Lab Manual for BME 361
Biomeasurement Laboratory

The Department of Electrical, Computer, and Biomedical Engineering
University of Rhode Island
Kingston, Rhode Island, 02881
Course website: http://www.ele.uri.edu/courses/bme360/

Spring 2016
Preface

Biomeasurement is fundamental to our understanding of biology, physiology, and medicine. While biomeasurement may refer to any observation (pulse, respiration rate, etc.) increasingly, measurements rely on the advances of technology and engineering to provide more accurate estimation of biological signals (electrical and mechanical) and help interpret the underlying physiological mechanisms responsible for those signals. Perhaps the best known and widely used example of biomeasurement is the ECG (electrocardiogram, or EKG elektrokardiogramm (GER)). The history of the ECG can be traced to 1872 when an electrical engineering PhD student connected wires to the wrist of a patient in St. Bartholomew’s Hospital in London. Since then, it has become the gold standard for vital sign assessment. Though the underlying methodology of obtaining the ECG is little changed in the past 100 years, history, and a fair amount of signal processing techniques, have allowed us to interpret the ECG with surprising diagnostic acumen. To be sure, without the ECG and the countless other forms of biomeasurement, medicine and health care would have a remarkably different landscape.

In this laboratory, you will be introduced to some of the critical design considerations when attempting to acquire a biological signal. Issues such as noise, small signal-to-noise ratios, electrode interface and biocompatibility, as well as signal processing all combine to make the acquisition of a biological signal a non-trivial exercise. Further considerations are given to analog to digital (A/D) conversion, microprocessor-based circuitry, and LCD screen display.
## Contents

1 Introduction to PIC18F4525 and MPLAB: Binary Counter .............................................. 5

2 ECG Simulation .............................................................................................................. 9
   2.1 Lab 2 - ECG Simulation Supplement ................................................................... 12

3 Echo and Derivative Programs ..................................................................................... 14

4 Implementation of Various Modes and LCD Display ................................................... 17
   4.0.1 LAB 4 Supplemental ......................................................................................... 20

5 Introduction to Soldering: ECG Printed Circuit Board .................................................. 26

6 Digital Filters: Low Pass, High Pass, Median, and 60 Hz Notch .................................... 32

7 QRS Detection .............................................................................................................. 34

8 Heart Rate Meter ......................................................................................................... 37

A Introduction to MPLab - Learning Exercise .................................................................. 38
   A.1 Creating a New Project in MPLAB ....................................................................... 38
   A.2 Constructing your Circuit and Connecting the Programmer ................................. 38

B Oscilloscopes 101 ......................................................................................................... 40
   B.1 Time Scale ............................................................................................................ 40
   B.2 Volt Scale ............................................................................................................. 40
   B.3 Coupling ............................................................................................................... 40
   B.4 Trigger ............................................................................................................... 41
   B.5 Autoset ................................................................................................................. 41
   B.6 Other Functions ................................................................................................. 41
List of Figures

1.1 Schematic for test circuit ........................................... 6
1.2 5V regulators and pin assignments .................................. 7
  (a) LM7805 - TO-220 packaging .................................. 7
  (b) Pin assignments ................................................. 7
1.3 Microchip MPLAB ICD3 Programmer .................................. 8

2.1 Schematic of circuit with DAC for ECG simulation .................. 10
2.2 Timing diagram with waveform intervals for ECG simulation .... 12

4.1 Schematic of circuit for LCD screen and multi-mode display .... 18
4.2 ASCII table for generating letters, numbers and symbols .......... 25

5.1 Table of parts. Note that some parts listed on the PCB are no longer used - they are crossed out in this table .......................... 27
5.2 Schematic of PCB layout ............................................... 28
5.3 PCB images with and without components .......................... 30
  (a) Populated PCB. Note the wire placement ......................... 30
  (b) Blank PCB shown with the wire placement and the capacitor orientations ......................... 30
5.4 Back of PCB images without components. Note wire placement ... 30
  (a) Note the placement of the wires you’ll need to solder .......... 30
  (b) The soldered wires .................................................. 30
5.5 Electrode attachment recommendation and resulting ECG waveform ... 31
  (a) Recommended electrode placement .............................. 31
  (b) Oscilloscope showing waveform .................................. 31

7.1 Schematic of completed circuit ..................................... 35
7.2 2N2222 Transistor .................................................... 35

A.1 There are comprehensive instructions to guide you to creating your first project successfully .... 38
A.2 Minimum working example (MWE) circuit .......................... 39

B.1 Waveform appears smeared or blurred ............................ 42
B.2 Stable waveform ..................................................... 43
B.3 Fourier transform of the signal in figure B.2. Notice that the time scale is set to 500 Hz per block and the first peak occurs at the second block, i.e. 1000 Hz. This is to be expected as the signal is a 1 kHz square wave. The other peaks are referred to as the harmonics and occur at ODD integer multiples of the fundamental.
Lab 1: Introduction to PIC18F4525 and MPLAB: Binary Counter

**If you have never used MPLab please refer to Introduction to MPLab - Learning Exercise in the appendix.**

PURPOSE: Simple introduction to the PIC processor as well C++ programming using MPLAB. This lab will give you a basic idea of how to program the PIC processor as well as implement it on a breadboard.

GOAL: Today you will be creating a system with a binary counter that will count from 0-7 (using four LED’s). To complete this task you must first create the circuit shown in Figure 1 on your bread-board, as well as download the code for the binary counter from the course website http://www.ele.uri.edu/courses/bme360/ listed as Sample Program: C code. This is a stock program, meaning it provides basic functionality but does not perform the exact function you will need to build. In its current form, the program simply sets up the analog to digital conversion (A/D) function of the chip. Inside the main loop, you can build the binary counter.

You will need to build the circuit and follow the procedure portion of the lab to get the initial counter working. Once you have the binary counter working correctly, you may move onto the Tasks portion of the lab.

**IMPORTANT: The stock code performs A/D. The program initializes a pin on the chip to use for this purpose. The problem here is that if you leave the pin floating (not connected to anything), the onboard A/D has a difficult time getting a value and so it will affect the performance of your program. There are two ways to deal with this: hardware or software. For the hardware solution, you can simply tie that pin to ground or 5V with a 10 kΩ resistor (this is called a pull-down or pull-up resistor, respectively). Alternatively, you can comment out the section of the code that performs the A/D function. In either case, you will need to determine where in the code this is performed and which pin is assigned to the A/D function.**
MATERIALS:
- 9 Volt Battery & Clip
- Breadboard
- 4 LED’s
- (4) 470 Ω Resistor
- 10 kΩ Resistor
- Microcontroller PIC18F4525
- 4 MHz ceramic resonator
- LM7805 5V voltage regulator
- 6 pin ICSP header

SCHEMATIC:

Figure 1.1: Schematic for test circuit.
PROCEDURE:
Hardware

1. Using your breadboard, implement the circuit seen above in Figure 1.

2. Be sure you have all the necessary components, and that they are oriented correctly. (Note: if you do not know/remember the pin assignment for a certain component, look up the datasheet and use it as a reference.)

*IMPORTANT: The ceramic resonator shown in Figure 1 must be orientated correctly with the label side facing away from the PIC.

Software

1. Open MPLAB on either a desktop computer in the lab, or your own laptop; if you would like to download MPLAB onto your personal laptop, you can follow the installation manual.
2. Download the stock program.

TASKS:

1. You must add code to execute a binary counter in the stock code. You shouldn’t have to change the original code, just write your own new code in the ’main’ section.

2. Once you have modified the code you will need to download the program to your PIC:

   - Connect the MPLAB ICD3 programmer to the ICSP Header and to the PC, and download the program onto the PIC (figure 1.3).
   - YOU MAY NEED TO REMOVE THE LEDs from pins 40 and 39 as the diode creates a capacitive load on the programming pin. This capacitance interferes with the fast changing signals being
sent to your microcontroller by the programmer and computer. Hmm, seems you may have heard something about capacitance and high frequency signals somewhere along your academic prerequisites.

- You do not need to disconnect the MPLAB ICD3 programmer from the ICSP Header. At this point you should see the LED’s lighting up as a binary counter.

![](image)

Figure 1.3: Microchip MPLAB ICD3 Programmer

TROUBLESHOOTING:

There are some common errors related to improperly building the circuit shown in figure 1.1. Pay close attention to the power connections associated with pin 1 as well as the 5V connections. Often, the 5V connections appear after the 10 kΩ resistor that connects to pin one. That is, it’s on the wrong side of the resistor. Remember, if you have a voltage supply on one side of a resistor, a certain amount of the voltage will drop across that resistor depending on what other resistance is in that leg of the circuit. This means that you WILL NOT be supplying 5V to the power connections that require 5V.

Check the value of all resistors. Remember ROYGBIV - Red is 100 multiplier, Orange is 1000 multiplier, Yellow is 10000 multiplier, etc. For instance, a brown stripe followed by a black stripe followed by a yellow stripe is a 100 kΩ resistor - 10 (brown is 1, black is 0) times 10000 = 100000.

Make sure your configuration bits are correctly set. Refer to appendix A, Introduction to MP Lab - Learning Exercise if you are unsure.
Lab 2: ECG Simulation

**You must have a fully functional PIC circuit from LAB 1 to continue on with this lab.**

PURPOSE: Further explore the functional relationship between the PIC Microprocessor and C++ Programming using MPLAB. This lab will include the digital to analog converter and the voltage converter; these chips are essential for the implementation of the ECG simulation. By understanding how these two chips work you will have a greater appreciation for the PIC microprocessor and MPLAB and realize their many possibilities.

GOAL: In today’s lab you will be creating an ECG Simulation by building upon the circuit you constructed in Lab 1. You will do this by adding new components (highlighted in Figure 1), both the DAC0808 (Digital to Analog Converter) and LMC7660 (Voltage Converter) chips, as well as adding a new program.

MATERIALS:
- DAC 0800 chip
- LMC7600
- (2) 0.1uF Capacitor
- 0.01uF Capacitor
- (2) 10uF Capacitor
SCHEMATIC:

Figure 2.1: Schematic of circuit with DAC for ECG simulation

PROCEDURE:

Hardware

1. Using your breadboard and circuit from Lab 1, add the highlighted section seen above in Figure 1 to your previously existing circuit.

2. Be sure you have all the necessary components, and that they are oriented correctly. (Note: if you do not know/remember the pin assignment for a certain component, look up the datasheet and use it as a reference.)

Software

1. Open MPLAB and download the ECG Simulation program. You will need to add this code to the program from Lab 1 - this will be the beginning of your "master program".

2. Connect the MPLAB ICD3 programmer to the ICSP Header and to the PC, and download your new program onto the PIC.

3. Disconnect the MPLAB ICD3 programmer from the ICSP Header.
TASKS:

1. Once you have programmed the PIC you need to use an oscilloscope to test your circuit.

2. Connect the oscilloscope between pins 2 and 4 on the DAC 0800chip.

**NOTE: This means that the "GND" wire of the oscilloscope will be at pin 4 (not connected to GND on your breadboard), while the normal lead will be at pin 2.**

3. The oscilloscope should now display an ECG signal automatically. If at first the signal is too small, you may need to adjust the settings on the oscilloscope to see the signal more clearly.
2.1 Lab 2 - ECG Simulation Supplement

This is a real-time program. Instead of storing the entire ECG waveform in the memory and playing it back, compute what to send to the D/A converter on a point-by-point basis. You can implement the code in the interrupt service routine. Set the timer to generate periodical interrupts at an appropriate interval - suggesting 1 ms. So you need to figure out the hexadecimal numbers for loading the TMR0H and TMR0L. Study the diagram in figure 2.2 carefully for the design of the ECG waveform. Learn to use the "switch" instruction to implement the various modes for generating the individual segments of the ECG waveform. See sample code on the following page.

![ECG Waveform Diagram](image)

Figure 2.2: Timing diagram with waveform intervals for ECG simulation.
1 Declare global variables:
2
3 unsigned char da_output, mode;
4 unsigned int count;
5
6 void _highPriorityInt(void){
7     checkflags:
8     if(INTCONbits.TMR0IE == 1) { // When there is a timer0 overflow, this loop runs
9         INTCONbits.TMR0IE = 0;    // Disable interrupt
10        INTCONbits.TMR0IF = 0;   // Reset timer 0 interrupt flag to 0
11        switch (function) {
12            case 0:     // LAB 1 – BINARY COUNTER
13                TMROH = 0x??;    // Reset timer count: high-order and low-order bytes
14                TMROL = 0x??;   // 0xFFFF – 0x???? = 0x03E8 = 1000 (decimal) ~ 1ms
15                break;
16            case 1:    // LAB 2 – ECG SIMULATION
17                //ad_input = ReadADC();    // Bonus Section – Variable Heart rate
18                TMROH = 0x??;    // Load upper 8 bits to TMROH
19                TMROL = 0x??;   // Load lower 8 bits to TMROL
20                switch (mode) {
21                     case 0:    // P wave up
22                        count++;   
23                        da_output++;   
24                        if (count == 30) mode++;   
25                        break;
26                     case 1:    // P wave flat
27                        count--;   
28                        if (count == 0) mode++;   
29                        break;
30                     case 2:    // P wave down
31                        count++;   
32                        da_output--;   
33                        if (count == 30) mode ++;
34                        break;
35                }
36            }
37            PORTD = da_output;
38            INTCONbits.TMR0IE = 1;    // Re-arm by enabling the Timer0 interrupt
39            goto checkflags;   // Check again in case there is a timer interrupt
40    }
41 

Lab 3: Echo and Derivative Programs

PURPOSE: Introduce students to writing and implementing various programs using C++ and MPLab. Students will modify the provided Echo program to display the derivative of a signal. This lab will enhance students’ programming skills in the C++ language.

GOAL: In today’s lab you will be exploring an echo program and later implementing a derivative program - quite literally taking the derivative (discrete time) of your input signal. You will do this by studying the provided code, and then adapting it to fit the specifications presented in the Tasks portion of this lab.

PROCEDURE:

(You will be using your breadboard and circuit from Lab 2.)

Software

1. Open MPLAB and add the following code. Before you can add this code you must make some minor modification to the ECG simulation code. Essentially, you will be placing all the switch-case statements inside a larger switch-case. Follow the layout of lab 2 when the ECG code was made case 1: (Recall, we also made the Binary Counter case 0:). Now case 2: will be the ECHO and case 3: will be the DERIVATIVE. It will take you a little while to get comfortable with this but it will save time later.

```c
1 // LAB 3 - ECHO
2 case 2:
3  TMR0H =0x??; // Reset timer count for 240 Hz
4  TMR0L = 0x??; // 0xFFFF-0x???? = 4167 => 240 Hz
5  da_output = ReadADC(); // Read A/D and send it to output
6  break;
7 // LAB 3 - DERIVATIVE
8 case 3:
9  TMR0H =0x??; // Reset timer count for 240 Hz
10 TMR0L = 0x??; // 0xFFFF-0x???? = 4167 => 240 Hz
```
data1 = data0;  // Save the previous sample in data1
data0 = ReadADC();  // Read current ADC and save in data0
dumb = data0;
dumb -= data1;  // Backward difference: data0 - data1
dumb += 128;  // Shift baseline up
if (dumb > 255) dumb = 255;
if (dumb < 0) dumb = 0;
da_output = dumb;
break;

2. Connect the MPLAB ICD3 programmer to the ICSP Header and to the PC, and upload the program onto the PIC.

3. Disconnect the MPLAB ICD3 programmer from the ICSP Header.

Hardware

4. You can now test the Echo program using a signal generator and an oscilloscope. You will do this by connecting the positive lead of the signal generator to pin 2 of the PIC (the negative goes to ground) and the oscilloscope probe to pin 2, with the probe ground to pin 4 of the DAC 0800, just like lab 2.

The echo program will take an analog input signal (from the signal generator) and convert that analog signal to a digital signal so it can be used as digital input to the PIC, where it will be reproduced exactly - or at least to within quantization error. The output signal will then be converted back to analog on a digital to analog converter where it can be viewed on an oscilloscope.

5. You may chose to input any type of wave from the signal generator (square, sign, or triangle) and the oscilloscope should display the exact same wave. Be sure to use both channel 1 and 2 on the oscilloscope. Do this by connecting channel 1 to the signal generator and channel 2 to the output (pin 2) of the DAC 0800. By doing this you will be able to determine if your output is the exact "echo" of your input.

TASKS:

1. Now that you have seen what the echo program does, you will need to study the code in order to modify it to produce a derivative program.

The derivative program will take an input from the signal generator just like the echo program, but this time your program in the PIC will produce the derivative of the original input. For example if your input signal is a triangle wave, than your output signal would be a square wave.
HINTS:

1. You will need to declare three unsigned characters (one for the current data point, one for the previous data point, and one for the output), and at least one integer as a "dummy variable".

2. In order to obtain the derivative you will need to subtract each current data point from the previous data point.

3. You will then need to store that value as your dummy variable (at this point this value may be positive or negative depending on the two points you have used). Since you will be running this value through the PIC and ADC, you must find the absolute value of this variable as you cannot use negative values.

4. After you have converted the output back to an unsigned character, you must assign its value to PORTD of the PIC.
Lab 4: Implementation of Various Modes and LCD Display

PURPOSE: Introduce students to writing and implementing numerous modes using C++ and MPLab, as well as introduce a push button and LCD Display to your project.

GOAL: In today’s lab you will be exploring modes and later using them to creating your own program. You will do this by studying code you have been given, and modifying it to fit the specifications presented in the Tasks portion of this lab. You will also have the opportunity to create your own code which will implement a push button to drive the various modes in your program. Finally, you will be give code for an LCD Display which must be slightly modified to accommodate your project.

MATERIALS:

- LCD Display Screen
- Push Button
- 10 kΩ Resistor

SCHEMATIC:
Figure 4.1: Schematic of circuit for LCD screen and multi-mode display

PROCEDURE:

Hardware

1. Using your breadboard and circuit from Lab 3, add the highlighted section seen above in Figure 1 to your previously existing circuit.

Software

1. Open MPLAB and add both the LCD Display and Modes programs to your already existing master program.

TASKS:

1. Since you have been given code for the LCD Display you will only need to create your own function to output variables to the LCD. The program will already be set up to display information on the screen - now you just need to tell it what to display.

2. You have been given sample code for basic mode and LCD programs - now you will need to modify them to include multiple modes to display all the programs you have used in the previous labs.

So far you have worked on an ECG Simulation, as well as Echo and Derivative programs. You will
need to include at least three modes to accommodate these programs.

3. Reference PortB on your datasheet for the PIC to determine how to program the pushbutton you have just added at pin 33. This will require you to use a high priority interrupt routine.

The purpose of an interrupt routine is to put the normal program on hold, execute a subroutine, and then continue on with the normal program. The subroutine can only be executed after a certain event (interrupt) has occurred. In this case you may want to implement a flag which will activate the high priority interrupt.
4.0.1 LAB 4 Supplemental

```c
//BME361/BME463 Lab #4 Implementation of Various Modes and LCD Display  
Supplement

/****************************** Global variables */
unsigned char output, counter, mode, function, data0, data1;
unsigned char LCD_update, LED_count, LED_count1, buttondelay;
int dummy;
.......

/***************************Prototype functions***************/
void backlight(unsigned char state);
void SetPosition(unsigned char position);
void PrintLine(char *string, unsigned char numChars);
void PrintNum(unsigned char value, unsigned char position);
void SetupSerial();
void SetupADC(unsigned char channel);
void delay_ms(unsigned char x);
void ClearScreen();

void _highPriorityInt(void) /***** high priority interrupt service routine  */
*************************/
{
    checkflags:
    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
        if (buttondelay != 0) buttondelay--;    // Delay to debounce pushbutton
        switch (function) {
            case 0:                // 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
                        counter++;          
                        output++;          
                        if (counter == 30) mode++;           
                        break;
                    case 1:            // P wave flat
                        .......
                    .......
```
........
break;
}
break;
case 1:  // Echo
TMR0H = 0xEF;  // Reset timer count: high-order and low-order bytes
TMR0L = 0xB8;  // 0xFFFF-0xFFFF8 =
1047 = 4167 => 4.167ms => 240Hz
output = ReadADC();  //ReadA/DandsendittooutputGO = 1;  //Restart
1047 = 4167 => 4.167 ms => 240 Hz

data1 = data0;  // Save the previous sample in data1
data0 = ReadADC();  // Read ADC and save the present sample in data0
dumb = data0;
dumb -= data1;  // Backward difference: data0 - data1
dumb += 128;  // Shift baseline up
if (dumb > 255) dumb = 255;
if (dumb < 0) dumb = 0;
output = dumb;
break;
PORTD = output;  // Output to the D/A via the parallel port D
INTCONbits.TMR0IE = 1;  // Enable timer interrupt
if (INTCON3bits.INT1IF == 1) {  // When the button (pin 34) is pushed, this
interrupt is called
INTCON3bits.INT1IE = 0;  // Disable interrupt
INTCON3bits.INT1IF = 0;  // Reset interrupt flag
if (buttondelay == 0) {  // If buttondelay is not 0, it's a switch bounce
function++;  // Increment the function mode
if (function == 3) {
    function = 0;  // Back to ECG simulation mode
    output = 50;
    mode = counter = 0;
}
}
LCD_update = 1;  // Signal the main program to update LCD
buttondelay = 200;  // Delay by 100 timer periods to debounce switch
}
INTCON3bits.INT1IE = 1;  // Enable interrupt
goto checkflags;  // Check again in case there is a timer interrupt
}
void Transmit(unsigned char value) {
    while(!P1R8bits.TXIF) continue; // Wait until USART is ready
    TXREG = value; // Send the data
}

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

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

void SetPosition(unsigned char position) {
    Transmit(254);
    Transmit(128 + position);
}

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

void PrintNum(unsigned char value, unsigned char position) {
    unsigned char units, tens, hundreds;
    SetPosition(position); // Set at the present position
    hundreds = value / 100; // Get the hundreds digit, convert to ASCII and send
    if(hundreds != 0) Transmit(hundreds + 48);
    else Transmit(20); // If hundreds = 0, display a space
    tens = value - hundreds * 100; // Get the tens digit
    tens /= 10;
    Transmit(tens + 48); // Convert to ASCII and send
    units = value - hundreds * 100; // Get the units digit
    units -= tens * 10;
    Transmit(units + 48); // Convert to ASCII and send
}
void SetupSerial() { /* Set up the USART Asynchronous Transmit (pin ↔
   25) */
  TRISC = 0x80; // Transmit and receive, 0xCO if transmit only
  SPBRG = 25; // 9600 BAUD at 4MHz: 4,000,000/(16x9600) - 1 = ↔
  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 main() { /* main program ↔
   */
  mode = function = counter = buttondelay = LED_count = 0; // Initialize
  output = 50;
  LCD_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
  backlight(1); // turn backlight on
  ClearScreen(); // Clear screen and set cursor to first position
  SetPosition(0); // Set cursor position to the beginning of line 1
  PrintLine((rom unsigned char)"Hello BME361 :)", 15);
  SetPosition(67); // Go to beginning of Line 2
  PrintLine((rom unsigned char)"Team ???", 8); // Put your trademark here
  delay_ms(255); // Take a deep breath ...
  delay_ms(255);
  delay_ms(255);
  delay_ms(255);
  delay_ms(255);
  delay_ms(255);
  TOCON = 0b10001000; // Setup the timer control register for interrupt
  // bit 7 = GIE - global interrupt enable
  // bit 5 = TMROIE - Timer 0 overflow interrupt enable
  // bit 2 = TMROIF - Timer 0 interrupt flag
  INTCON = 0b10100000;
  INTCON2bits.INTEDG1 = 0; // Enable pin 34 (RB1/INT1) for pushbutton ↔
  interrupt
  INTCON3bits.INTIE1 = 1; // Enable INT1 interrupt
  ADCON1 = 0b00000101; // Make PORTB bits digital
  while (1) {
    if (LCD_update == 1) {
      LCD_update = 0;
    }
ClearScreen();
SetPosition(1);
PrintLine("Function: ", 10);
switch (function) {
case 0:
    PrintNum(function, 11);
    SetPosition(64);
    PrintLine((rom unsigned char*)"ECG Simulation", 14);
    break;

case 1:
    PrintNum(function, 11);
    SetPosition(64);
    PrintLine((rom unsigned char*)"Echo", 4);
    break;

case 2:
    PrintNum(function, 11);
    SetPosition(64);
    PrintLine((rom unsigned char*)"Derivative", 10);
    break;
}

LED_count++;
LED_count1 = LED_count & 0xF0;
PORTB = LED_count1;
delay_ms(62);  // LED count to RB4-RB7 at 1 Hz
## Table of ASCII Codes:

<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(null)</td>
<td>0 0000 0000</td>
<td>(sp)</td>
<td>32 0040 0020</td>
<td>@</td>
<td>64 0100 0040</td>
<td>\</td>
<td>96 0140 0060</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(back)</td>
<td>1 0001 0001</td>
<td>(0x20)</td>
<td>33 0065 0013</td>
<td>A</td>
<td>65 0101 0041</td>
<td>a</td>
<td>97 0141 0061</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tab)</td>
<td>2 0010 0002</td>
<td>(0x02)</td>
<td>34 0066 0014</td>
<td>B</td>
<td>66 0102 0042</td>
<td>b</td>
<td>98 0142 0062</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ltb)</td>
<td>3 0011 0003</td>
<td>#</td>
<td>35 0067 0015</td>
<td>C</td>
<td>67 0103 0043</td>
<td>c</td>
<td>99 0143 0063</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ctrl)</td>
<td>4 0012 0004</td>
<td>$</td>
<td>36 0068 0016</td>
<td>D</td>
<td>68 0104 0044</td>
<td>d</td>
<td>100 0144 0064</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(sv)</td>
<td>5 0013 0005</td>
<td>%</td>
<td>37 0069 0017</td>
<td>E</td>
<td>69 0105 0045</td>
<td>e</td>
<td>101 0145 0065</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(asc)</td>
<td>6 0014 0006</td>
<td>&amp;</td>
<td>38 006A 0018</td>
<td>F</td>
<td>70 0106 0046</td>
<td>f</td>
<td>102 0146 0066</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(del)</td>
<td>7 0015 0007</td>
<td>'</td>
<td>39 006B 0019</td>
<td>G</td>
<td>71 0107 0047</td>
<td>g</td>
<td>103 0147 0067</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(en)</td>
<td>8 0016 0008</td>
<td>(</td>
<td>40 006C 001A</td>
<td>H</td>
<td>72 0108 0048</td>
<td>h</td>
<td>104 0148 0068</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ht)</td>
<td>9 0017 0009</td>
<td>)</td>
<td>41 006D 001B</td>
<td>I</td>
<td>73 0109 0049</td>
<td>i</td>
<td>105 0149 0069</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(nl)</td>
<td>10 0018 0010</td>
<td>*</td>
<td>42 006E 001C</td>
<td>J</td>
<td>74 010A 004A</td>
<td>j</td>
<td>106 014A 006A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(rt)</td>
<td>11 0019 0011</td>
<td>+</td>
<td>43 006F 001D</td>
<td>K</td>
<td>75 010B 004B</td>
<td>k</td>
<td>107 014B 006B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(np)</td>
<td>12 001A 0012</td>
<td>-</td>
<td>44 0070 001E</td>
<td>L</td>
<td>76 010C 004C</td>
<td>l</td>
<td>108 014C 006C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(cr)</td>
<td>13 001B 0013</td>
<td></td>
<td></td>
<td>45 0071 001F</td>
<td>M</td>
<td>77 010D 004D</td>
<td>m</td>
<td>109 014D 006D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(sn)</td>
<td>14 001C 0014</td>
<td>,</td>
<td>46 0072 0020</td>
<td>N</td>
<td>78 010E 004E</td>
<td>n</td>
<td>110 014E 006E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(st)</td>
<td>15 001D 0015</td>
<td>.</td>
<td>47 0073 0021</td>
<td>O</td>
<td>79 010F 004F</td>
<td>o</td>
<td>111 014F 006F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ctl)</td>
<td>16 001E 0016</td>
<td>/</td>
<td>48 0074 0022</td>
<td>P</td>
<td>80 0110 0050</td>
<td>p</td>
<td>112 0150 0070</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ASCII table of all printable letters, numbers, and symbols.**
Lab 5: Introduction to Soldering: ECG Printed Circuit Board

**If you have never used a soldering iron before please do a few practice rounds before starting this lab**

PURPOSE: Introduce students to the use of a soldering iron.

GOAL: Today you will be creating an ECG amplifier on a previously designed printed circuit board. To complete this task you must create the circuit shown in the schematic on a printed circuit board. Each group will be given a set of leads which will be connected via electrodes to one member of your group. This will enable you to collect real data which you will be able to view on an oscilloscope.

You will need to build the circuit by following the procedure portion of the lab. Once you have the ECG PCB completely soldered, you may move onto the Tasks portion of the lab.

MATERIALS:
- Printed Circuit Board (P.C.B.)
- Soldering Iron Station
- All components listed in table 5.1
- ECG Leads (Right Arm, Left Arm, Left Leg)
- Battery and battery clip
- Auxiliary jack connector (¼” audio jack)

COMPONENTS:
- (2) 8-Pin Sockets (U3 and U4)
- 14-Pin Socket (U1)
- Mono Audio Jack
- 2 and 4 pin Headers
<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
<th>Color</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>27K</td>
<td>Red</td>
<td>E</td>
</tr>
<tr>
<td>R2</td>
<td>1K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R3</td>
<td>27K</td>
<td>Red</td>
<td>E</td>
</tr>
<tr>
<td>R4</td>
<td>1K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R5</td>
<td>100K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R6</td>
<td>22K</td>
<td>Red</td>
<td>E</td>
</tr>
<tr>
<td>R7</td>
<td>330K</td>
<td>Org</td>
<td>E</td>
</tr>
<tr>
<td>R8</td>
<td>10K</td>
<td>Poten.</td>
<td>E</td>
</tr>
<tr>
<td>R9</td>
<td>10K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R10</td>
<td>10K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R11</td>
<td>10K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R12</td>
<td>10K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R13</td>
<td>12K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R14</td>
<td>27K</td>
<td>Org</td>
<td>E</td>
</tr>
<tr>
<td>R15</td>
<td>27K</td>
<td>Org</td>
<td>E</td>
</tr>
<tr>
<td>R16</td>
<td>5.1K</td>
<td>Blk</td>
<td>E</td>
</tr>
<tr>
<td>R17</td>
<td>3.3K</td>
<td>Org</td>
<td>E</td>
</tr>
<tr>
<td>C1</td>
<td>226 nF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>47 μF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>100 μF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.1: Table of parts. Note that some parts listed on the PCB are no longer used - they are crossed out in this table.
Figure 5.2: Schematic of PCB layout.
PROCEDURE:

1. Using the Printed Circuit Board you are given, create the circuit seen in the schematic. To complete this circuit all you need to do is find the component on the schematic, match it up to the corresponding label on the P.C.B., and solder the component to the board.

**Note 1: You will NOT be soldering the larger chip components (324, 741, 7660, 620) directly to the PCB - instead you will be soldering 8 and 16-pin sockets to the board and then inserting the components into the sockets when you have completed soldering the board. **Note 2: The values of R1 and R3 (27 kΩ) must be exactly matched - be sure to check these resistors before soldering!**

2. Make sure you have all the necessary components, and that they are oriented correctly.

**Note: if you do not know/remember the pin assignment for a certain component, look up the datasheet and use it as a reference.**

3. There are four jumper wires that need to be added to the circuit: one grounding wire on the front, and three signal wires on the back.

(U5, Pin 2 → U1, Pin 9) (U5, Pin 3 → U1, Pin 10) (U5, Pin6 → U1, Pin 8) (R13, Right side → R17, Left side)

*There are four capacitors, and two chips that are no longer used. These should be left blank. Capacitors 1 and 12 are also not labeled on the PCB but are labeled in figure 5.3a. These should be what the final product looks like.
Figure 5.3: PCB images with and without components.

(a) Populated PCB. Note the wire placement.  (b) Blank PCB shown with the wire placement and the capacitor orientations.

Figure 5.4: Back of PCB images without components. Note wire placement.

(a) Note the placement of the wires you’ll need to solder.  (b) The soldered wires.
TASKS:

1. Now you can connect the leads to your p.c.b. To do this you need to attach electrodes to you or one of your group members. The leads will be connected on the right arm, left arm, and left leg. Figure 5.5a above indicates the most successful diagram, but you may want to vary these slightly to obtain best results. These leads are attached to the alligator clips, then to the four-pin connector on the p.c.b.

2. In order to see a visible ECG, the circuit needs to be attached to the oscilloscope. Connect the socket end of the audio jack to the p.c.b, and the other end to the oscilloscope.

3. Last, connect your battery to the p.c.b.

4. You should now be able to see a functional ECG, similar to the one shown in figure 5.5.

Figure 5.5: Electrode attachment recommendation and resulting ECG waveform.
Lab 6: Digital Filters: Low Pass, High Pass, Median, and 60 Hz Notch

PURPOSE: To create digital filters using C++ coding instead of building them on your breadboard. The advantages to using digital filters are that they can be easily designed, tested, and changed without affecting the circuitry (hardware), whereas an analog filter can only be changed by redesigning the circuit. Another advantage is that digital filters are extremely stable and are not affected by the external environment and subject to temperature or component error like analog filter circuits.

GOAL: Today you will be adding code for High Pass, Low Pass, Median, and 60 Hz Notch filters to your previously existing code. To do this you must first add the given Low Pass Filter program to your master program and then modify it to produce the three programs described in the Tasks portion of the Lab.

PROCEDURE:

Hardware

1. You will be using both the circuit and p.c.b. from Lab 5.

Software

1. Open MPLAB and add Low Pass Filter program to your already existing master program.

TASKS:

1. You were given the code for a Low Pass Filter - now you need to modify that code to create High Pass, Median, and 60 Hz Notch Filters.

Low Pass (LP) Filters eliminate all frequencies above the predetermined cut-off frequency; while leaving the frequencies below the cut-off unchanged.

High Pass (HP) Filters are the opposite of a LP - they eliminate all frequencies below the predetermined cut-off frequency; while leaving the frequencies above it unchanged.
Median Filters are smoothing and moving filters which are used to eliminate noise while preserving the edge values of your data.

60Hz Notch Filters also eliminate noise but only noise at 60 Hz.

All of these filters will help clean up your ECG signal.

2. Once you have finished the three new filters you can add them to your master program.

3. You also need to add four new modes to your program for the LP, HP, Median, and 60 Hz Notch filters, as well as add some code for the LCD Display.

4. Once you have programmed your code you can connect leads to one of your group members and run the signal through your ECG p.c.b. and then through your four new filters. The pushbutton will allow you to switch between filters.

5. You should be able to view and ECG waveform after it has run through your filters on the oscilloscope.
Lab 7: QRS Detection

PURPOSE: To brings students one step closer to a fully functional heart rate meter. With the use of a buzzer and the QRS code, students will hear a beeping sound every time a peak is detected. By doing this, we will be almost finished creating a heart rate meter similar to the ones used by hospitals.

GOAL: To combine the ECG printed circuit board you created in Lab 5, with a previously written QRS Detection Program. This program was written to detect the QRS peak of an ECG waveform; the program sends a signal through the PIC to a Buzzer and LED every time a certain threshold has been reached. This will create the "beep"beep"beep" you hear on a hospital heart rate meter.

MATERIALS:
- TDB-12PN Buzzer
- LED
- 2N2222 NPN Transistor
- 470 Ω Resistor
- 10 kΩ Resistor

SCHEMATIC:
PROCEDURE:

Hardware

1. Using your breadboard, add the highlighted section seen in the schematic to your previously ex-
isting circuit.

**NOTE: The Buzzer is very loud, so in order to mute this device it is advisable that you cover the top hole with masking tape or another similar material.**

Software

1. Open MPLAB and add the QRS Detection program to your master program.

2. Connect the MPLAB ICD3 programmer to the ICSP Header and to the PC, and download the program onto the PIC.

3. Disconnect the MPLAB ICD3 programmer from the ICSP Header.

TASKS:

1. Now that the output from your ECG p.c.b. is connected to pin 2 of the PIC, you can attach the ECG leads to one of your group members and the signal should run through the ECG circuit and then through the QRS Detection Program in your PIC. If your circuit is functioning correctly your LED and Buzzer will be turned on and off continuously.

2. You can now test your circuit using an oscilloscope; connect channel 1 at the output of your ECG p.c.b., and channel 2 at pin 36 of the PIC. This will show you the ECG signal as well as the function of the QRS Detection Program. **NOTE: The group member who is connected to the ECG should remain as still as possible.**

You will be using an algorithm known as the Multiplication of Backward Differences. As the name implies, rather than running a forward derivative, we'll be performing a backward derivative (in discrete time, a derivative is well approximated by a difference, i.e. a subtraction.) Those differences are then multiplied, providing a very robust peak detector. The following code will help implement the algorithm.
Lab 8: Heart Rate Meter

PURPOSE: In this lab you will be combining all the work you have done throughout the semester to create a functional heart rate meter with an LCD screen that displays eight separate modes.

GOAL: To create the final mode of your project and combine it with all previous components.

PROCEDURE:

Hardware

1. You will be using the circuit you constructed in Lab 7. If you would like to view a larger version of the schematic it can be found in the file Final Schematic.

TASKS:

1. You will now need to add another mode for the Heart Rate Meter; this mode will correspond to the data from the QRS Detection. You will also need to write code for the LCD Display:

To do this you must write a function to take points from the input signal, create three consecutive positive differences, take the average, and then create an algorithm which uses this data to display the beats per minute on your LCD screen.

At this time your program should include the following eight modes:


The names of modes 1 - 7 should be displayed on the LCD screen as you switch between modes. Mode 8 should display the name, as well as beats per minute.
Appendix A: Introduction to MPLab - Learning Exercise

A.1 Creating a New Project in MPLAB

In the MPLABX start-screen, click the Quick Start icon and follow the instructions.

Figure A.1: There are comprehensive instructions to guide you to creating your first project successfully.

A.2 Constructing your Circuit and Connecting the Programmer

A schematic for the circuit is shown below:

**Note: Before connecting the PIC, plug in the battery and test to make sure the correct voltages are visible at the appropriate locations. A clear sign that the wrong voltage is being supplied to the PIC is the D/A converter will heat up very rapidly. Supplying the wrong voltage to the PIC can permanently damage it, so continually check voltages throughout your use to prevent this from happening.**
Figure A.2: Minimum working example (MWE) circuit.
Appendix B: Oscilloscopes 101

By now, you have undoubtedly seen an oscilloscope. You may even have played a bit with more of the knobs and buttons than just the auto-set. However, you may not know how to set all the parameters to insure that you will see the signal you’re interested in seeing. For instance, having the oscilloscope set to AC coupling instead of DC coupling will give a completely different waveform. If that last sentence has you scratching your head, keep reading.

There is a fairly comprehensive tutorial written by the folks at Tektronix (the company that makes the oscilloscopes we use) at www.tek.com/Measurement/App_Notes/XYZs/03W_8605_2.pdf. It’s 64 pages long and worth a look. As shorter, more compact reference can be found at http://oscilloscope-tutorials.com/Oscilloscope/controls.asp. We’ll address some of the key points and most often overlooked settings when using an oscilloscope - and keep it to just a few pages.

B.1 Time Scale

We begin with the time scale setting since this is the one that is most likely to be misunderstood. Each block on the vertical axis of the oscilloscope (there are grid lines in the horizontal and vertical axes, called ticks) has a time scale. This means that each block represents a certain period of time. If the time scale is set to 10 microseconds and there are 10 blocks on the screen, the signal on the screen represents 100 microseconds of the signal. If the signal you are interested in viewing is very long, upwards of a second (such as a QRS complex), it would be necessary to set the time scale resolution to 0.1 seconds in a 10 block window.

B.2 Volt Scale

Obviously, the volt scale allows you to make the signal you are viewing appear bigger or smaller as a result of changing the scale. But what often presents a problem is that the probe you are using may have a gain setting of 10 times the signal you are using. This is handy if you are measuring really low amplitude signals, nanovolts or less. But for most signals, your probe should be set to 1 times gain. Regardless of which setting you are using, make sure the oscilloscope setting matches the same value as the probe.

B.3 Coupling

There are three types of coupling: AC, DC, and ground coupling. AC coupling allows you to see a signal at 0 mean (no DC bias or offset) and allows for faithful representation of an oscillatory signal. If one
were to have a pure sinusoid that was riding on a 5 volt DC bias and viewed the sinusoid in AC coupling
mode, the signal would appear on the scope to be oscillating about 0 volts. Use this mode if all you
want to view is the oscillatory pattern.
Similarly, if you are viewing DC signals, i.e. signals composed of discrete DC values, you should use DC
coupling. This does not mean that you will not see oscillatory patterns. On the contrary, if your signal
has oscillations but are described at specific DC offsets (like a simulated QRS complex) you will be able
to determine where the signal is rising and falling against its DC offset. If one were to look at such a
signal in AC coupled mode, the signal would oscillate about 0 volts and would look different than the
DC mode.
Ground coupling literally ties the signal you are trying to measure to ground. It really is just a way to
measure your signal against ground. Remember, ground is a relative term, it may not be 0 volts.

B.4 Trigger

Triggering is very helpful when you want to see a stable waveform. Often, digital oscilloscopes have
difficulty sampling a high-frequency signal fast enough to maintain the waveform structure (fig. B.1).
By setting the moving the trigger cursor (ellipse) up to the waveform, the oscilloscope freezes the
waveform at the first rising edge encountered by the cursor (fig. B.2). This oscilloscope, like others,
allows for a host of triggering options.
Similar results can be achieved with the run/stop button. The problem with this approach is that there
is a delay between the time you press the button and the capture. This leads to the possibility of
capturing a portion of the waveform that was not intended.

B.5 Autoset

The longtime favorite of students around the globe. The oscilloscope’s software analyzes the signal,
looking for DC bias, rising and falling edges, and frequency content. Once it has these parameters
calculated, it sets the oscilloscope tick spacing (in both voltage and time) and sets the correct type of
coupling, which will generally either be AC or DC. It MAY NOT be the best way to view your signal,
but can often get you close so that your "tuning" will be minimized.

B.6 Other Functions

Some oscilloscopes have a built-in math function that allows you to perform some type of transformation,
such as a Fourier transform (fig. B.3). You typically also have control over where to place your cursors
to allow you to reference your signal in some tighter interval. The more expensive the oscilloscope, the
more options. Measuring very fast signals (high frequency, maybe transient) with high bandwidth often
means very expensive oscilloscopes. For instance, to measure high frequency laser light (10s of gigahertz)
it is not uncommon for oscilloscopes to cost upwards of $100k.
Figure B.1: Waveform appears smeared or blurred.
Figure B.2: Stable waveform.
Figure B.3: Fourier transform of the signal in figure B.2. Notice that the time scale is set to 500 Hz per block and the first peak occurs at the second block, i.e. 1000 Hz. This is to be expected as the signal is a 1 kHz square wave. The other peaks are referred to as the harmonics and occur at ODD integer multiples of the fundamental.