Lab 3: Bargraph Intensity

Reference

Ch. 11 of 16F87x data sheet, Analog-to-Digital Converter Module, Ch. 14.11 Interrupts, Ch. 5 Timer0.

Project Description

For this project you will modify Lab 2 to change the intensity of the LEDs based on the output of the second potentiometer.

Analog-to-Digital Converter

On the board, the top pot, designated RP1 is connected to RA1/AN1 [the 1 position in Port A and in the AD converter]. The second potentiometer, designated RP2, is connected to the RA0/AN0 input on the PIC. In this lab, RP1 will control the bar chart, while RP2 will control the intensity of the LEDs by using software pulse width modulation of the on time of the LEDs.

For this lab, download bar_intensity.c from the website. This code is basically the solution for Lab 2. Change the ADConvert() subroutine so that it also converts the value read from the RP1 pot as well as the RP0 pot. Save the RP1 result in a variable called LEDIntensity. Remember, that the analog port that you are converting is controlled by the CHS0 bit. Note, also, that you will need to delay for A/D acquisition and wait for ADGO for both RP0 and RP1. Remove the line that sets PORTD to LEDValue. We will set that in the interrupt service routine.

Timer0 Interrupts

Change the initialization of the INTCON register to include the setting of the T0IE bit [timer zero interrupt enable bit]. Note that setting the GIE bit [global interrupt enable] should always be the LAST thing you do in Initial(). Also, we want to use 32:1 prescaling for TIMER0, so you will also need to add the line “OPTION = 0b00000100;” to the Initial() routine. This will mean that TIMER0 will take 4.446 ms to count through its 256 states.

Intensity Interrupt Handler

An interrupt handler is a subroutine that responds to an interrupt. It is always identified with the interrupt keyword and there should be only one interrupt handler per program. The first thing you should do in the interrupt handler is check if the Timer0 is causing the interrupt by checking the TMR0IF flag. If so, you should clear the TMR0IF flag before continuing on. The timer interrupts every time the TMR0 register overflows, and we want it to interrupt twice every
4.446 ms – once to set the LEDs to LEDValue and a second time to turn off the LEDs. The timing of the interrupts depends on the value of LEDIntensity. The proportion of time that we leave the LEDs on is directly related to the value of LEDIntensity. So, if LEDIntensity is equal to X, the LEDs should be on X/256 of the time. We can use this information to set the TMR0 register. If we want the LEDs to be on LEDIntensity counts out of 256, we can set the TMR0 register to 256-LEDIntensity. After LEDIntensity counts, the interrupt will trigger and then we can clear the LEDs and then set TMR0 to LEDIntensity. After 256-LEDIntensity counts, the interrupt will trigger again and this time we can set the LEDs to LEDValue and set TMR0 back to 256-LEDIntensity. In the interrupt service routine you can check if PORTD is 0 and that will let you know whether you should set TMR0 to LEDIntensity or to 256-LEDIntensity.

**Measurements**

**Measuring Duty Cycle**
When setting PORTD to LEDValue, set bit 4 of PORTA with RA4=1
After clearing PORTD, clear bit 4 of PORTA with RA4=0

You can then get a waveform of the duty cycle on a scope by connecting a scope probe to Port RA4 located on J5 pin 7. Ask the TA or instructor for assistance.

**Assignment**
Demonstrate your project and show the waveform in class and hand in your well-commented C code by Tuesday of next week.