return to the next line for typing. If the user enters more than 80 characters hyperterm will automatically wrap the line so don't worry about that.

Background Task:

The background task is to be actuated by pressing the push button 1 (PB1) on the SLK board. It is connected to Port P, bit 5. When the switch is pressed, an interrupt service routine is to be entered and a 4-bit counter, displayed on LED1-LED4 is to be incremented. The counter should start at 0, step up to 15 and then wrap around.

Overall Program Characteristics:

- a. The counter is to increment once and only once when you push the switch.
- b. The push button switch bounces so you will have to debounce it in the interrupt service routine.
- c. The foreground job should continue to run if the user holds the pushbutton switch depressed.
- d. The switch bounce sometimes causes the counter to increment when you release the switch. This is unacceptable and you must eliminate any of these.

Helping Hand #1:

A handout describing the operation of Port P will accompany this lab assignment. It is also available on the EE371 lab web page.

Helping Hand #2:

Solving the switch bounce problem should be fairly easy if you use an integrating debouncer that returns the final state of the switch at the end of the bounce. For example, it could return a 0 for the switch in one state and 1 for the other. Your interrupt service routine would then increment the counter based on the final state of the bouncy switch.

Helping Hand #3:

It is perfectly OK to execute a subroutine (jsr sub) in an ISR. Think about making your debounce algorithm a subroutine that you can call whenever you need it.

2. Grading: 10 points demo

