CpE213 – Digital Systems Design

Homework 10
Interrupts

1. Write a short program in C that has 2 interrupts. The first interrupt comes in on P3^3 and consists of a series of pulses from some external pulse generator (like a sensor used to calculate RPMs. Note that the example handed out in class comes in on a different port pin). Every time the interrupt is activated, the interrupt service routine simply adds one to the current "count". Make count an int. This interrupt should have a high priority. The second interrupt comes in on P3^2. Every time the interrupt is activated, this interrupt service routine clears the count of pulses (i.e. count = 0). This interrupt should have a low priority. Both interrupts should initially be set to trigger on the falling edge of the input. Your main program doesn't have to do anything (i.e. while(1){;}). You may test your program simply by setting the port pins to 0 and 1 (i.e. creating an edge or low level on these pins). Answer the following questions:
    a. Which interrupt runs first when both P3^2 and P3^3 are set/cleared simultaneously? Does the second interrupt run immediately after?
    b. When the external interrupt runs, is the interrupt flag (IE0 and IE1) cleared automatically?
    c. Simulate your code such that the first interrupt has just started when the second one comes in (i.e. there is an edge on P3^2). Does the second interrupt "interrupt" the first? Why/why not?
    d. Simulate your code such that the second interrupt has just started when the first one comes in (i.e. there is an edge on P3^3). Does the first interrupt "interrupt" the second? Why/why not?
    e. Timing is often critical with interrupts. Watching the machine-cycle counter in uVision, measure exactly how many machine cycles pass between when the edge appears on one of the ports and when the first C-instruction of the interrupts begins to execute (it won't be 0!). How long was this delay? What's the reason for this delay?
    f. Look at the disassembled version of your C-Code.
        i. What ASM instruction is located at C:0x0000? Why?
        ii. At what location does the first interrupt begin? The second interrupt? Why?
        iii. Are the interrupts both written in one contiguous block of code, or does the processor have to "jump around" from one location to another to execute the interrupt? Why do you think the assembler did it this way?
        iv. How do the interrupts end? With a RET or a RETI?
    g. Re-write your code so that the first interrupt (the pulses) is level triggered rather than edge triggered. Re-simulate your code. What happens when the first interrupt comes in this time? Why?

    Along with the above questions, please turn in a copy of your code.

2. The following interrupts have the following priorities and "come in" (hardware sets the interrupt flag) at the following times. Draw a timing diagram like we did in class that shows which routine (main, timer 1, serial, ex0) is executed an in what order. Assume each of the interrupt service routines (timer 1, serial, ex0) take 30 uS to execute. Each time an interrupt occurs, note on your diagram if the interrupt was started because of polling or because of priority or because it was the only one.

Interrupt	Priority	Interrupt occurs at…			
timer 1	high	100uS	210uS	310uS	400uS
serial	low	100uS	200uS		410uS
Ex0	high	100uS	220uS	300uS	