This code sets up to the LCD character display and facilitates a push button to scroll through the different dunctions (binary counter, ECG simulation, echo, and derivative).

. . . . . .

```
/******************************** Define Prototype Functions *********************************/
unsigned char ReadADC();
void Delay_ms(unsigned int x);
void Transmit(unsigned char value);
void PrintNum(unsigned char value1, unsigned char position1);
void SetupSerial();
void SetupADC(unsigned char channel);
void ClearScreen();
void Backlight(unsigned char state);
void SetPosition(unsigned char position);
void PrintLine(const unsigned char *string, unsigned char numChars);
void interrupt isr(void);

/************************************ Global variables ***************************************/
unsigned char output, count, mode, function, data0, data1;
unsigned char update, LEDcount, LEDcount1, debounce1;
int dumb;

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 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++);
}

void Transmit(unsigned char value) {  /********** send an ASCII Character to USART ***********/
    while(!PIR1bits.TXIF) continue;     // Wait until USART is ready
    TXREG = value;                      // Send the data
    while (!PIR1bits.TXIF) continue;    // Wait until USART is ready
    Delay_ms(5); // Give the LCD some time to listen to what we've got to say.
}

void PrintNum(unsigned char value1, unsigned char position1){ /** Print number at position ***/
    int units, tens, hundreds, thousands;
    SetPosition(position1);         // Set at the present position
    hundreds = value1 / 100;        // Get the hundreds digit, convert to ASCII and send
    if (hundreds != 0) Transmit(hundreds + 48);
    else Transmit(20);
    value1 = value1 - hundreds * 100;
    tens = value1 / 10;             // Get the tens digit, convert to ASCII and send
    Transmit(tens + 48);
    units = value1 - tens * 10;
    Transmit(units + 48);           // Convert to ASCII and send
}

void SetupSerial(){  /************ Set up the USART Asynchronous Transmit (pin 25) ************/
    // For LCD - use SPBRG = 25;  9600 BAUD at 4MHz: 4,000,000/(16x9600) - 1 = 25.04
    // For Bluetooth - use SPBRG = 1; 115200 BAUD at 4MHz: 4,000,000/(16x115200) - 1 = 1
    TRISC = 0x80;                   // Transmit and receive, 0xC0 if transmit only
    SPBRG = 25;
    TXSTAbits.TXEN = 1;             // Transmit enable
    TXSTAbits.SYNC = 0;             // Asynchronous mode
    RCSTAbits.CREN = 1;             // Continuous receive (receiver enabled)
    RCSTAbits.SPEN = 1;             // Serial Port Enable
```

```
    TXSTAbits.BRGH = 1;             // High speed baud rate
}

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 ClearScreen(){   /************************** Clear LCD Screen ****************************/
    Transmit(254);                  // See datasheets for Serial LCD and HD44780
    Transmit(0x01);                 // Available on our course webpage
}

void Backlight(unsigned char state){  /************** Turn LCD Backlight on/off ***************/
    Transmit(124);
    if (state) Transmit(0x9D);      // If state == 1, backlight on
    else Transmit(0x81);            // otherwise, backlight off
}

void SetPosition(unsigned char position){   /********** Set LCD Cursor Position  *************/
    Transmit(254);
    Transmit(128 + position);
}

void PrintLine(const unsigned char *string, unsigned char numChars){ /*** Print characters ***/
    unsigned char count1;
    for (count1=0; count1<numChars; count1++) Transmit(string[count1]);
}

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 TMR0 interrup
        INTCONbits.TMR0IF = 0;      // Reset timer 0 interrupt flag to 0
        if (debounce1) debounce1--; // switch debounce delay counter for INT1
        TMR0H = 0xF0;               // Reload TMR0 for 4.167 ms count, Sampling rate = 240 Hz
        TMR0L = 0x7C;               // 0xFFFF-0xEFB8 = 0x1047 = 4167, adjust for delay by 68 us
        switch (function) {
        case 1:         // ECG simulation
            TMR0H = 0xFC;           // Reload Timer 0 for 1 ms count
            TMR0L = 0x17;           // 0xFFFF-0xFC17 = 0x3E8 = 1000
            switch (mode) {         // ECG Simulation from Lab #2
            case 0:     // P wave up
                count++;
                output++;
                if (count == 30) mode++;
                break;
            case 1:     // P wave flat
```

```
            . . . . . .
            break;
            . . . . . .
        case 13:
            . . . . . .
            break;
        }
        break;
    case 2:             // Echo
        data0 = ReadADC();      // Read A/D and save the present sample in data0
        output = data0;
        break;
    case 3:                     // Derivative
        data1 = data0;          // Save the previous sample in data1
        data0 = ReadADC();                  // Read A/D, save the present sample in data0
        dumb = (int)data0 - data1 + 128;    // Take derivative and shift to middle
        output = (unsigned char)dumb;
        break;
    }
    PORTD = output;             // Output to the D/A via the parallel port D
    INTCONbits.TMR0IE = 1;      // Enable timer TMR0 interrupt
}
if (INTCON3bits.INT1IF == 1) {  // Pushing the button at pin 34 activates this interrupt
    INTCON3bits.INT1IE = 0;     // Disable interrupt
    INTCON3bits.INT1IF = 0;     // Reset interrupt flag
    if (debounce1 == 0) {       // If debounce1 is not 0, it's a switch bounce
        if (function >= 3) function = 0;    // Set function range 0-3
        else function++;
        if (function == 1) {    // mode 1: ECG simulation
            count = mode = 0;   // Initialize for ECG simulation
            output = 50;
        }
        update = 1;             // Signal the main program to update LCD display
        debounce1 = 10;         // Delay by 41.67 ms (10 x 4.167) to debounce switch
    }
    INTCON3bits.INT1IE = 1;     // Enable INT1 interrupt
}
}

void main(){   /****************************** Main program *********************************/
    mode = function = count = debounce1 = LEDcount = 0; // Initialize
    output = 50;
    update = 1;
    TRISD = 0b00000000;     // Set all port D pins as outputs (connected to D/A)
    TRISB = 0b00000010;     // RB1 as input for pushbutton, others outputs
    SetupADC(0);            // Call SetupADC() to set up channel 0, AN0 (pin 2)
    SetupSerial();          // Set up USART Asynchronous Transmit for LCD display
    Delay_ms(3000);         // Wait until the LCD display is ready
    Backlight(1);           // turn LCD display backlight on
    ClearScreen();          // Clear screen and set cursor to first position
    PrintLine((const unsigned char*)"  BME 361 Lab", 13);
    SetPosition(64);        // Go to beginning of Line 2;
    PrintLine((const unsigned char*)" Biomeasurement",15);  // Put your trademark here
    Delay_ms(3000);
    ClearScreen();          // Clear screen and set cursor to first position
    PrintLine((const unsigned char*)"Function", 8);
    T0CON = 0b10001000;     // Setup the timer control register for interrupt
    INTCON = 0b10110000;        // GIE(7) = TMR0IE = INT0IE = 1
    INTCON2bits.INTEDG1 = 0;    // Set pin 34 (RB1/INT1) for negative edge trigger
    INTCON3bits.INT1IF = 0;     // Reset INT1 interrupt flag
    INTCON3bits.INT1IE = 1;     // Enable INT1 interrupt (function up)
    INTCONbits.TMR0IE = 1;      // Enable TMR0 interrupt
    while (1) {
        if (update == 1) {          // LCD update flag is set by INT1 bush button
            INTCONbits.TMR0IE = 0;  // Disable TMR0 interrupt
            update = 0;             // Rest the LCD update flag
```

```
            PrintNum(function, 8);  // Update the function number on LCD display
            SetPosition(64);        // Go to beginning of Line 2;
            switch (function) {
                case 0: PrintLine((const unsigned char*)"Binary counter  ",16); break;
                case 1: PrintLine((const unsigned char*)"ECG simulation  ",16); break;
                case 2: PrintLine((const unsigned char*)"Echo (A/D - D/A)",16); break;
                case 3: PrintLine((const unsigned char*)"Derivative      ",16); break;
            }
            if (function) {   // display "function" in binary on the LEDs
                LEDcount = function << 4;   // shit 4 bots to the left
                PORTB = LEDcount;
            }
            INTCONbits.TMR0IE = 1;  // Ensable TMR0 interrupt
        }
        if (function == 0) {
            LEDcount++;
            LEDcount1 = LEDcount & 0xF0;    // Mask out the lower 4 bits
            PORTB = LEDcount1;              // LED count to RB4-RB7 at ~3 Hz
            Delay_ms(20);
        }
    }
}
```