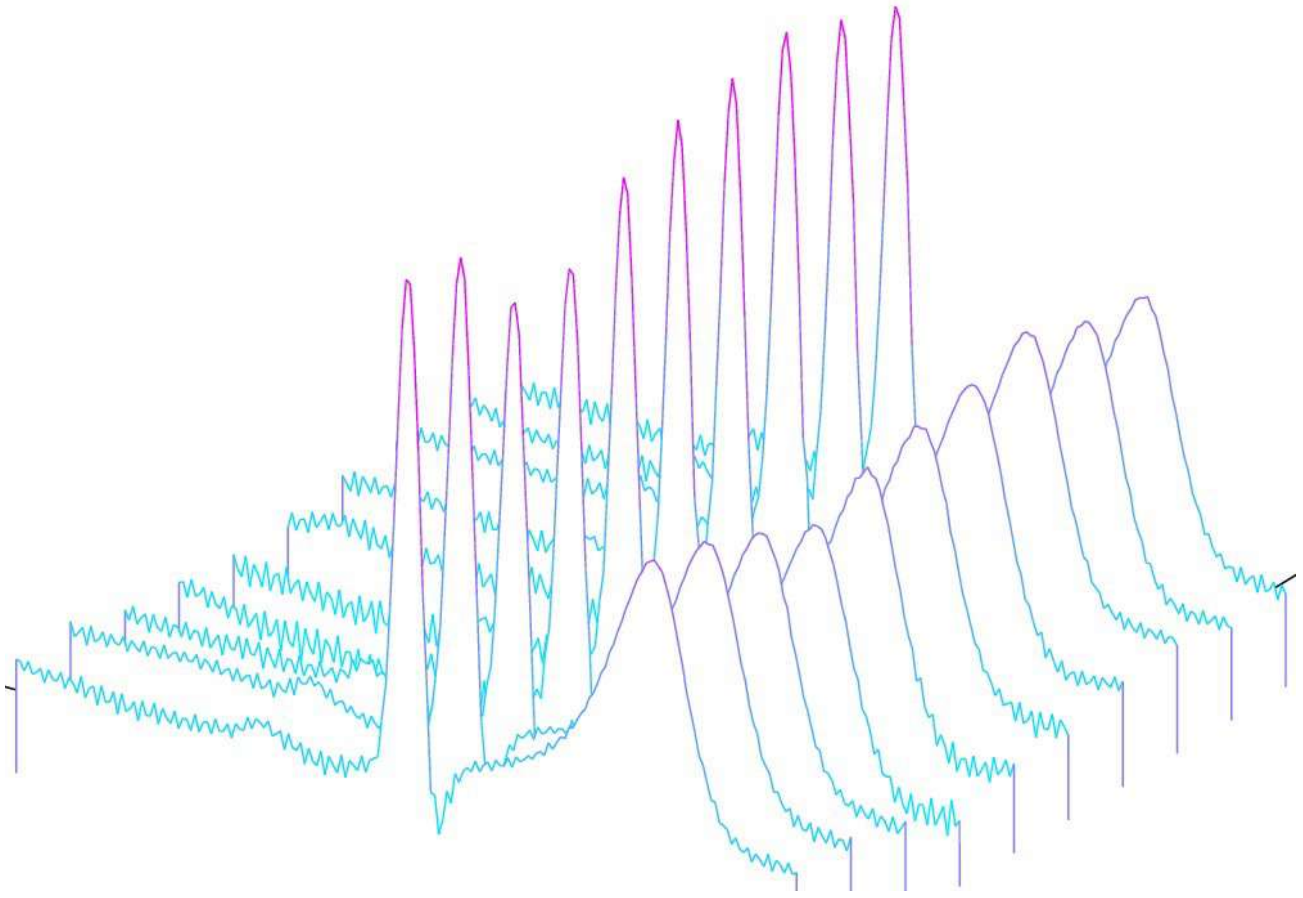
UNIVERSITY OF RHODE ISLAND



BME 361 Lab Manual

Biomeasurement Laboratory

Department of Electrical, Computer, and Biomedical Engineering

http://www.ele.uri.edu/courses/bme360/

Fall 2018

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

List of Figures

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 breadboard, 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 blink functionality but does not perform the exact final 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 code working, you may move on to modifying the code.

****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:

5 Volt Rechargeable Battery
USB Micro Breakout Board and USB cable
Breadboard
(4) LED's
(4) 470 Ω Resistor
10 kΩ Resistor
PIC18F4525 Microcontroller
4 MHz ceramic resonator

SCHEMATIC

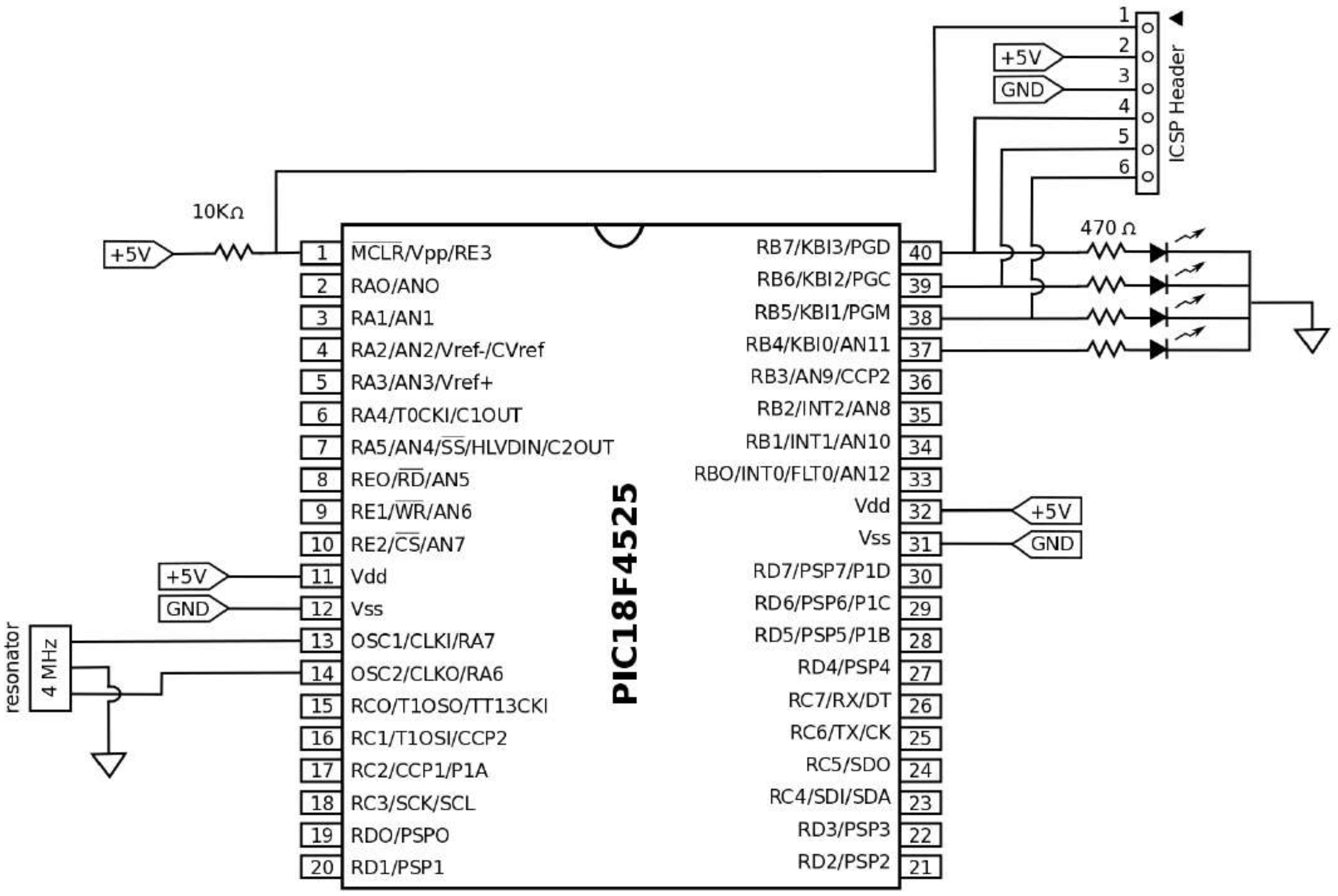


Figure 1.1: Schematic for lab 1

PROCEDURE:

Hardware

1. Using your breadboard, implement the circuit seen above in Figure 1.1.
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 data-sheet and use it as a reference.)

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 in the appendix at the end of this lab manual.
2. Download the stock program from the course webpage.

TASKS:

1. Run your sample C code. Make sure it builds and compiles. Download the program to your PIC: Connect the MPLAB ICD3 **OR** PicKit3 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. Hmmm, seems you may have heard something about capacitance and high frequency signals somewhere along you academic prerequisites. You do not need to disconnect the MPLAB ICD3 programmer from the ICSP Header. At this point you should see a single LED Blink.

** The arrow on the PICkit 3 and ICD 3 (usually blue wire) refer to the arrow on the ICSP Header. This means this connects to pin 1 on the PIC. **

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 can repeat the steps to program the PIC.

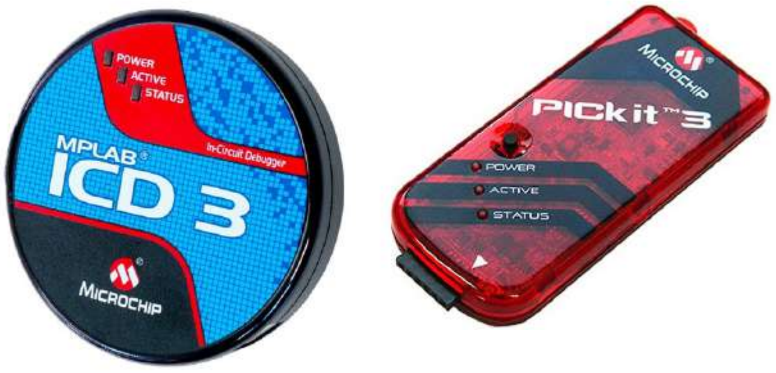


Figure 1.2: Microchip MPLAB ICD3 and PICkit 3 Programmer

TROUBLESHOOTING:

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.. Double check that your PIC pins 37-40 are connected directly to the ICSP header. If you connect this after the resistor/LED , you'll essentially be shorting your communication lines to ground. 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 MPLab 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 inverter; 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 2.1), the DAC 0800 and LMC7660, as well as adding a new code.

MATERIALS:

DAC 0800 Digital to Analog Converter	1.5 Ω Resistor
LMC7600 Voltage Inverter	(2) 4.7 Ω Resistor
0.01uF Capacitor (103)	(2) 10K Ω Resistor
0.1uF Capacitor (104)	
10uF Capacitor	
100uF Capacitor	

SCHEMATIC:

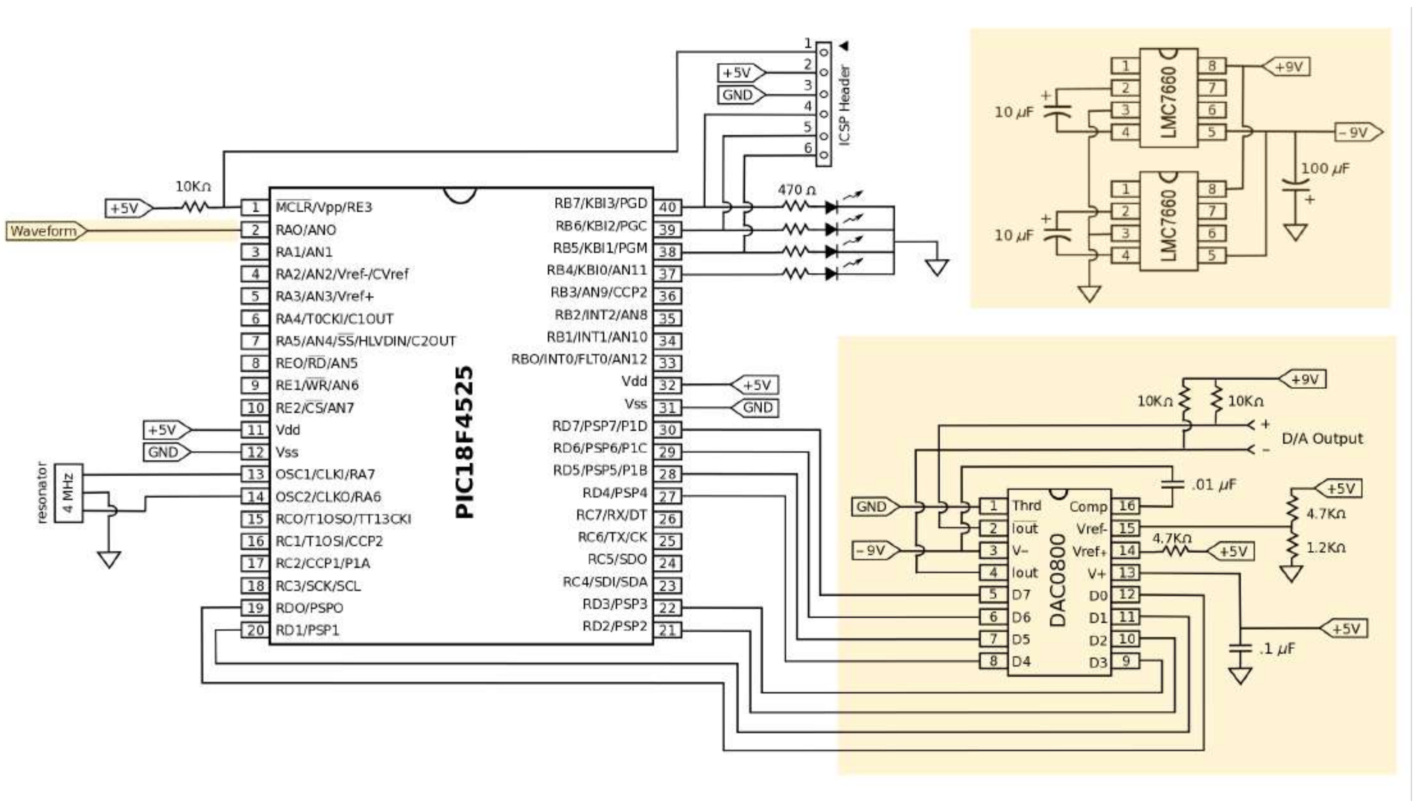


Figure 2.1: Schematic of circuit with DAC for ECG simulation

PROCEDURE:

Hardware

1. Using your breadboard and circuit from Lab 1, add the highlighted section seen above in Figure 2.1 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 data sheet and use it as a reference.)
3. The voltage inverter (LM7660) only requires one chip. The schematic shows a two-chip setup which provides a steadier power supply.

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 theMPLAB ICD3 programmer to the ICSP Header and to the PC, and download your new program onto the PIC.
3. Disconnect theMPLAB 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" alligator clip 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 or “auto range” to see the signal more clearly.

LAB 2.1 : 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.

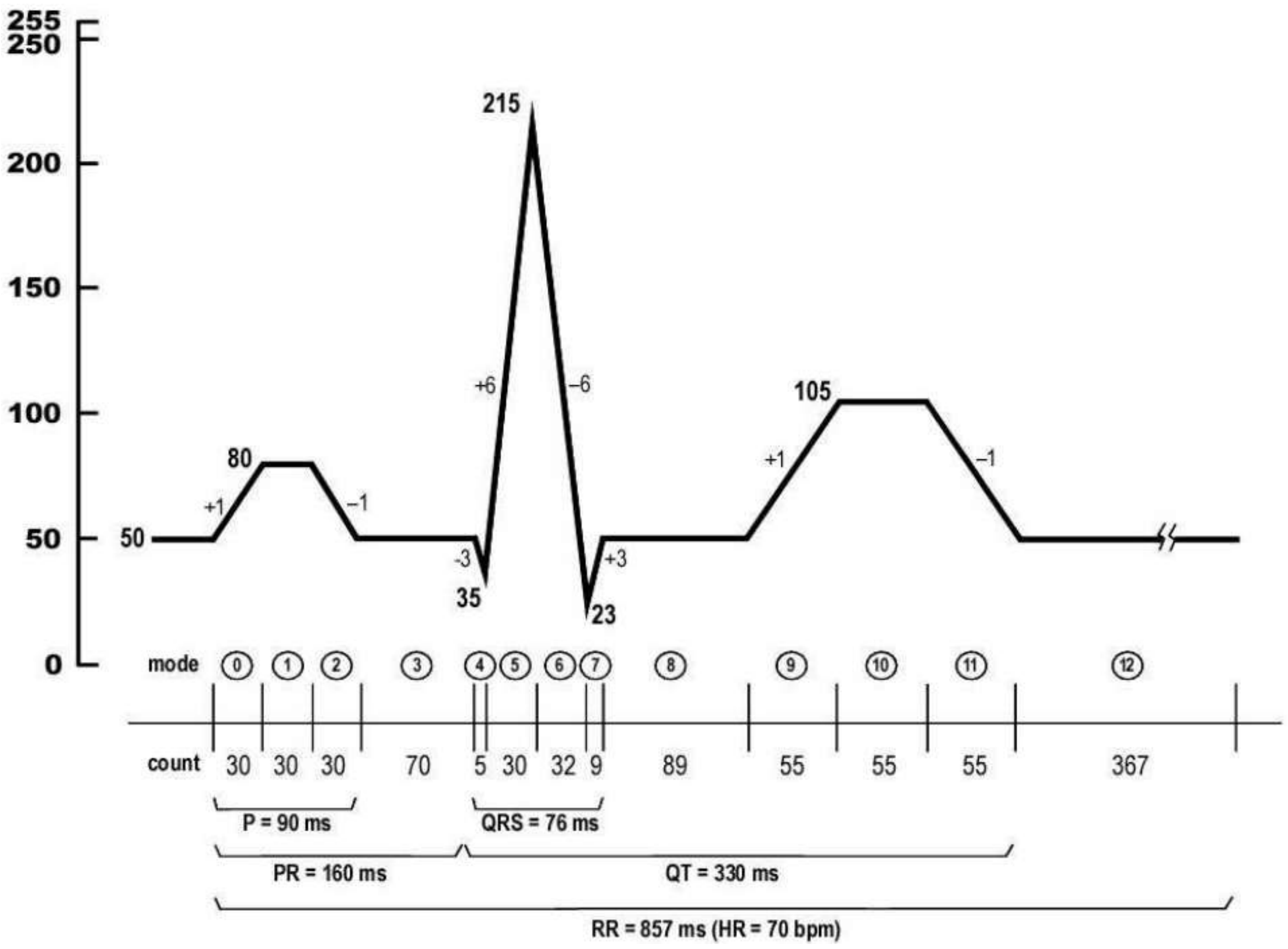


Figure 2.2: Timing diagram with waveform intervals for ECG simulation.

Declare global variables: unsigned char da-output, count, mode.

```
……
void _highPriorityInt(void){
                if(TMR0IF == 1) {       // Check for timer interrupt
                TMR0IE = 0;             // Disable interrupt
                TMR0IF = 0;             // Reset timer 0 interrupt flag to 0
                TMR0H = 0x??;           // Set timer 0 for 1 ms interrupts
                TMR0L = 0x??;
                switch(mode){
                case 0:                 // P wave up
                        count++;
                        da_output++;
                        if (count == 30) mode++;
                        break;
                case 1:                 // P wave flat
                        count--;
                        if (count == 0) mode++;
                        break;
                case 2:                 // P wave down
                        count++;
                        da_output--;
                        if (count == 30) mode ++;
                        break;
                ……
                }
                PORTD = da_output;
                TMR0IE = 1; // Re-arm by enabling the Timer0 interrupt
                }
}
```

Figure 2.3: Sample code

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 the specifications presented in the Tasks portion of this lab.

PROCEDURE:

Hardware

(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 modifications 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.

```
// LAB 3 – ECHO
case 2 :
        TMR0H =0x ? ? ;              // Reset time r count for 240 Hz
        TMR0L = 0x ? ? ;             // 0xFFFF–0x ???? = 4167 => 240 Hz
        da_output = ReadADC ( ) ;    // Read A/D and send it to output
        break ;
// LAB 3 – DERIVATIVE
case 3 :
        TMR0H =0x ? ? ;              // Reset timer count for 240 Hz
        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 += 1 2 8;               // Shift baseline up
        if ( dumb > 2 5 5 ) dumb = 2 5 5;
        if ( dumb < 0 ) dumb = 0 ;
        da_output = dumb ;
        break ;
```

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

3. Disconnect theMPLAB ICD3 programmer from the ICSP Header.

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 toPORTD 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 t 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 modi ed to accommodate your project.

MATERIALS:

LCD Display Screen

(2) Push Button

(2) 10 kΩResistor

SCHEMATIC:

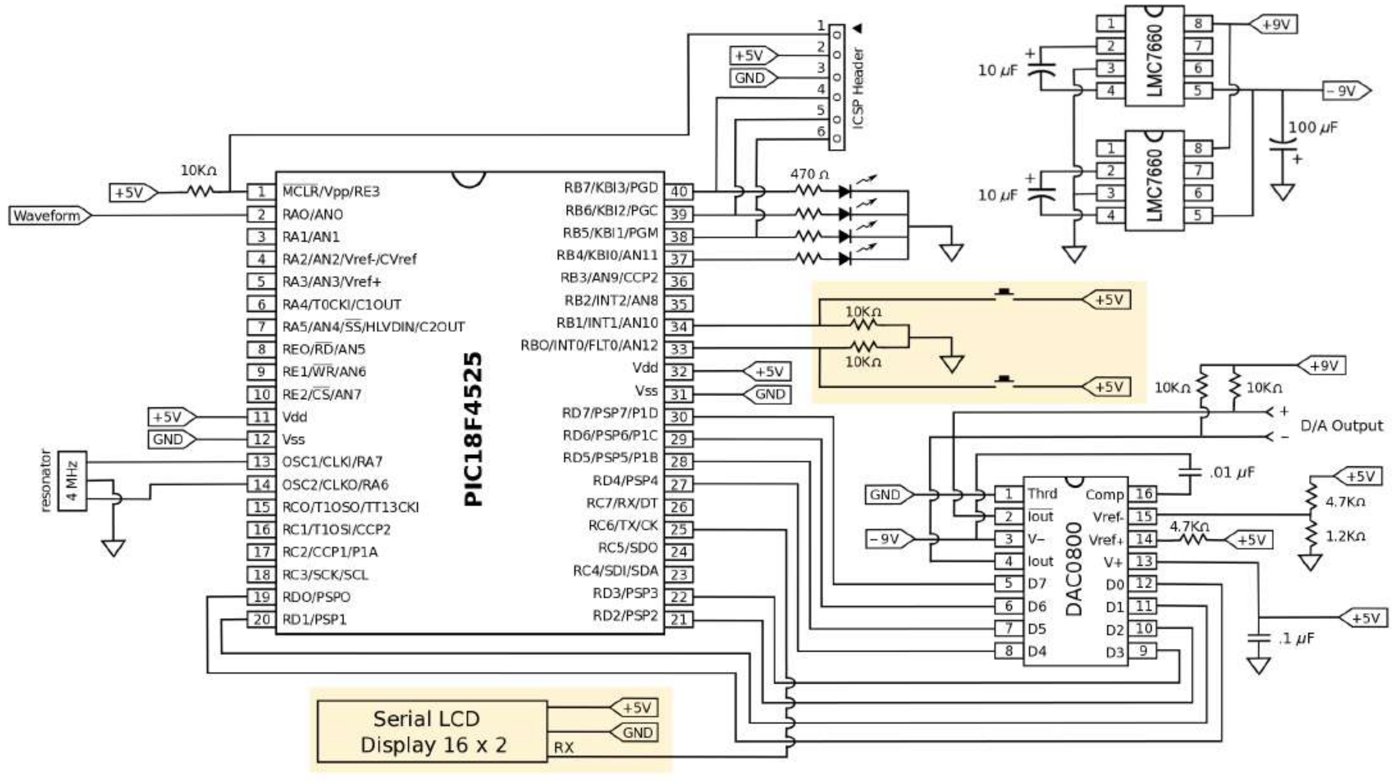


Figure 4.1: Schematic of circuit for LCD screen and multi-mode display

PROCEDURE:

Hardware

1. Using your breadboard and circuit from Lab 3, add the highlighted section seen above in Figure 4.1 to your circuit.

Software

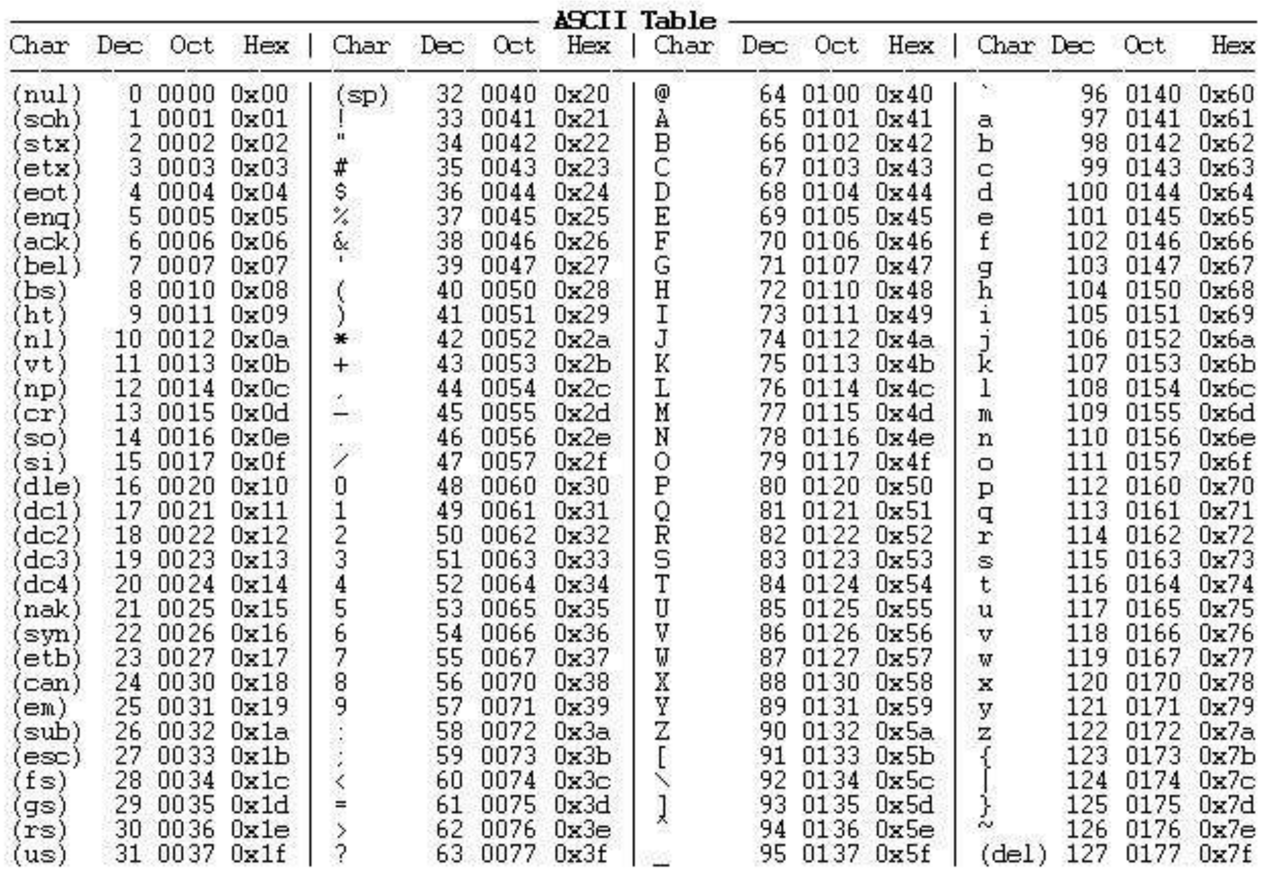
1. Open MPLAB and add both the LCD Display and Modes programs to your already existing master program.

TASKS:

1. Since you have been given code for the LCD Display you will only need to create your own functions 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 you data sheet for the PIC to determine how to program the pushbutton you have just added at pin 33 and 34. 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.

Table of ASCII Codes:



ASCII Table

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

Figure 4.2: ASCII table for generating 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

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 (PCB)
- Soldering Iron Station
- All components listed in table 5.1
- ECG Leads (Right Arm, Left Arm, Left Leg)
- ECG electrodes (stickers)
- Battery and battery clip
- Auxiliary jack connector (3.5mm audio jack)

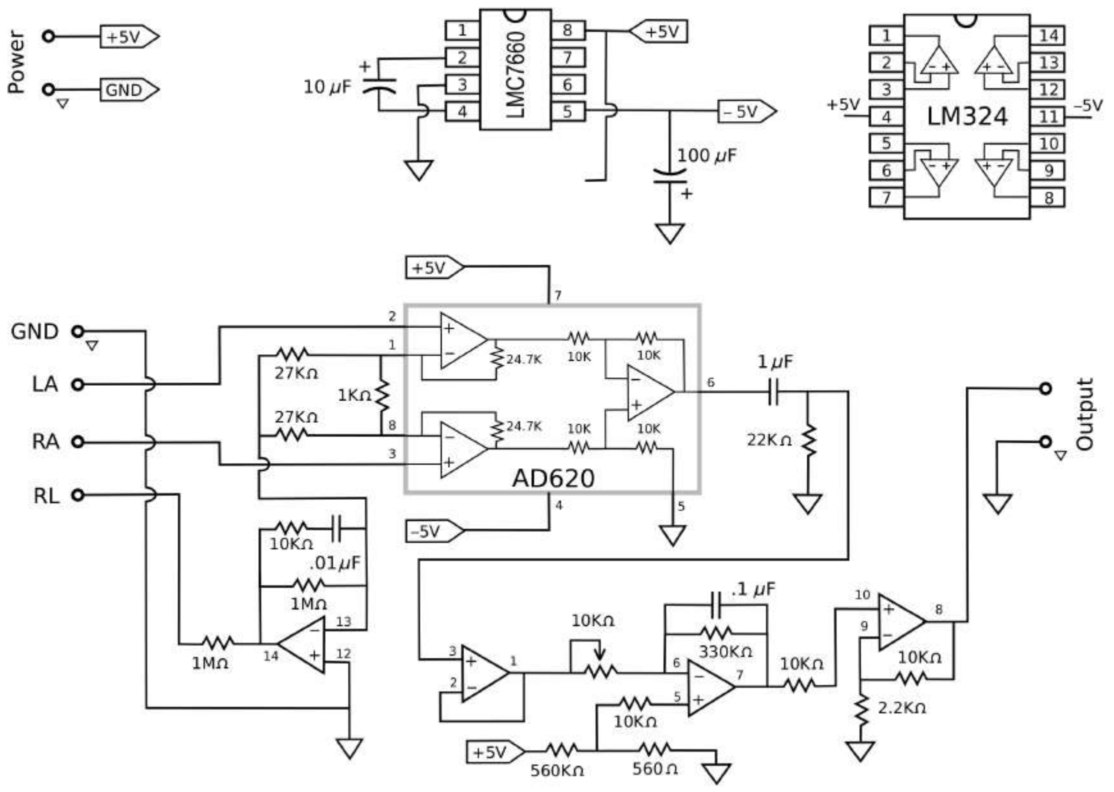
COMPONENTS:

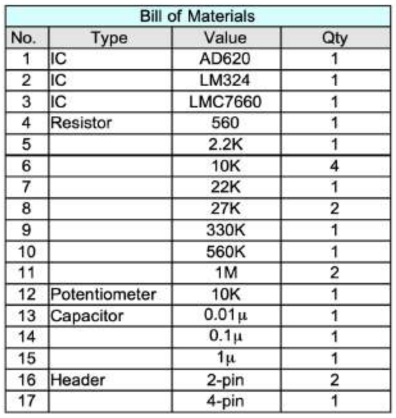
- (2) 8-Pin Sockets (U3 and U4)
- 14- Pin Socket (U1)
- Mono Audio Jack
- 2 and 4 pin Headers

R1	27K	Red Vio Org	C1	226nF
R2	1K	Brw Blk Red	~~C2~~	
R3	27K	Red Vio Org	~~C3~~	
R4	1K	Brw Blk Red	~~C4~~	
R5	100K	Brw Blk Yel	C5	47uF
R6	22K	Red Red Org	C6	100 uF
R7	330K	Org Org Yel	~~C7~~	
R8	10K Pot	Potentiometer	C8	1uF
R9	10K	Brw Blk Org	C9	1nF
R10	10K	Brw Blk Org	C10	100nF
R11	10K	Brw Blk Org	C11	100nF
R12	10K	Brw Blk Org	C12	10nF
R13	12K	Brw Red Org	U1*	LM324
R14	27K	Red Vio Org	~~U2~~	
R15	27K	Red Vio Org	U3*	LM7660
R16	5.1K	Grn Brw Red	U4*	AD620
R17	3.3K	Org Org Red	~~U5~~	

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. *Make sure you are using sockets for all IC's*

SCHEMATIC:





Bill of Materials			
No.	Type	Value	Qty
1	IC	AD620	1
2	IC	LM324	1
3	IC	LMC7660	1
4	Resistor	560	1
5		2.2K	1
6		10K	4
7		22K	1
8		27K	2
9		330K	1
10		560K	1
11		1M	2
12	Potentiometer	10K	1
13	Capacitor	0.01μ	1
14		0.1μ	1
15		1μ	1
16	Header	2-pin	2
17		4-pin	1

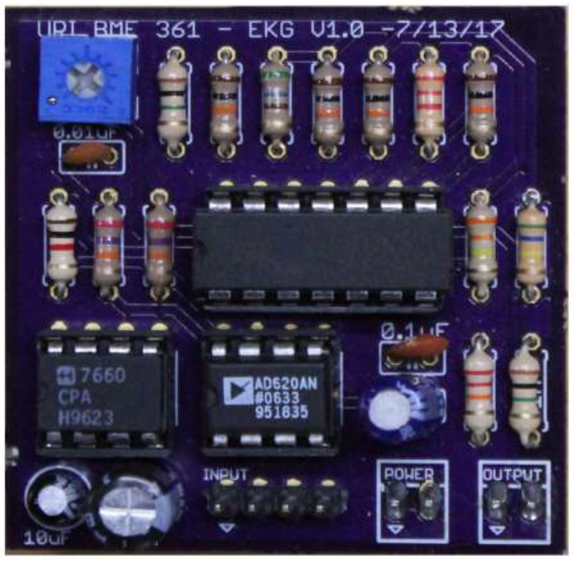


Figure 5.2: Schematic of PCB layout.

PROCEDURE:

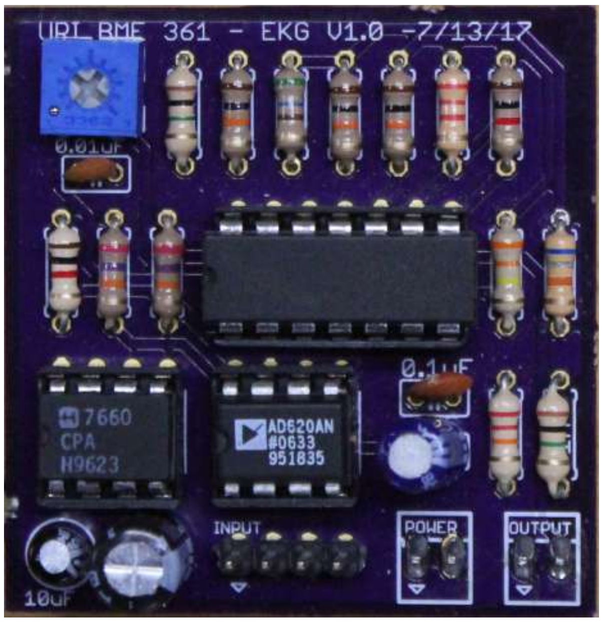
1. Using the Printed Circuit Board you are given, create the circuit seen in the schematic. To complete this circuit all you need to do is add the component on the schematic, match it up to the corresponding label on the P.C.B., and solder the component to the board.

   Note 1: You will NOT be soldering the larger chip components (LM324, LM7660, AD620) 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 data sheet and use it as a reference.

(a) Populated PCB. Note the wire placement. (b) Blank PCB shown with the wire placement and the capacitor orientations. Notice the changes that are marked in red.



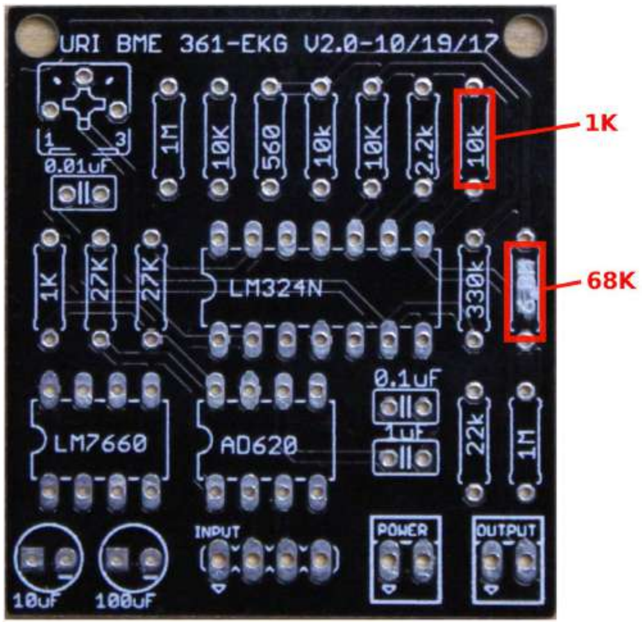
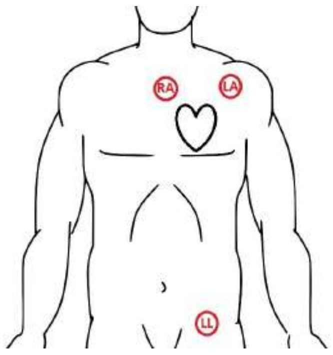
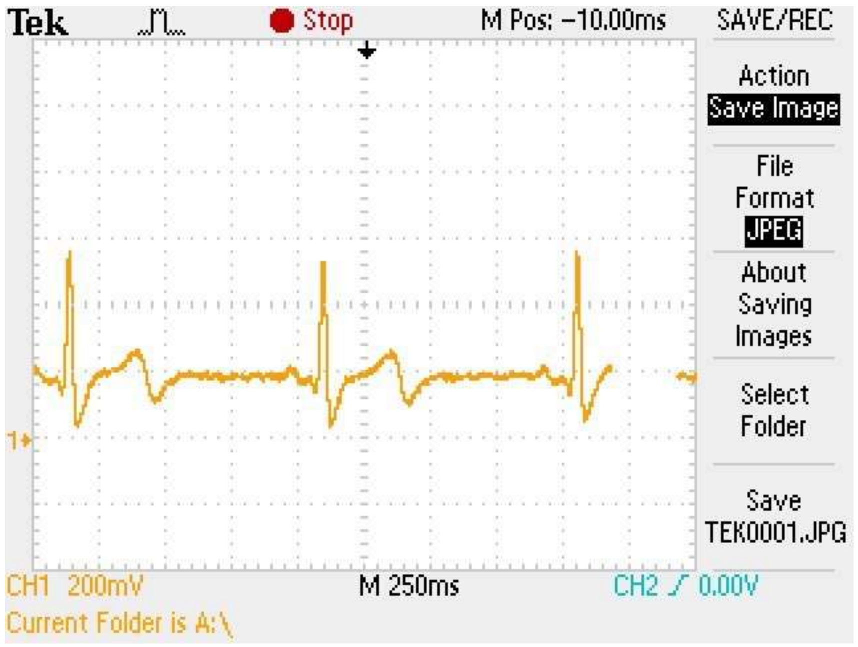


Figure 5.3: PCB images with and without components.

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 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 PCB and the other end to the oscilloscope.
3. Last, connect your battery to the PCB
4. You should now be able to see a functional ECG, similar to the one shown in figure 5.





(a) Recommended electrode (b) Oscilloscope showing waveform. placement.

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 PCB from Lab 5.

Software

2. 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-o frequency; while leaving the frequencies below the cut-o unchanged. High Pass (HP) Filters are the opposite of a LP - they eliminate all frequencies below the predetermined cut-o 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.
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, test your four new filters by use of signals from the function generator. Connect the function generator output to channel 1 of the oscilloscope before sending the signal to the PIC. Select a square wave. Adjust the offset and magnitude. Make sure the signal is within the range of the AD converter of the PIC, which is between 0V and 5V. Then, you can send the signal to AN0 (pin 2) of the PIC. The DA converter output should be connected to channel 2 of the oscilloscope.
5. The pushbutton will allow you to switch between filters. You should be able to view the effects of the filters. Use the square wave input to test the low-pass filter, high-frequency enhancement filter, and the median filter. Use the sinusoidal wave input to test the 60 Hz notch filter.

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

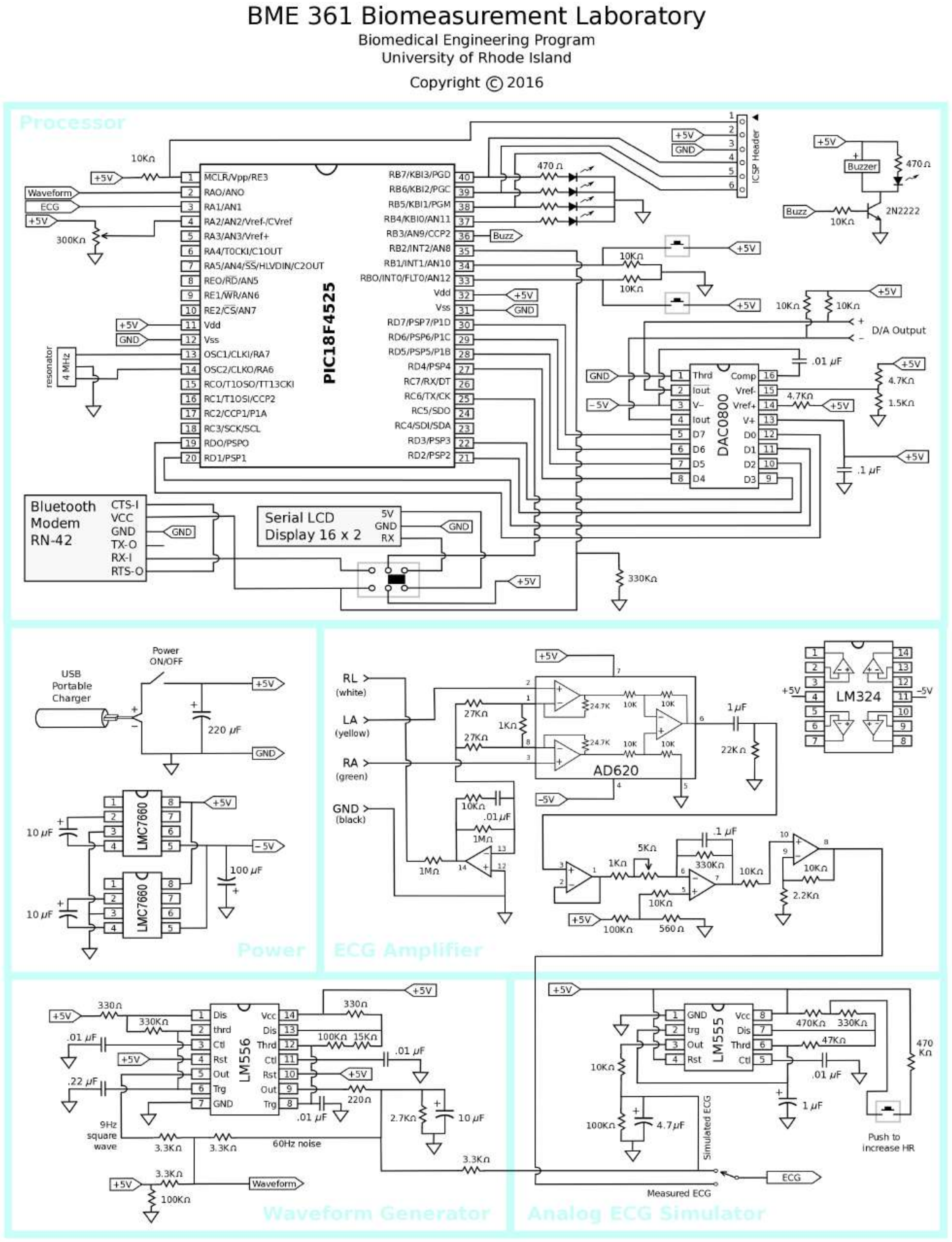


Figure 7.1: Schematic of completed circuit

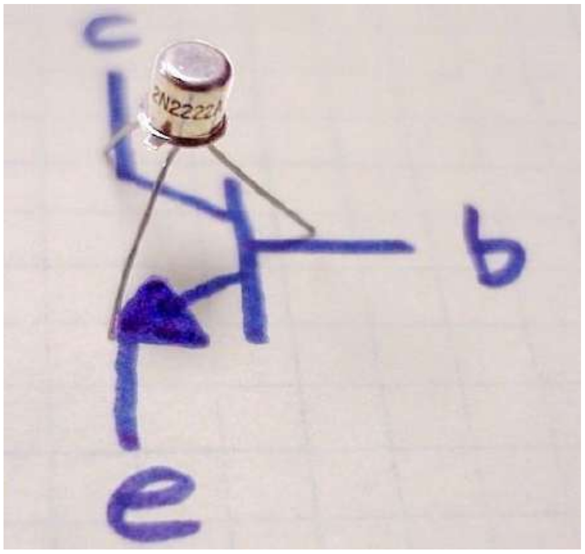


Figure 7.2: 2N2222 Transistor

PROCEDURE:

Hardware

1. Using your breadboard, add the highlighted section seen in the schematic to your previously existing 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 theMPLAB ICD3 programmer to the ICSP Header and to the PC, and download the program onto the PIC.
3. Disconnect theMPLAB ICD3 programmer from the ICSP Header.

TASKS:

1. Now that the output from your ECG PCB 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 to your heartbeat.
2. You can now test your circuit using an oscilloscope; connect channel 1 at the output of your ECG PCB, 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 (MOBD). 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 signal 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 Final Schematic.

Software

1. You will be modifying code to implement different modes.

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:
    1. ECG Simulation
    2. Echo
    3. Derivative
    4. LP Filter
    5. HP Filter
    6. Median Filter
    7. 60 Hz Notch Filter
    8. Heart Rate Meter
2. 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 A1: Installing MPLAB X

A.1.1 for Windows

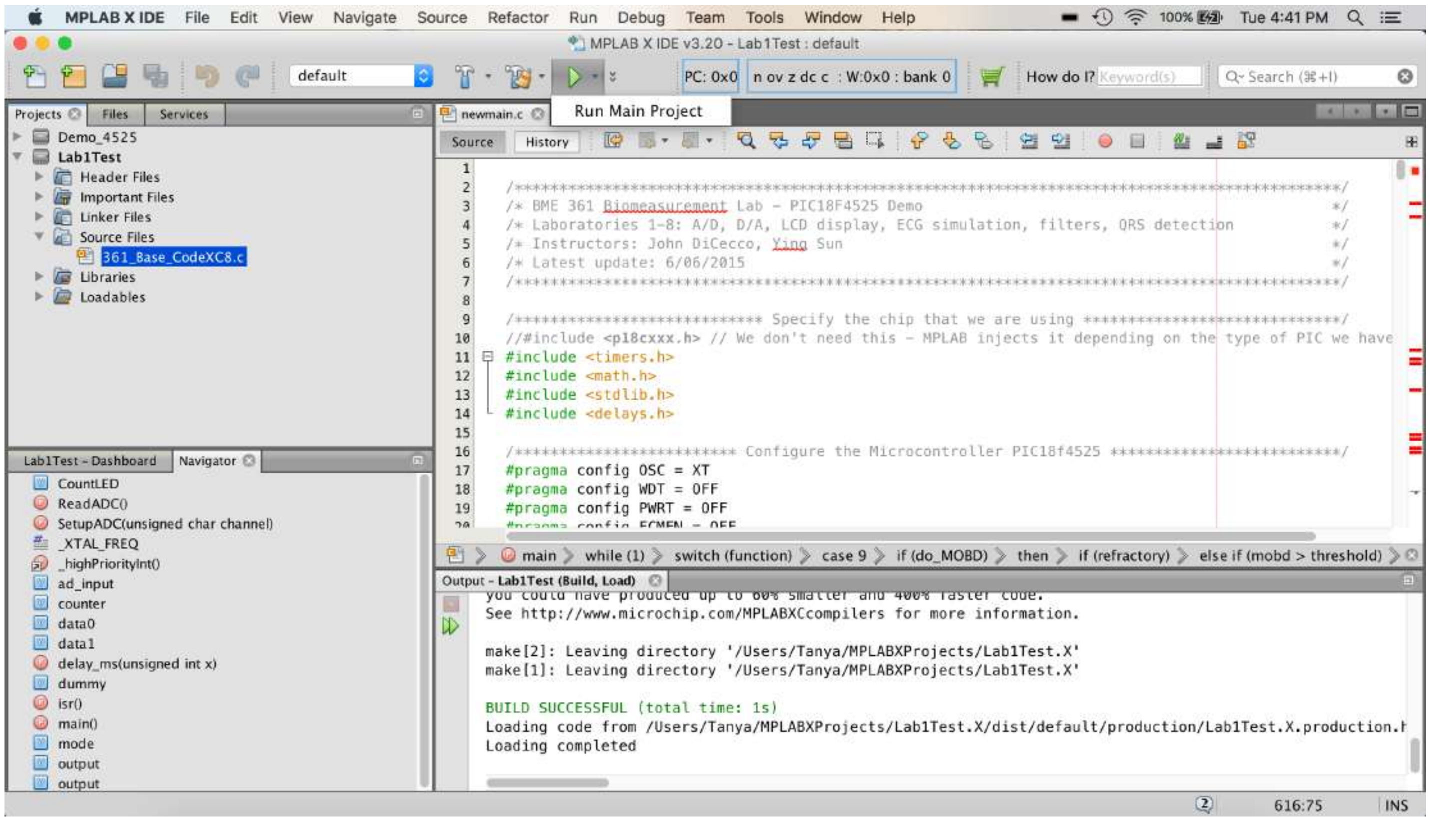
1. Install MPLAB X XC-8 Compiler (http://www.microchip.com/mplab/compilers) Scroll down >Downloads> Windows (x86/x64)>MPLAB XC8 Compiler vX.XX
2. Follow XC8 installer directions, When prompted: Free>check both for settings > install
3. Install MPLAB X IDE (http://www.microchip.com/mplab/mplab-x-ide) Scroll down >Downloads> Windows (x86/x64)> MPLAB X IDE v 3.XX
4. Follow MPLAB X installer instructions. When prompted: select both IDE and IPE programs > install

A.1.2 for OSX

1. Install MPLAB X XC-8 Compiler (http://www.microchip.com/mplab/compilers) Scroll down >Downloads> Mac (10.X)>MPLAB XC8 Compiler vX.XX
2. Follow XC8 installer directions, When prompted: Free>check both for settings > install
3. Install MPLAB X IDE (http://www.microchip.com/mplab/mplab-x-ide) Scroll down >Downloads> Mac(10.X)> MPLAB X IDE v 3.XX
4. Follow MPLAB X installer instructions. When prompted: select both IDE and IPE programs > install

Appendix A2: Creating a new project

1. Double-click on the MPLAB icon on the desktop. When it has opened, go to File > New Project.
2. Under choose project, Microchip Embedded > Standalone >Next.
3. Under Select Devices,Family: *Advanced 8-Bit MCU's (PIC 18)* , Device: *18F4525*
3. Select the tool desired, either ICD 3 or PICkit 3.
4. Under select compiler, choose the XC8, and select next.
5. Name your project something distinct and be aware that if you are using a lab computer, other sections may be naming their labs similar things. Don't just call it 'lab1'. It would be VERY WISE to also save your c code elsewhere, incase another section accidentally modifies or deletes your code. Select Finish.
6. Now you have a project with multiple files. Right click on Source Files > Add Existing Item, and find the C code you downloaded from the course webpage. Once added, you can double click on this file and it will display your 361 Base Code.
7. There are three icons, a hammer or 'build', a hammer + brush or "build and clean", and a play button "run". You can also find these under the Run tab from the navigation toolbar. At this point, you can build your program, and you should get a "BUILD SUCCESSFUL (total time: XXs)" in your output box. If your breadboard is correctly wired, powered and connected through a programer (Double check your power! *Have a T.A double check your circuit before you connect power*), you should also be able to run and program your device.



```
/*****************************************************************************************/
/* BME 361 Biomeasurement Lab - PIC18F4525 Demo                                          */
/* Laboratories 1-8: A/D, D/A, LCD display, ECG simulation, filters, QRS detection        */
/* Instructors: John DiCecco, Ying Sun                                                    */
/* Latest update: 6/06/2015                                                               */
/*****************************************************************************************/

/*************************** Specify the chip that we are using ***************************/
//#include <p18cxxx.h> // We don't need this - MPLAB injects it depending on the type of PIC we have
#include <timers.h>
#include <math.h>
#include <stdlib.h>
#include <delays.h>

/************************* Configure the Microcontroller PIC18f4525 *************************/
#pragma config OSC = XT
#pragma config WDT = OFF
#pragma config PWRT = OFF
#pragma config FCMEN = OFF
```

```
you could have produced up to 60% smaller and 400% faster code.
See http://www.microchip.com/MPLABXCcompilers for more information.

make[2]: Leaving directory '/Users/Tanya/MPLABXProjects/Lab1Test.X'
make[1]: Leaving directory '/Users/Tanya/MPLABXProjects/Lab1Test.X'

BUILD SUCCESSFUL (total time: 1s)
Loading code from /Users/Tanya/MPLABXProjects/Lab1Test.X/dist/default/production/Lab1Test.X.production.h
Loading completed
```

Figure A.2.1: Successful Build with source file and output displayed.

Appendix A3: Troubleshooting MPLAB X and programer

A.3.1 Build Unsuccessful

1. If you are using the given code, there may be something wrong with your project or your compiler.

A.3.2 Failed to connect

1. Make sure your battery is plugged in correctly, and all your “5V” are at this value. Use a multimeter to check.
2. Check to see that the ICSP connector pins are oriented in the correct direction (As mentioned previously, the arrow connects to pin 1 of the PIC.)
3. Unplug the USB extension from the computer and reinsert, try connecting again.
4. Double check PIC pins 37-40 of the PIC. These are the communication lines, and if wired incorrectly, your LED/ resistors may interfere with programming.
5. Check to make sure your PIC is not reversed on your breadboard, if it is, immediately disconnect the battery to prevent damaging the PIC and/or your breadboard.
6. Review your breadboard for loose connections, short circuits, broken components, and overheating.
7. Try replacing the PIC with a new one. In the case that this works, the PIC you were using is damaged.

Appendix B: Oscilloscopes 101

In order to view the results of your program on an oscilloscope, you must connect a DAC (Digital to Analog converter). Notice that pin 2 of the DAC is the output pin, and can be directly connected to the oscilloscope probe, and pin 4 connects to the reference.

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

1. 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

1. 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, nanovolt 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

1.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 o set) 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 o sets (like a simulated QRS complex) you will be able to determine where the signal is rising and falling against its DC o set. 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.

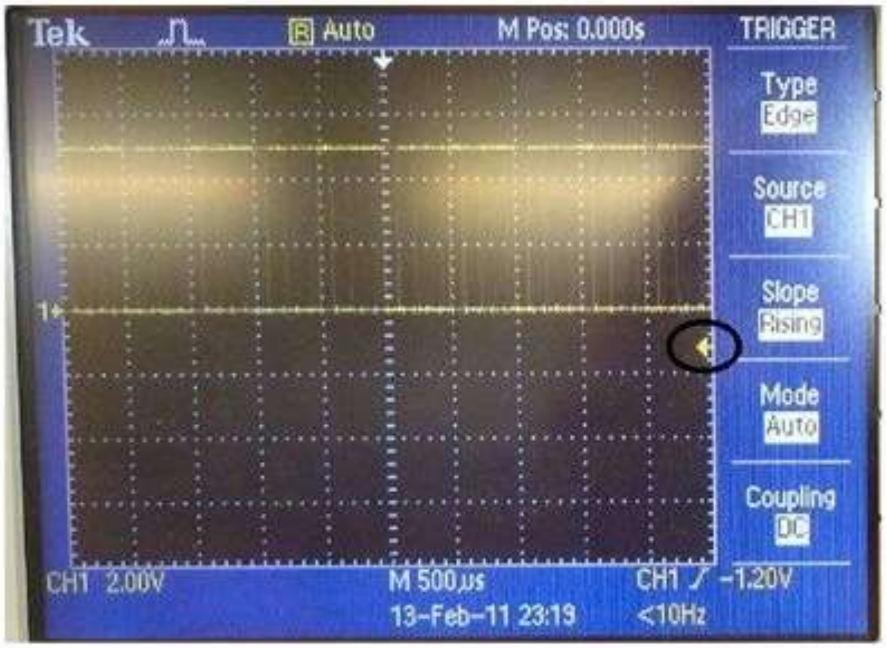


Figure B.1: Waveform appears smeared or blurred.

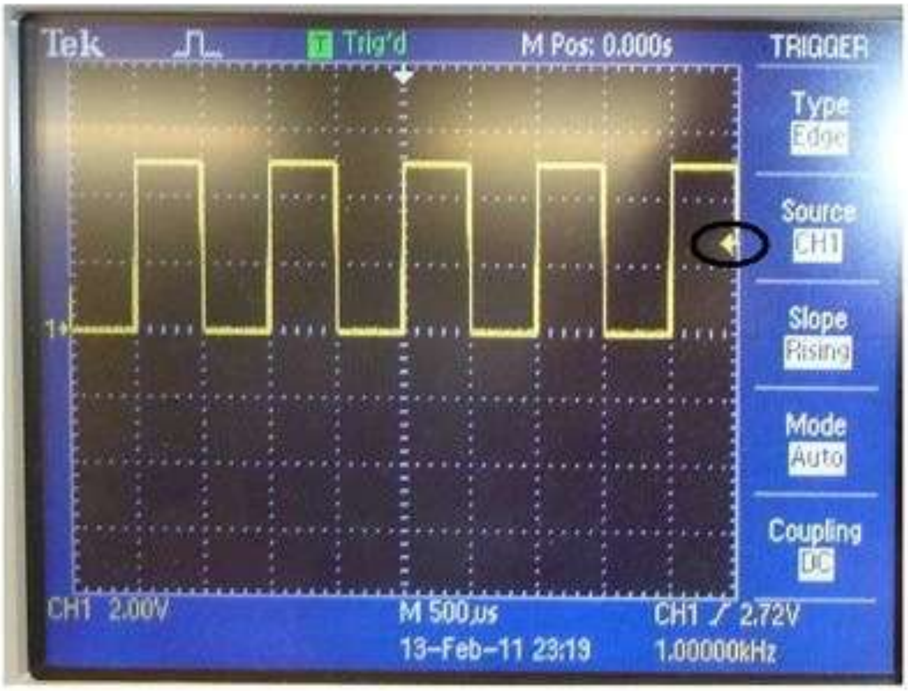


Figure B.2: Stable waveform.

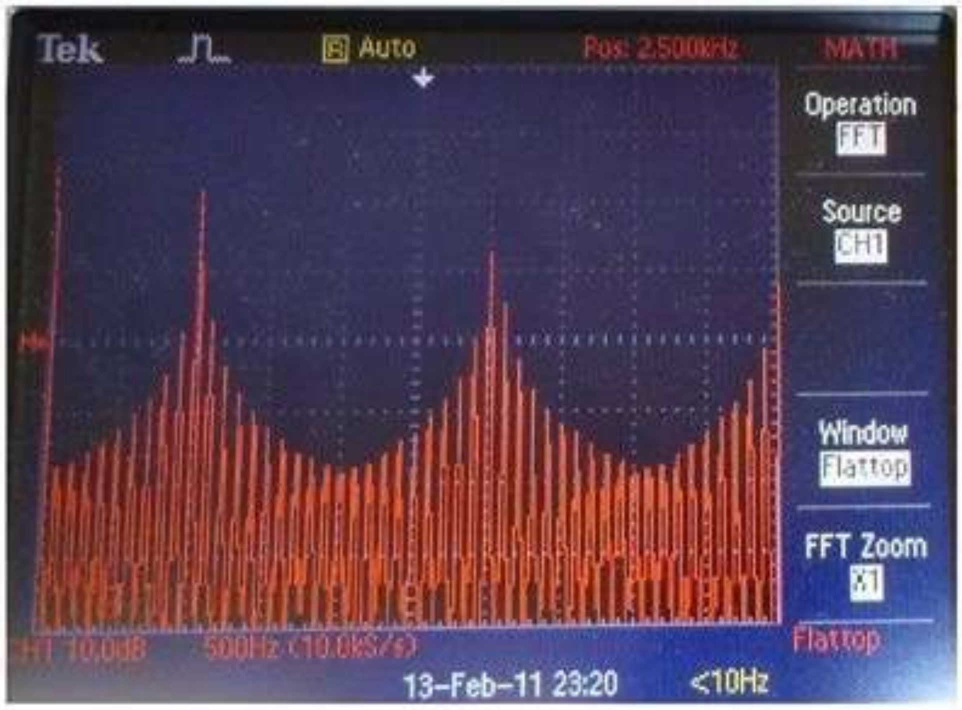


Figure B.3: Fourier transform of the signal in figure B.2. Notice that the time scale is set to 500 Hz per block and the first peak occurs at the second block, i.e. 1000 Hz. This is to be expected as the signal is a 1 kHz square wave. The other peaks are referred to as the harmonics and occur at ODD integer multiples of the fundamental.