RCET 3375 Experiment 6

Interrupt Service Routines

Goals: The student will be able to:

Write and troubleshoot interrupt driven programs.

Write a priority interrupt program.

Background Information:

One way of determining if a peripheral device is requesting service is by *polling* it. If the device can provide a "service request" or "ready" signal, it could be connected to an input port. The CPU could then be instructed to check each bit on that input port. If any are true the CPU would call a corresponding subroutine which would instruct the CPU in what to do to service the requesting device, and then return to the main program. It would then check other port inputs for additional service requests. The main program could be written to either give priority to a certain peripheral device or share time equally with all devices.

One disadvantage of the polling technique is that while polling, the CPU is continually asking "do you need service", to which the reply is usually no. So, the CPU is wasting time asking when no service is needed. Interrupt methods of handling peripherals do not have this problem.

Interrupts are a mechanism by which the CPU is notified of an important event outside of the CPU's normal process. This will cause the CPU to call a special subroutine called an Interrupt Service Routine (ISR). The ISR, when executed, will instruct the CPU in what to do to satisfy that particular interrupt. For example, let's assume that a PIC controlled system is controlling your home, and an intruder opens a window. The intrusion signal would be wired to one of the CPU's interrupt inputs, and would signal the CPU to break from its normal control program, and branch to the intrusion ISR. The ISR would instruct the CPU to ring the alarms, turn on the lights, and call the police. After executing the ISR the CPU would return to the normal control program.

Tasks: *(You only need to Include the code and flow chart for final program in your lab notebook)*

1. Write a program that will display a 1 in the dot matrix display then repeat.

2. Add an interrupt service routine and any other necessary instructions that will display 7 for 2 seconds any time the PORTB,0 button is pressed, then return from

the ISR. The display should show either 1 or 7 only.

3. Add another interrupt service routine and any other necessary instructions that will display 6 for 2 seconds any time the PORTB,1 switch is pressed.

4. Now, your main program should display 1 all of the time until either the 6 or 7 interrupt inputs are pressed true, at which time 6 or 7 respectively should be displayed for about 2 seconds. You may not poll 6 or 7 in main code. They must be initiated by an interrupt.

5. Modify your program so that when in the 6 ISR, the CPU still responds to a 7 interrupt. When the 7 interrupt is complete, the PIC will complete the 6 ISR time. The display must always indicate the specific ISR routine or main code that is being executed. The 6 ISR must be able to be interrupted by the 7 ISR multiple times, however the 6 ISR cannot interrupt the 7 ISR

6. Instructor Checkpoint: The instructor will test the operation of your final program as well as your flow chart. (Hint, he will try to overflow your stack.)

7. Complete all bitmaps, schematics and calculation required for this experiment.