Timers With Interrupts Practice

1. We will make a new project for this mini-lecture:

   - Go to the "calendar" page of the ME430 website and download "timers with interrupts.c" to a new folder.
   - It uses the LCD so also get "LCD module.c" and "LCD module.h" from the "courseware" page and add those to the folder.

   Make a new project, build it, program the target device, and run the code. It should display the value of Timer0 as it counts up, and also count how many times the timer has overflowed by lighting up the LEDs.

2. Study the code.

3. Try the following:

   a. Add the command

      ```
      WriteTimer0(30000);
      ```

      inside the `if` block in the high interrupt service routine (high_isr) . What happens when we run the code now?

   b. Without changing the clock frequency, change the `WriteTimer0` command in the isr so that an interrupt happens every second. What number do we need to put in the command? _____

   c. Try shutting off the interrupts three ways:
      i. `OpenTimer0(TIMER_INT_OFF & …)`
      ii. Comment out the `INTCONbits.GIE=1;` line.
      iii. Uncomment the `INTCONbits` line, and change it to `INTCONbits.GIE = 0;`

      When the interrupts are off, what changes—i.e. what doesn't happen any more?_____

      Turn the interrupts back on before you go to the next step.

   d. Set up Timer1 (similar to the quiz) and have it display on the second line of the LCD. (Timer0 is still displaying on the first line.) Use PS 1:8, 16 bits, interrupts off. Make sure that this works!

e. Now turn on interrupts for Timer1, and have it toggle (change the state of) RB3. In order to turn on the interrupts, be sure to change the OpenTimer1 statement and also enable peripheral interrupts. The code for toggling RB3 will get added to the high_isr routine. However, notice that the Timer1 interrupt flag is ***not*** INTCONbits.TMR1F. You can go to the "courseware" page of the ME430 website and look at the "Where is that interrupt bit?" pdf to figure out the name of the flag. See the pretty lights!

When this is done you can go back to working on labs ☺

```
/********************************************************************
* FileName:        timers with interrupts.c
* Processor:       PIC18F4520
* Compiler:        MPLAB C18 v.3.06
*
* This file displays the timer 0 on the PICDEM 2 LCD screen.
*
*  As a bonus it increments the PORTB LEDs using an interrupt when
*   the timer overflows.
*
* Author               Date        Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
// David Fisher                10/3/07


/** Processor Header Files *********************************************/
#include <p18f4520.h>
#include "LCD module.h"
#include <stdio.h>
#include <timers.h>

/** Define Constants Here **********************************************/
#define SAMPLE 100

/** Local Function Prototypes ******************************************/
void low_isr(void);
void high_isr(void);

// ==========================================================
// Configuration Bits
// For details on PIC18F configuration bit settings, see
// PIC18 Configuration Settings in MPLAB-IDE Help
// ==========================================================
#pragma config OSC = EC  // External 4MHz crystal
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOREN = OFF

/** Declare Interrupt Vector Sections **********************************/
#pragma code high_vector=0x08
void interrupt_at_high_vector(void)
{
   _asm goto high_isr _endasm
}

#pragma code low_vector=0x18
void interrupt_at_low_vector(void)
{
   _asm goto low_isr _endasm
}

/** Global Variables ***************************************************/
        char line1[17];
        char line2[17];
        unsigned int result0, result1;

/*****************************************************************
* Function:        void main(void)
*****************************************************************/
#pragma code

void main (void)
{
                // Setup Output LEDs
                ADCON1 = 0x0F;
                TRISB = 0xF0;
                PORTB = 0x00;

                // Initialize the LCD
                XLCDInit();
                XLCDClear();


                //A configure timer0
                OpenTimer0( TIMER_INT_ON & T0_16BIT & T0_SOURCE_INT & T0_PS_1_128 );
                // Clock Frequency = 4 MHz
                // Instruction cycle Frequency = 1 MHz
                // Timer 0 Frequency = 1 MHz / 128 = 7812.5 Hz
                // Timer 0 Period =  128 uSeconds <-- Always equals prescaler when clock is 4 MHz
                // Overflow every 128 uSecond * 2^16 ticks = 8.4 seconds
```

```
                // This code is only necessary for the interrupt

                // Enable the Timer zero interrupt (set by library function)
//              INTCONbits.TMR0IE = 1;

                // Enable Peripheral interrupts (to use Timer 1 as interrupt)
//              INTCONbits.PEIE = 1;

                // Enable Global Interrupts
                INTCONbits.GIE = 1;


                while( 1 )
                {
                        result0 = ReadTimer0(); // read timer
                        sprintf(line1,"Timer 0 -> %#5.5u",result0);
                        XLCDL1home();
                        XLCDPutRamString(line1);
                }
}


/*****************************************************************
* Function:        void high_isr(void)
* Input:                Occurs on a timer overflow
* Output:               Modify the PORTB LEDs
* Overview:             The interrupt counts overflows using LEDs
*****************************************************************/
#pragma interrupt high_isr                      // declare function as high priority isr
void high_isr(void)
{
        if(INTCONbits.TMR0IF)  // Check wether it was the timer 0 interrupt
        {
                INTCONbits.TMR0IF = 0;  // Clear interrupt flag for TIMER Zero
                PORTB++;
        }
}

/*****************************************************************
* Function:        void low_isr(void)
* Input:
* Output:
* Overview:
*****************************************************************/
#pragma interruptlow low_isr            // declare function as low priority isr
void low_isr(void)
{
}
```