```
/*****************************************************************************************/
/* BME 361 Biomeasurement Laboratory:  Base Code                                         */
/* Instructors: Prof. Ying Sun, Prof. John DiCecco                                       */
/* Last update: 20 October, 2018                                                         */
/*****************************************************************************************/

/******************************** Specify the chip that we are using *********************/
#include <xc.h>

/******************************** Configure the Microcontroller **************************/
#pragma config OSC = XT
#pragma config WDT = OFF
#pragma config PWRT = OFF
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config BOREN = ON
#pragma config BORV = 2
#pragma config WDTPS = 128
#pragma config PBADEN = OFF
#pragma config DEBUG = OFF
#pragma config LVP = OFF
#pragma config STVREN = OFF
#define _XTAL_FREQ 4000000

/********************************** Define Prototype Functions ***************************/
unsigned char ReadADC();
void interrupt isr(void);
void SetupADC(unsigned char channel);
void delay_ms(unsigned int x);

/************************************** Global variables *********************************/
unsigned char data0, data1, output;
unsigned int dummy;
unsigned char ad_input, mode, counter, output;
unsigned char CountLED;

unsigned char ReadADC(){   /*************** start A/D, read from an A/D channel ******************/
unsigned char ADC_VALUE;
    ADCON0bits.GO = 1;                // Start the AD conversion
    while(!PIR1bits.ADIF) continue;   // Wait until AD conversion is complete
    ADC_VALUE = ADRESH;               // Return the highest 8 bits of the 10-bit AD conversion
    return ADC_VALUE;
}

void interrupt isr(void){   /************** High Priority Interrupt Service Routine ***************/
    if(INTCONbits.TMR0IF == 1) {      // When there is a timer0 overflow, this loop runs
        INTCONbits.TMR0IE = 0;    // Disable interrupt
        INTCONbits.TMR0IF = 0;    // Reset timer 0 interrupt flag to 0
        TMR0H  =0xF6;             // Reset timer count: high-order and low-order bytes
        TMR0L = 0x4F;             // $FFFF - $F64F = $09B0 = 2480 (decimal) => ~ 2.5ms
        PORTBbits.RB2 = !PORTBbits.RB2; // toggle RB2 to generate a 200 Hz clock
        data1 = data0;                  // Save the previous sample in data1
        data0 = ReadADC();              // Read A/D and save the present sample in data0
        output = data0;                 // Send data right back without processing
        PORTD = output;                 // Output to the D/A via the parallel port D
        INTCONbits.TMR0IE = 1;          // Enable Interrupt
    }
}
```

```
void SetupADC(unsigned char channel){   /*********** Configure A/D and Set the Channel ************/
    TRISA = 0b11111111;             // Set all of Port A as input
    // ADCON2 Setup
    // bit 7: Left justify result of AD (Lowest 6bits of ADRESL are 0's) (0)
    // bit 6: Unimplemented
    // bit 5-3: AD aquisition time set to 2 TAD (001)
    // bit 2-0: Conversion clock set to Fosc/8 (001)
    ADCON2 = 0b00001001;
    // ADCON1 Setup
    // bit 7-6: Unimplemented
    // bit 5: Vref - (VSS) (0)
    // bit 4: Vref + (VDD) (0)
    // bit 3-0: Configuration of AD ports (Set A0-A4, others to digital, including the PORTB)
    ADCON1  = 0b00001010;
    // ADCON0 Setup
    // bit 7,6 = Unimplemented
    // bits 5-2 = Channel select
    // bit 1: GO Bit (Starts Conversion when = 1)
    // bit 0: AD Power On
    ADCON0 = (channel << 2) + 0b00000001;
    PIE1bits.ADIE = 0;              // Turn off the AD interrupt
    PIR1bits.ADIF = 0;              // Reset the AD interrupt flag
}

void delay_ms(unsigned int x){   /******** Generate a delay for x ms, assuming 4 MHz clock ********/
unsigned char y;
    for(;x > 0; x--) for(y=0; y< 82;y++);
}

/************************************** MAIN PROGRAM ***************************************/
void main(){
    output = data0 = data1 = 0;     // Initial Global Variables
    CountLED = 0;
    TRISD = 0b00000000;             // Set all Port D pins as outputs (connected to D/A)
    TRISB = 0b00000000;             // Set all Port B pins as outputs
    PORTB = 0x00;
    //SetupADC(0);                  // Call SetupADC() to set up Channel 0, AN0 (pin 2)
    T0CON = 0b10001001;             // Setup the timer control register
    // Setup timer interrupt support
    // bit 7 = GIE - Global Interrupt Enable
    // bit 5 = TMR0IE - Timer 0 Overflow Interrupt Enable
    // bit 2 = TMR0IF - Timer 0 Interrupt Flag
    INTCON = 0b10000000;            // Global GIE set (bit 7), TMR0 interrupt enable (bit 5)
    while(1) {                      // forever loop
        PORTBbits.RB4 = !PORTBbits.RB4 ;   // toggle RB4 to turn LED on and off
        delay_ms(500);                     // LED toggles every 1/2 second
    }
}     /************************************ END MAIN ***************************************/
```