Crystal Breath

Senior Design Capstone Project: Respiration Monitor

Nick Castro, Suong Do, Joe Duffy, Joe Levri, John Mullaney

5/9/2013

TABLE OF CONTENTS

I. INTRODUCTION

Crystal Breath spent the spring semester of 2013 building a respiration monitor. This device was designed to aid anesthesiologists in the operating room by offering additional visual confirmation that the patient is indeed breathing. The method used to detect respiration mimics a stethoscope technique used by anesthesiologists in times of uncertainty. The method our device emulates entails pressing a stethoscope to the neck of the patient and listening for the sound of air passing through the larynx. Initially, our design allowed for identical placement of the microphone; it was planned to attach the microphone used for sound detection just below the Adam's apple in the center of the neck, the same position the stethoscope head is placed.

The printed circuit board consisted of a microphone input, a low pass filter amplifier, a potentiometer used for gain control, a microcontroller capable of analog-digital and digital-analog conversion, a power amplifier, two seven segment LED displays, a row of LEDs for volume visualization, and a speaker jack output.

The placement of the microphone was the first deviation from our initial design. The filters and amplification circuits used did not reduce noise enough for placement on the neck. Touching the microphone to the neck resulted in an unacceptably low signal-to-noise ratio. This problem was unanticipated, as we had no trouble with excess noise earlier in the semester when the kit board's microphone was placed on the neck. The device currently works with the microphone positioned close to the mouth, touching the lips. It detects exhalation well because expelled air contacts the microphone and registers large volumes for relatively small volumes of air.

The second major deviation from initial design was the absence of audio playback. Audio playback was seen as an important aspect of the device in order to give the anesthesiologist raw data so that the judgment of safety of breathing stayed under the anesthesiologist's jurisdiction. Our device did not have functioning audio playback, which is a significant drawback; however, the other subsystems of our device functioned well enough to properly illustrate the patient's status.

A row of nine LEDs, three red, three yellow, and three green, light up in succession according to the volume detected by the microcontroller. Nine volume thresholds were established, each of them corresponding to a certain amplitude of breathing volume. The microcontroller interprets the signal, determines which threshold the breath falls in, and lights a certain number of LEDs. Two seven segment LEDs display a two digit number which represents the respiration rate in breaths per minute.

While the audio playback is not yet functional and the breath rate monitor needs further refinement, the row of LEDs responds to sound beautifully. The number of lit LEDs increases with louder volumes and updates instantaneously. The bar of light created by the LEDs swells and diminishes in real time with zero delay.

II. Detailed System Requirements

In order to have a functioning design, our device must update the row of LEDs as well as the seven segment LEDs at a rate fast enough to appear seamless and fluid. It must update the LED row by analyzing what analog input it is receiving and assigning a digital number to the input. Once the amplitude of volume is determined, the microcontroller must determine which threshold the amplitude falls in, and light the corresponding LEDs. The more intense the breath is, the more LEDs turn on. In addition, the microcontroller must be able to determine when a breath occurs, keep time until the next breath occurs, and extrapolate to give an instantaneous respiration rate.

III. Detailed Project Description

A. System Theory of Operation

The breath monitor was powered using a wall wart. The microphone of the device was supposed to detect the air rushing in and out the patient's neck. However, due to large amount of electrical noise from the device, we were not able to manipulate breath signals. Thus, we had to hold the microphone below the nose in order for the device to detect breath signals. The device can only detect the exhaling period because our electret microphone is unidirectional. These signals were transmitted to the DSP board. First, the signals went through a pre-amplifier and anti-aliasing filter to prevent overlapping samples. Our microcontroller sampled these amplified signals by performing ADC. Depending on the intensity of each breath signal, a certain number of LED will light up in succession. There are 9 LEDs total (3 red, 3 amber, and 3 green). The more LEDs that light up, the stronger the breath intensity is. Our device also has two seven-segment LEDs displaying the breath rate from 6 breaths/min to 24 breaths/min after each breath. The safe breath rate when one is at rest is between 8 breaths/min to 20 breaths/min. At the same time, the microcontroller also performs DAC so that we can output the analog breath signals to a speaker so that doctors can hear the patient breathing in real time. The doctor will then have enough information to decide whether or not the patient is in danger.

B. Block Diagram

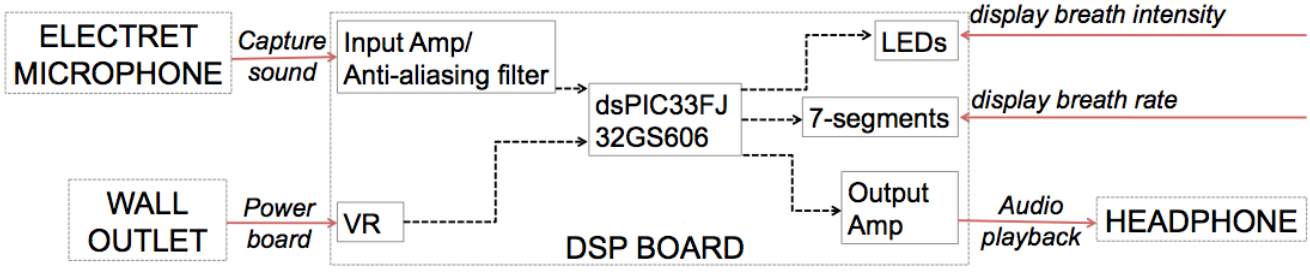


Figure 1: System Block Diagram

C. Attachment and Sound Capture

The first subsystem involves the attachment of the sound capture device to the subject and the sound capture device itself. The point of emphasis is that the device be noninvasive and sensitive enough to pick up the faint air movements of the subject's breath. The definitive choice for location of attachment is on the neck of the subject adjacent to where the Adam's apple would be on male patients. The apparatus for attachment consists of a microphone attached to a small breadboard with vias for the ground and signal connections or leads on the microphone. After attaching the microphone to the breadboard, wires of appropriate length were to be used to run the signal from the leads of the microphone to a headphone connecter that with the capability of plugging into any standard audio jack. Such an apparatus satisfies the subsystem requirements and would allow for interface with the processing breadboard. As far as securing the apparatus to the patient's neck, the most economically efficient method is to use adhesive tape to hold the microphone in place. Initial plans involved the use of MEMs microphones which proved to be difficult to work with in terms of circuitry and soldering. The process of selecting a microphone proved more difficult than initially anticipated; however, a unidirectional condenser microphone, proved to small enough to satisfy the subsystem requirements in terms of being noninvasive and sensitive. While such a microphone picks up sounds from around the room, the sounds are subdued and thresholds could easily be modified in other subsystems to ensure noise from the room does not adversely affect the results. Issues encountered in terms of electrical noise are usually not associated with this subsystem.

Figure 2: Electret Condenser Microphone

The second portion of sound capture subsystem involves the interface between the board and the microphone as well as amplification of the signal coming from the microphone. A standard 3.3 V SJ-3524-SMT audio jack, shown below in figure 5.3.2, is used for interface between the board and the audio connector from the microphone.

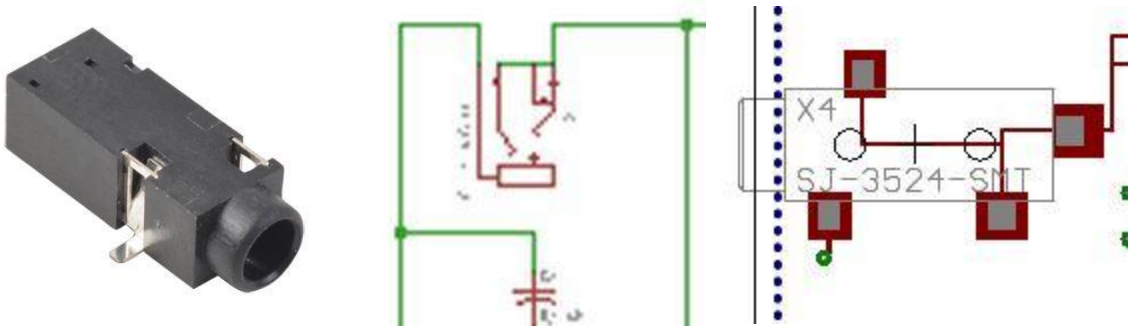


Figure 3: (from left to right) Standard SJ-3524-SMT package ; schematic symbol; package on breadboard

The audio jack then feeds the from the microphone signal into a circuit, which contains a preamplifier and an anti-aliasing filter, which prepares the signal for processing via the microcontroller. The preamplifier powers the jack and contains a 500 kOhm potentiometer for adjusting gain. The anti-aliasing filter is built around the MCP6024 device, a package containing 4 rail-to-rail, 10 MHz input operational amplifiers. The unamplified signal coming from the microphone lies within a range of 0 to 50 mV. The purpose of this circuit is to modify the voltage of the signal by expanding the range from 0 to 50mV to an enhanced range of 0 to 3.3 V.

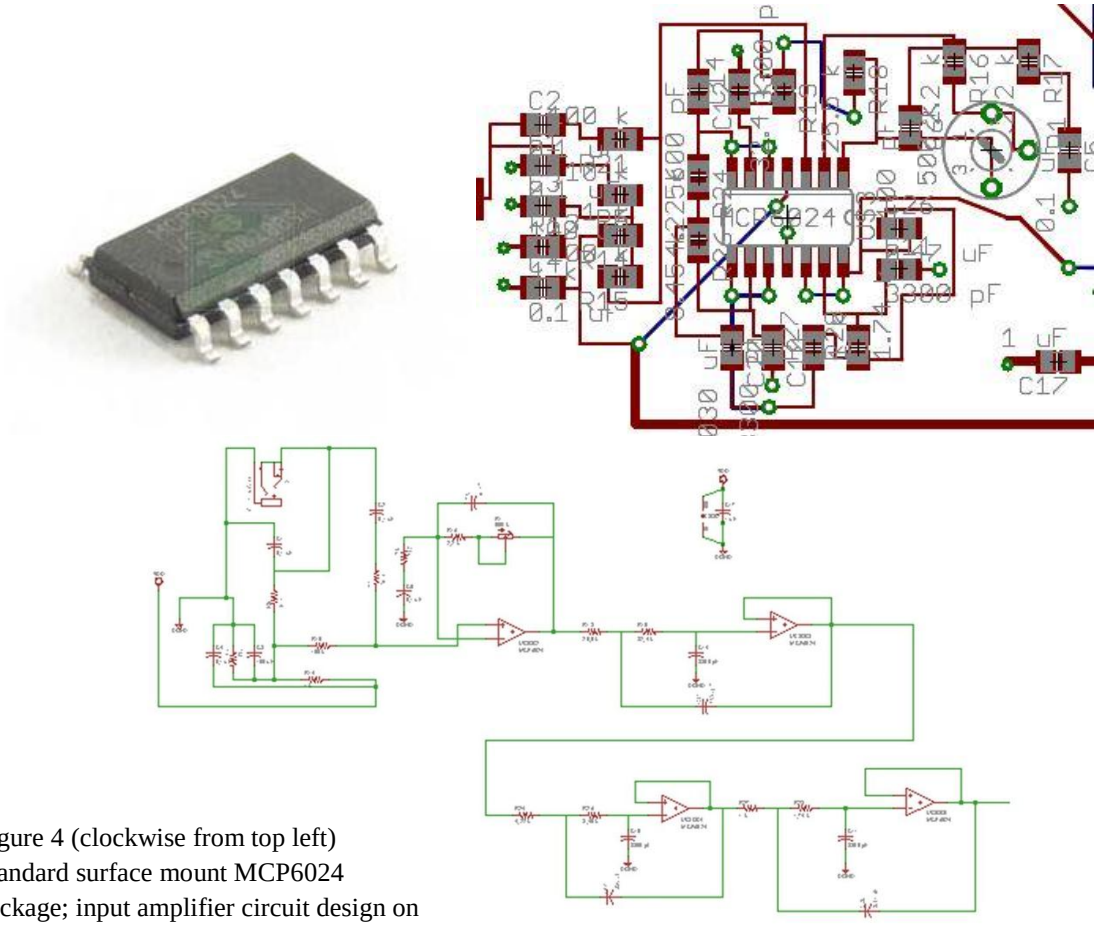


Figure 4 (clockwise from top left) Standard surface mount MCP6024 package; input amplifier circuit design on breadboard; amplifier circuit schematic.

D. Measuring Breath Volume/Intensity

One of the key objectives of the project is continuous breath detection. Measuring the intensity of breath in a particular subject is essential to detecting whether or not the patient is breathing. The second subsystem was designed to meet this requirement. This subsystem includes the microcontroller and 9 LEDs. The microcontroller was selected based on its conciseness. While we modeled our amplifier circuits after those in MPLAB Starter Kit for dsPIC Digital Signal Controllers, a different microcontroller was used in this project, namely the dsPIC33FJ32GS606. This change in hardware confines the signal processing elements of the board to the chip which enables the user to convert the analog signal to a digital signal and then either output the digital signal to certain subsystems or convert the signal back to an analog signal for audio playback. This subsystem involves an analog to digital conversion of the signal. The amplified input voltages are sampled by the microcontroller at a rate determined by the user and then assigned numerical values corresponding to the operating voltage range of the microcontroller, the values also being determined by the user. These values are then used to determine which, if any, of the LEDs will be turned on.

The 9 LEDs are chosen based on their color divided into three groups: red, amber, and green. Positioning, from left to right, three red LED's, followed by three yellow LED's, followed by three green LEDs on the breadboard adjacent to each other in a line, the LED's are used to represent a primitive version of a spectrum analyzer. The setup is shown below in Figure 5.4.1. Using MPLAB, a code written in C defines the threshold voltages for lighting up the LED's and tells the microcontroller to sample the input signal from the first subsystem at a frequency roughly equal to 500 Hz. The resulting sampling rate is 1 sample for every 2 thousandths of a second. Thus the sampling frequency of 500 Hz satisfies our desire for continuity as it renders the visualization fluid and coherent to naked eye. The based on the sampled voltage, the microcontroller then switches the LED's on or off given these directions. The leftmost red LED will be the first LED to light up in indicative of input voltages at or above 1.93 V, which is the defined threshold voltage for which breath occurs. Any sampled input voltage that is below this threshold will not induce the microcontroller to turn on any of the LEDs, which indicates that the no breath is being detected. Lack of detection in this case could be evidence that either the subject is not breathing or the subject is simply inhaling, a distinction that becomes imperative in the next subsystem. If the microcontroller samples the input and detects a voltage of 1.93 V, the microcontroller switches the left most red LED on. If the sampled input voltage is slightly greater, say 1.96 V, then the microcontroller switches the leftmost red LED as well as the adjacent red LED on. At an input voltage of 2 V, all of the red LEDs are on. This succession of lighting the LEDs continues from left to right as the input voltage increases, which is indicative of increasing volume of sound, which in turn indicates the exhale of the subject in the act of breathing. The lighting of the three amber LEDs in succession are indicative of input voltages between 2.02 and 2.08 V and the left most green LED will light up when the voltage surpasses 2.1 V. All LED's will be lit at or above 2.12 V. Rather than just having one LED light to indicate breath, this setup gives the observer visualization of the nature of the breath itself. The setup takes a complex problem and simplifies the solution; determining whether the person's breath is shallow or deep is just as immediately apparent to the observer as the answer to the question of whether or not the patient is breathing.

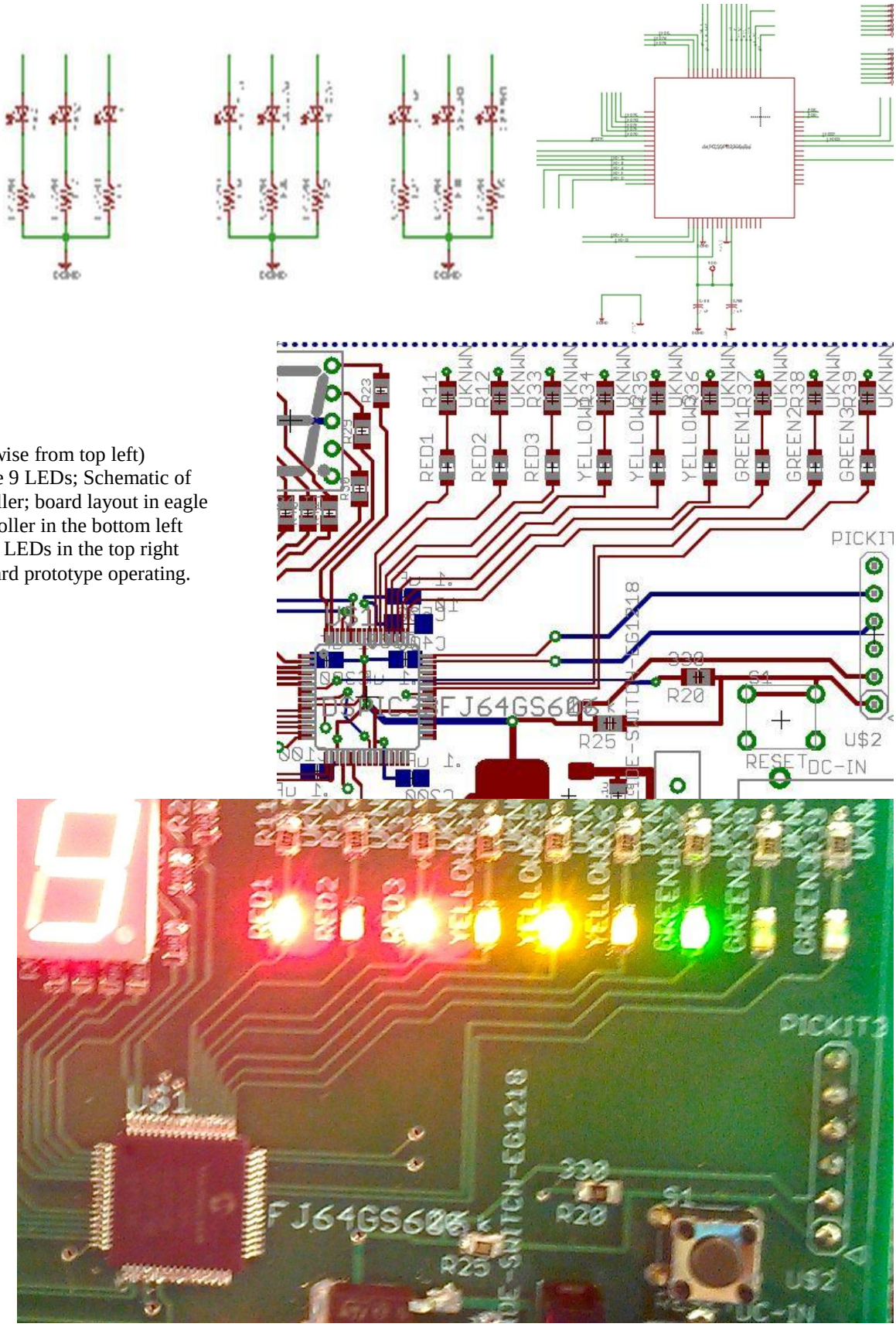


Figure 5 (clockwise from top left) Schematic of the 9 LEDs; Schematic of the microcontroller; board layout in eagle with microcontroller in the bottom left corner and the 9 LEDs in the top right corner; final board prototype operating.

E. Breath Rate Measurement

The subsystem for measuring breath rate contains two major components, namely the microcontroller and the two 7-segment LEDs. Much like the previous subsystem, the microcontroller samples the input signal, converts it from an analog signal to a digital signal, and then determines which segments on the 7-segment LEDs to turn on based on predetermined conditions from the user. The conditions for this subsystem are much different, however. This subsystem is programmed to detect the amount of time between two exhales of the subject and extrapolate the time frame to a minute by determining the equivalent a number of breaths per minute based on that time frame. The nature of the breath, while still important, is peripheral to the desire to know when the subject is breathing, more specifically, when the patient is exhaling and inhaling.

The microcontroller is programmed in MPLAB to sample the input signal. When the signal is below the threshold voltage of 1.93 V as defined in the previous section, the 7-segment LEDs remain idle or unresponsive. Once the threshold voltage is surpassed, that is, once the patient exhales and the exhale is detected by the sound capture subsystem. When the exhale is finished and inhalation begins, the voltage drops below threshold, and the microcontroller is programmed to initiate a counter to determine the time between two exhales. Upon detecting another exhalation, the microcontroller stops counting and multiplies the value of the counter by a specified value to determine the number of breaths per minute. Since the average breath rate of a human being is unlikely to surpass 99 breaths per minute and this particular project is tailored to its potential use with patients undergoing surgery, only two 7-segment LEDs were necessary for the display. The microcontroller turns on the correct segments of the LED to display breath rate in breaths per minute based on the counter and the multiplication factors determined by the programmer.

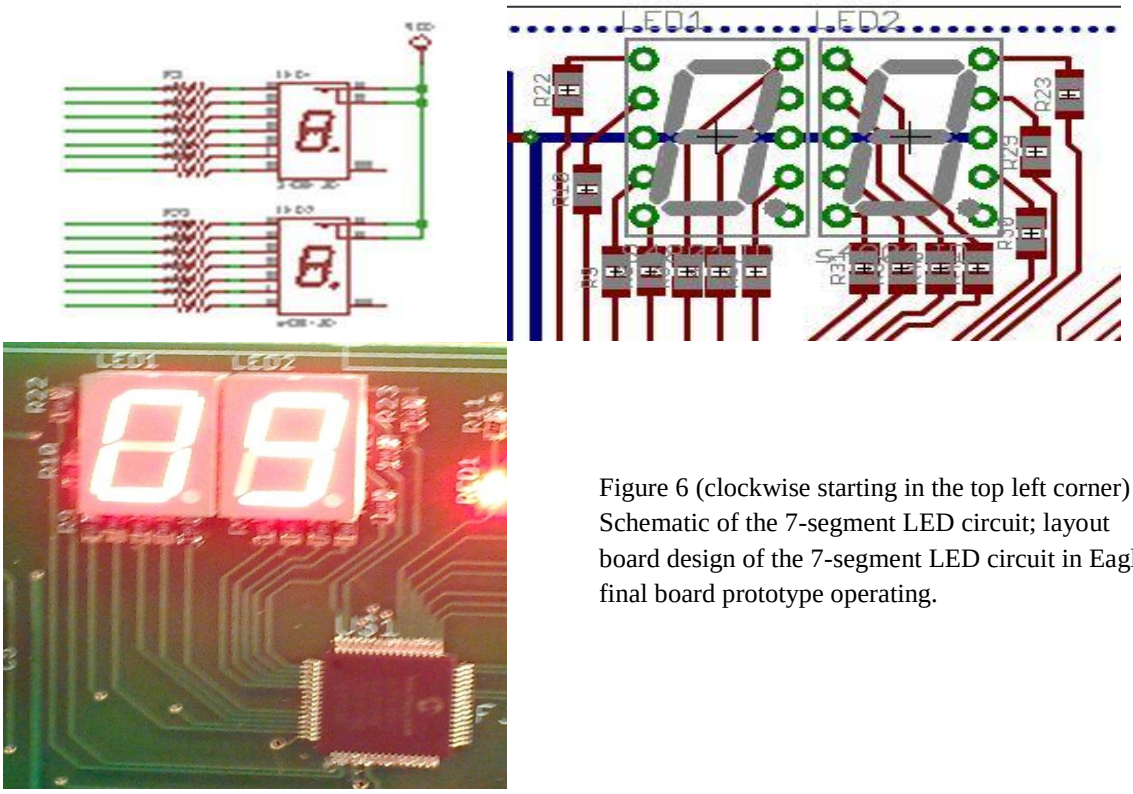


Figure 6 (clockwise starting in the top left corner) Schematic of the 7-segment LED circuit; layout board design of the 7-segment LED circuit in Eagle; final board prototype operating.

F. Audio Playback

This subsystem requires the conversion of a digital signal to an analog signal and then the amplification of the signal for playback. Thus, a microcontroller and an output amplifier circuit are necessary for playback of an audio signal. Following successful sound capture and conversion of the analog signal to a digital signal detailed in previous subsystems, the microcontroller is capable of replicating that signal provided the microcontroller is programmed to sample at a rate that satisfies the Nyquist Theorem. Sampling at 44.1 kHz is ideal, as such a signal sampling rate covers the entire range of human hearing. Using this rate, the microcontroller is capable of replicating the signal and reversing the analog to digital conversion it carried out in order to operate other subsystems. Once the signal is converted back to an analog signal, the signal can be sent to the output amplifier circuit. The output amplifier circuit is built around an LM4811 Boomer headphone amplifier. The package contains two operational amplifiers, one for each output channel for stereo output. The gain is digitally controlled by the microcontroller and can is usually set at a comfortable level for use with headphones. After the signal is amplified by this circuit, the signal is sent to the output audio jack, which is an SJ-3524-SMT identical to the one used in the sound capture subsystem.

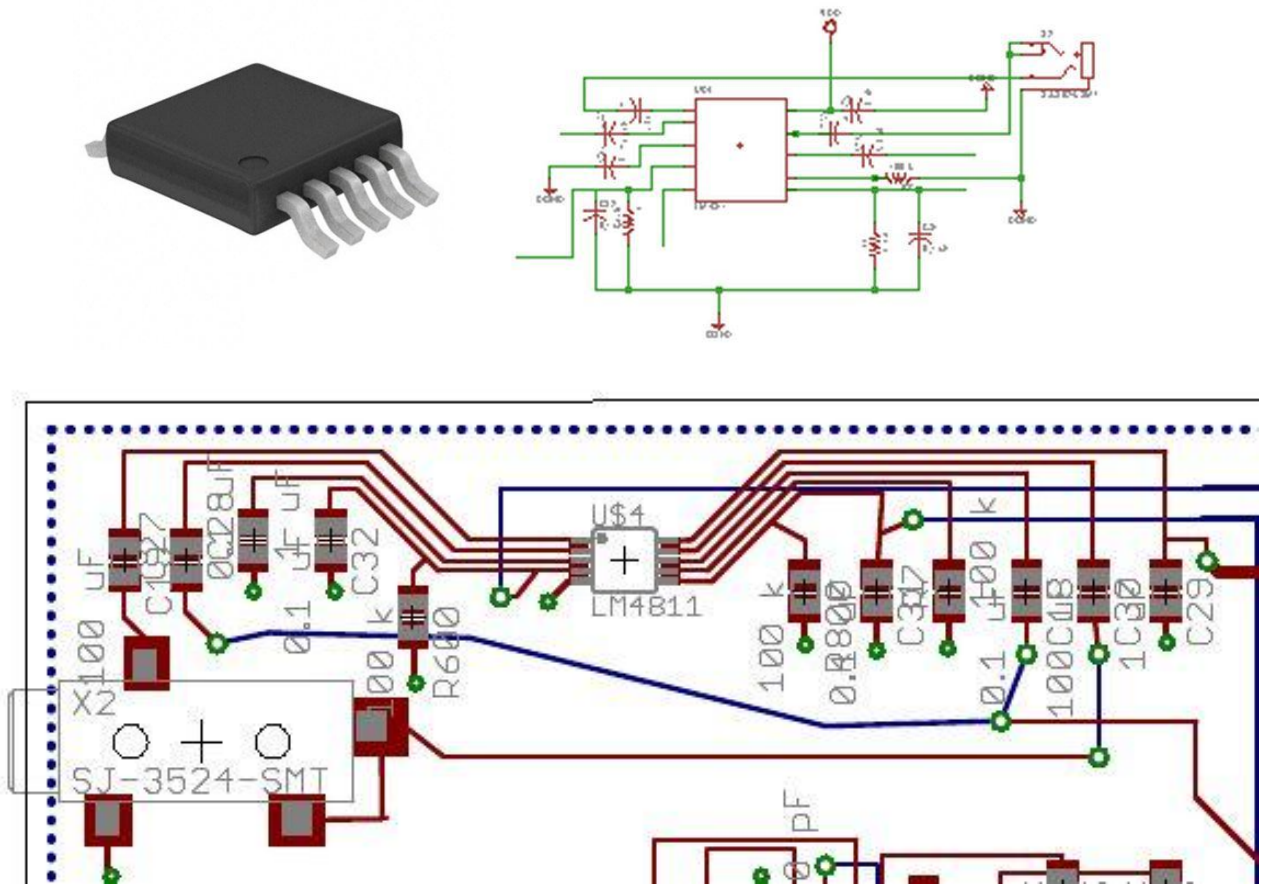


Figure 7 (clockwise from top left) Standard 10 pin LM4811 package; output amplifying circuit schematic; board layout in Eagle.

G. Powering the Board

The board is powered by a 10V walwart which is connected to a 3.3V voltage regulator. There is a physical switch to turn power on and off, and a reset button that turns the power off when it is pressed. A red LED turns on when the power is on, as a confirmation that the power subsystem is working (e.g. it's providing 3.3 V to the rest of the board).

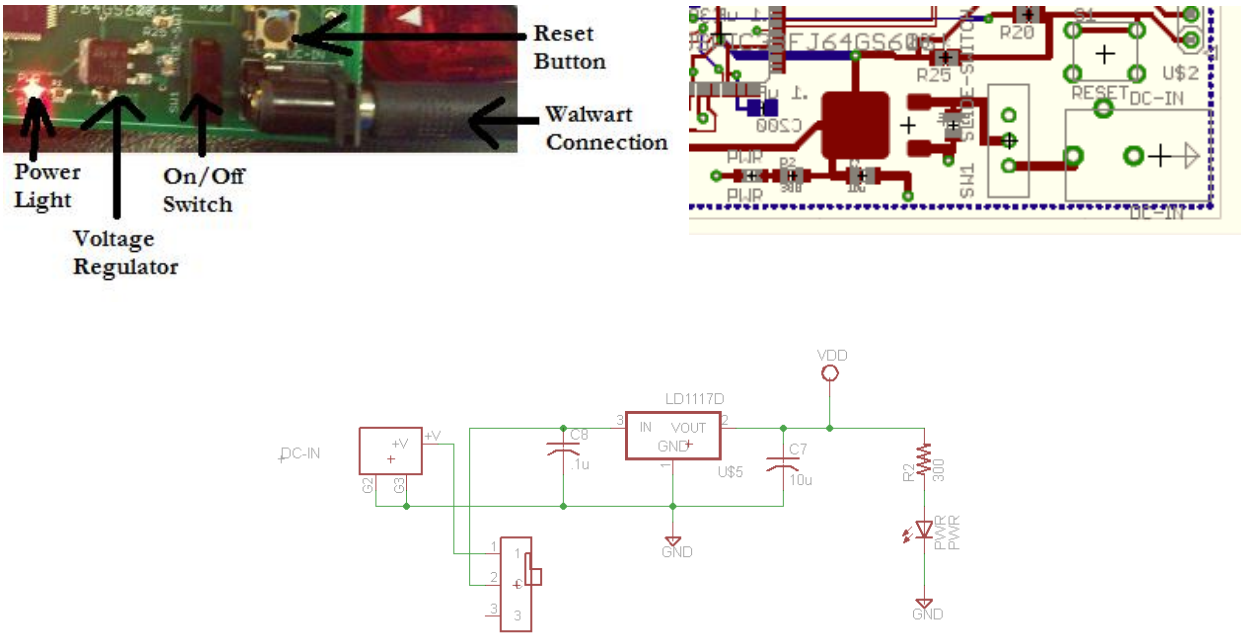


Figure 8: Power Subsystem-Photo, Board Layout, and Schematic

The voltage regulator is a L1117D 3.3V surface mount voltage regulator. This prevents the voltage from rising too high and damaging components on the board. The LED is red with a turn-on voltage of 2.0V that passes a current at forward bias of 5 mA. It came from the box of common parts that was shared between all senior design groups. The voltage coming out of the voltage regulator provides the 3.3V that powers the entire board. The high voltage on all the microcontroller pins is 3.3V. The switch is a black, plastic sliding switch with through-hole connections. There is a decoupling capacitor between Vdd and ground to protect the subsystem from transient effects when power is switched on or off.

Figure 9: L1117D 3.3V Voltage Regulator

IV. SYSTEM INTEGRATION TESTING

Once the board was delivered, all the components were soldered onto the board. Through-hole parts and some surface-mount parts were soldered using the soldering iron, while other surface-mount parts were attached to the board using solder paste and a hot air gun. When the parts were all attached to the board, we went to test it.

The power supply gave the board a voltage of 10 V, and the voltage regulator on the board scaled this down to 3.3V. The red LED power light turned on upon powering the board, and a voltage measurement with a voltmeter confirmed that the power across the voltage regulator was 3.3V. Upon powering the nine LEDs and the two 7-segment displays turned on. Using the Pickit3 programmer and debugger along with MPLABX as an IDE, we experimented turning different LEDs and portions of the 7-segments on and off. We confirmed that we were able to control which LEDs turned on and what number was displayed across the two 7-segment displays.

The next step was to test for the input of the microphone signal coming from the amplifier. After programming the microcontroller to perform an analog-to-digital conversion on the input signal and to turn on a specific number of LEDs corresponding to certain voltage levels, we put the microphone against a neck and breathed normally. The microcontroller didn't read any signal. Breaths were taken directly onto the microphone from the mouth, and oscilloscope readings were taken, but no breath signal was present coming from the amplifier.

Our amplifier malfunctioned. Our amplifier design employed the use of four cascading op-amps. This design was replicated from the dsPIC reference board we had been using and studying throughout the semester. Some of the resistors and capacitors we used had values similar, but not equal to those from dsPIC board design. The cascading effect caused the sensitivity to these changes in values to be high, and our amplifier saturated and never delivered a signal to the microcontroller. In retrospect, a simpler, non-cascading design would have been more prudent.

Since our amplifier circuit wasn't working, we connected the input pin of the microcontroller, the pin designed to receive the audio signal, and attached it to the output of the amplifier of the dsPIC reference board (note: All the dsPIC board did was supply an amplified audio signal. All signal processing and the analog-to-digital conversion occurred in our microcontroller on the board we designed.). After this we breathed into the microphone from the mouth, and stronger breaths lit up more LEDs, This indicated that the function of displaying the strength of the breath was working.

When the microphone was pressed against the neck, the breath signal was audible but the noise was louder. This was apparent in two ways. Firstly, listening to the signal by direct playback on the dsPIC board showed that the signal to noise ratio was very low. Also, there was no threshold voltage for the LEDs to show breath. At what would have been the right threshold voltage range for the breath, the noise caused all nine LEDs to remain permanently lit. This shows the need for either filtering out of the noise signal, or a more sensitive microphone capable of detecting the breath passing through the larynx and nothing else. The 7-segment displays showed breath frequency in breaths/minute, using the distance between two exhales. We confirmed the

accuracy of this measurement using stopwatches to calculate breath frequency while the board was doing the same thing. It showed that the board was accurate to ± 1 breath/minute. However the measurement took place after every consecutive breath. It didn't provide an average breath frequency over time but an instantaneous frequency, which is more valuable in the case of monitoring a patient under anesthesia.

We had four goals:

1. Record and amplify breath signal
2. Measure and display breath strength
3. Measure and display breath frequency
4. Playback the audio sound through outport connection

We accomplished goals 2 and 3 with our board, but not 1 and 4. We tested the board by programming the microcontroller and observing changes in what was displayed and when it was displayed. Use of a voltmeter and oscilloscope were also useful in testing. Determining the voltage thresholds of the LEDs was a trial and error experience using a variety of breaths at different strengths.

V. User's Guide

A. Custom Breath Monitor Layout

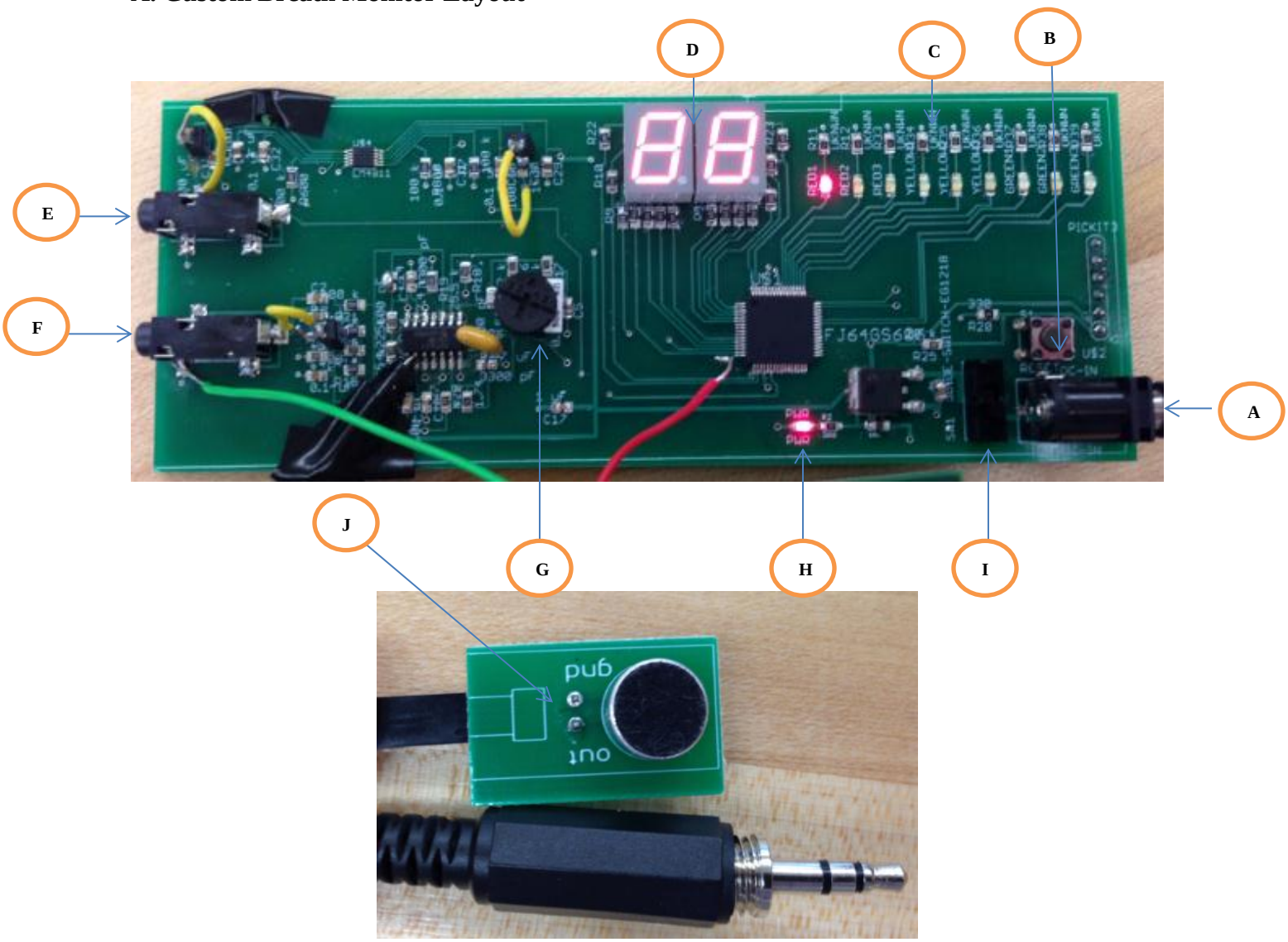


Figure 10: Board Layout and Microphone input

B. Breath Monitor Parts Description

Part Label	Part Name	Description
A	Power Connector	Connector for a 10 V wall wart. Used to power the device.
B	Reset Switch	Used to reset the device.
C	Respiration Intensity Indicators	LEDs that light in succession depending on the strength of the breath.
D	Breath Rate Display	Seven Segment LEDs that indicate the breathing rate of the user.
E	Headphone Jack**	Intended to be used for instant audio playback. Used to directly listen to the user's breathing.
F	Microphone Jack*	The input port for the microphone (Part J).
G	Microphone Gain	Twist to adjust the sensitivity of the microphone.
H	Power LED	An indicator that lets the user know when the board is both powered and the switch (Part I) has been moved to the ON position.
I	ON/OFF Switch	Controls whether the board is receiving power or not.
J	Microphone/Line-in	A microphone meant to be connected to the microphone jack (Part E).

*NOTE: The microphone jack circuitry on the custom board did not work properly during testing, so a second board will be used to provide the microphone input. This second board's layout is given below.
**NOTE: The headphone jack on this board did not function due to the microphone jack circuitry not working properly.

C. dsPIC Board Layout

Figure 11: dsPIC Board Layout

D. dsPIC Part Descriptions*

Part Label	Part Name	Description
A	USB Connector	A USB connector that is used to power the board.
B	Microphone Jack	The microphone jack that is used to provide the audio input. This takes the place of Part F in the custom board.
C	Power Indicator	An LED light that indicates the board is powered.

*NOTE: The dsPIC board and the custom board have and custom respiration monitor board have already been connected together in the necessary places.

E. Respiration Monitor Set-Up

1. Plug in the USB cord to the dsPIC board, and plug the other end of the cord into a computer's USB port.

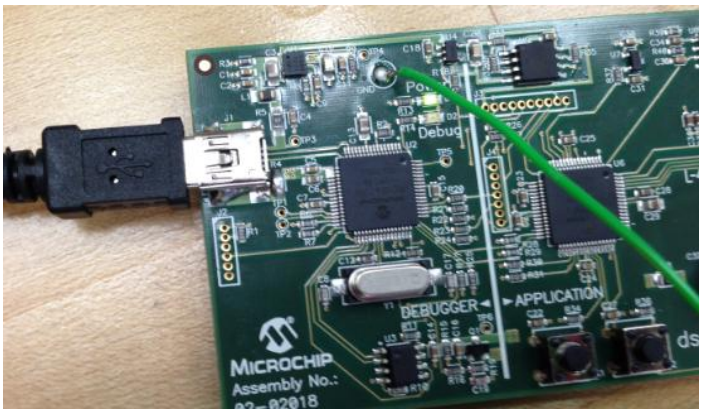


Figure 11: Powering the dsPIC board

2. Insert the microphone into the microphone jack on the dsPIC board.

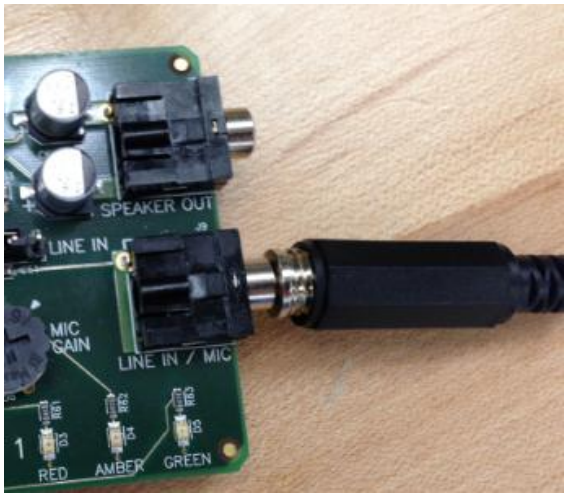


Figure 12: Microphone input to dsPIC board

3. Plug the wall wart power supply into the power connector of the custom board. Then switch the position of the power switch to the ON position. The power LED will turn red once power has been properly connected and the switch is in the ON position.

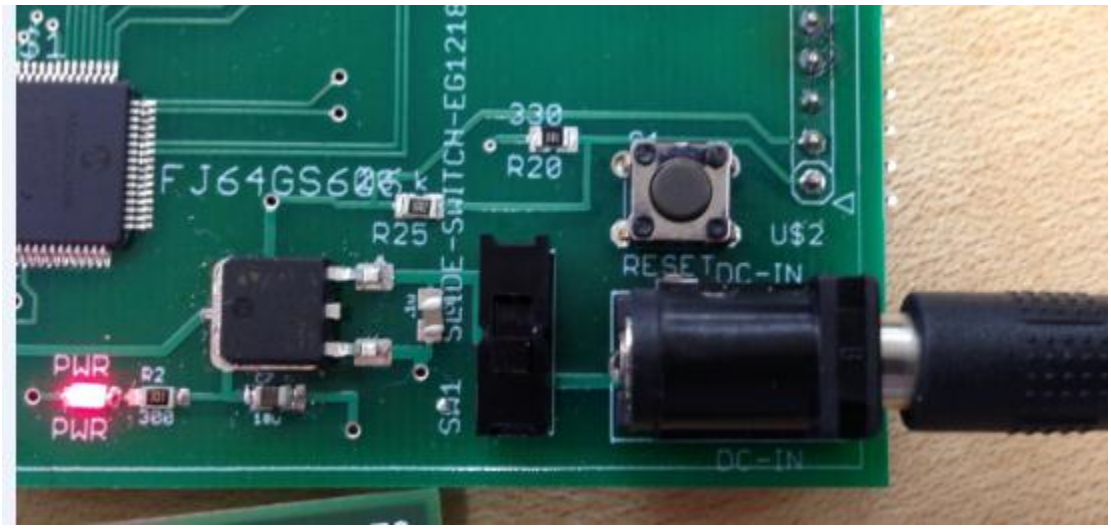


Figure13: Powering the custom board

4. Once both boards have been powered (both power LEDs are on) the device is ready for use.

F. Directions for use of Respiration Monitor

1. Follow instructions given in part E to set up and power the boards.
2. Hold the microphone very close to either your mouth of nose and breathe normally, with your breath directly hitting the center of the microphone.
3. With your first exhale, you should see the respiration intensity indicators (Part C on custom board) light in succession based on the intensity of the exhale. Note that the breath rate display (Part D on custom board) will not change with your first exhale.

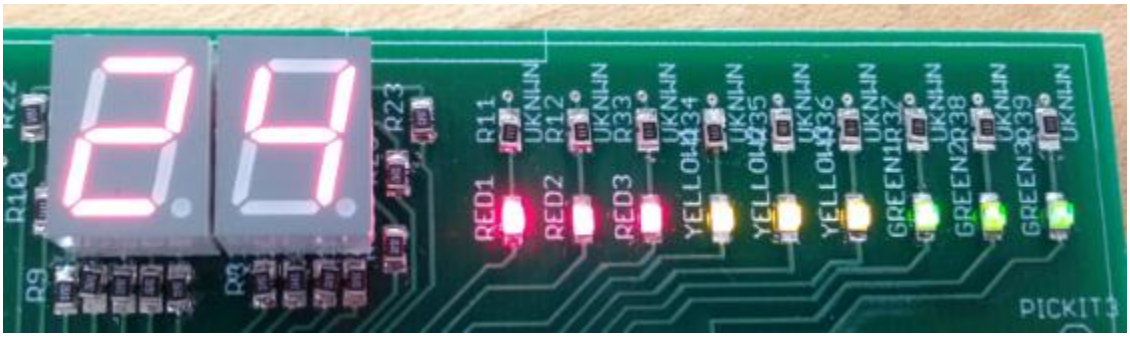


Figure 14: Breath Intensity Indicators and Breath Rate Display

4. Continue to breathe normally into the microphone. With each exhale, the breath rate display will change in accordance with your breathing rate. The longer the time between breaths, the lower the breath rate and vice versa. Note that the breath rate display will not change on your first exhale.

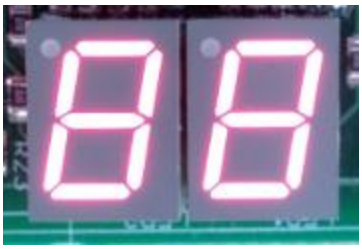


Figure 15: Breath Rate Display

5. Monitor how the breathing rate changes with each exhale. Be aware that the maximum breathing rate that the breathing rate display will show is 24 breaths per minute and the minimum breathing rate is 6 breaths per minute. The value increases or decreases in increments of 2 breaths per minute (i.e. the display will only show 6, 8, 10, 12, 14, 16, 18, 20, 22 or 24 breaths per minute). See the table below to reach a general understanding of what constitutes healthy breathing for different age groups.

Group	Age	Normal Rf (breaths/min)
Newborns	Up to 6 months old	30 - 60
Infants Toddlers	6 months to 5 years old	24 - 30
Children	6 to 12 years old	20 - 30
Adults	13 years old and up	12 - 20

G. Troubleshooting

1. The first step should be to be sure that both the dsPIC board and the custom respiration monitor board are powered correctly. Refer to Section E for instructions. Be sure that the power indicator is on for both boards. (The power LED is green for the dsPIC board and red for the custom board.)
2. *Both boards are powered properly but the breath intensity indicators are not lighting.*
    a. Make sure that the microphone is plugged into the microphone jack on the dsPIC board.
    b. Hold the microphone closer to your mouth/nose. The microphone will only pick up sounds very close to it.
    c. You can test to see that the microphone is still working properly by holding the microphone close to your mouth and speaking loudly into it.
3. *The sequential LEDs are working but the breathing rate display is stuck at either 6 breaths per minute or 24 breaths per minute.*
    a. Press the reset button. The breathing rate display should now read 88 (all segments on the display are ON). Start breathing normally into the microphone again.

b. If the breathing rate display continues to jump to 24 breaths per minute, you are most likely not holding the microphone close enough to the exhale source or not exhaling with enough intensity.

4. *The board is powered properly, the microphone is plugged in correctly, the microphone has been placed very close to the exhale source, but the sequential LEDs are not registering a breath.*
    a. There is most likely a broken connection between the dsPIC board and the custom board. Take the board to an available Crystal Breath team member for repairs.

VI. To-Market Design Changes

For this product to be a viable option on the market the microphone must be able to attach to the neck and measure a clear breath signal; there are already products that measure breath from the mouth. Our board was able to calculate breath strength and frequency when placed in front of the mouth, but when placed on the neck the signal to noise ratio was too low to determine strength of breath. Either we need to develop a high-quality bandpass filter that retains the breath signal and filters out the noise, or we need to invest in a better microphone, which can isolate the breath signal.

We would need a functioning amplifier to amplify the sound signal and supply it to the microcontroller. Our board's amplifier didn't work, but there are many different amplifier circuit designs in use which we could adapt or implement for use on our board. We would not design the cascading amplifier circuit unless we had the parts with the correct values to do so.

Audio playback would be a nice feature to add, so that the doctor can listen to the breath whenever he desires just as if he was wearing a stethoscope. Another useful feature, in addition to calculating breath rate, would be to calculate and store the breath rate for a specific amount of the time, so the doctor knows if the breath rate has been steady or not.

The deciding factor, and what would distinguish this product in the marketplace, is the ability to produce a clear breath signal from attachment to the neck.

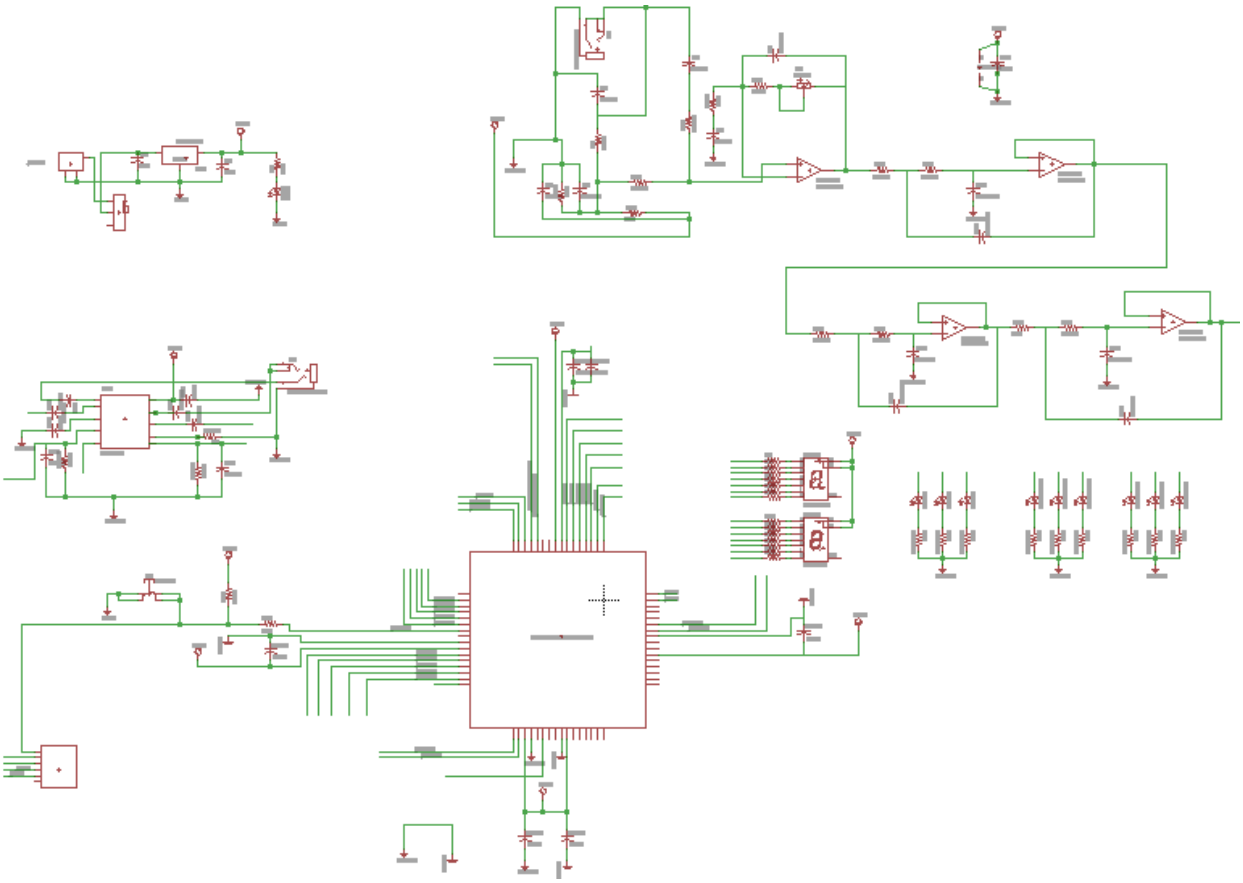
VII. Conclusions

Ultimately, we were not able to accomplish all of the goals we laid out for ourselves at the beginning of the semester. However, we were able to effectively design two of our main subsystems (breathing rate and breathing intensity). There is still plenty of room for improvement in our project. Instant audio playback of the microphone signal was a major goal of ours that went unaccomplished. Another problem we did not foresee was the sensitivity of the microphone. Unfortunately our microphone was not sensitive enough to overcome the electrical noise within the circuitry. This forced us to use the microphone in front of either the mouth or nose, rather than directly on the neck as we initially intended. The time spent attempting to develop a simple microphone circuit eventually caught up to us, preventing our group from having enough time to troubleshoot the problems with our board, such as the microphone amplifier and output headphone jack.

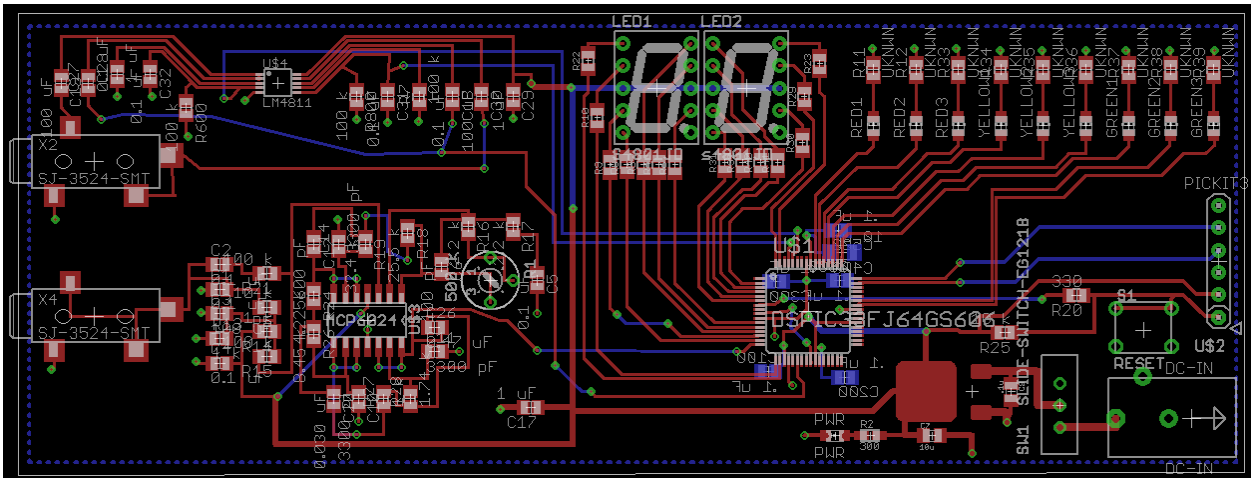
Our final product is not the most aesthetically pleasing product, but the functionality of our two working subsystems is effective. The breathing intensity indicators light sequentially with increasing breath intensity and the breathing rate display shows the breathing rate within 1 breath per minute.

VIII. APPENDIX

A. Schematic



B. Board Design



C. References

Microcontroller User's Guide
http://ww1.microchip.com/downloads/en/DeviceDoc/70591e.pdf
Timers
http://ww1.microchip.com/downloads/en/DeviceDoc/70205D.pdf
Interrupts and Interrupt Service Routines
http://ww1.microchip.com/downloads/en/DeviceDoc/70300C.pdf
High-Speed, 10-bit ADC
http://ww1.microchip.com/downloads/en/DeviceDoc/70000321g.pdf
Breath Rate Information
www.normalbreathing.com

D. Code

```
/*
* File: main.c
* Author: jduffy5
*
* Created on May 1, 2013, 6:51 PM
*/
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <p33FJ64GS606.h>
_FOSCSEL(FNOSC_FRC)
_FOSC(FCKSM_CSECMD & OSCIOFNC_ON)
_FWDT(FWDTEN_OFF)
_FPOR(FPWRT_PWR128)
_FICD(ICS_PGD1 & JTAGEN_OFF)
#define green1LED LATDbits.LATD1
#define green2LED LATDbits.LATD9
#define green3LED LATDbits.LATD8
#define yellow1LED LATDbits.LATD4
#define yellow2LED LATDbits.LATD3
#define yellow3LED LATDbits.LATD2
#define red1LED LATDbits.LATD7
#define red2LED LATDbits.LATD6
#define red3LED LATDbits.LATD5
#define LED1T LATBbits.LATB3
#define LED1TR LATBbits.LATB4
#define LED1BR LATBbits.LATB5
#define LED1B LATBbits.LATB1
#define LED1BL LATBbits.LATB2
#define LED1TL LATBbits.LATB6
#define LED1M LATBbits.LATB7
#define LED2T LATEbits.LATE3
#define LED2TR LATEbits.LATE4
```

```
#define LED2BR LATEbits.LATE6
#define LED2B LATGbits.LATG8
#define LED2BL LATGbits.LATG7
#define LED2TL LATGbits.LATG6
#define LED2M LATEbits.LATE7
//LED 1 Top RB3
//LED 1 Top Right RB4
//LED 1 Bot Right RB5
//LED 1 Bot RB1
//LED 1 Bot Left RB2
//LED 1 Top Left RB6
//LED 1 Middle RB7
//LED 2 Top RE3
//LED 2 Top Right RE4
//LED 2 Bot Right RE6
//LED 2 Bot RG8
//LED 2 Bot Left RG7//LED 2 Top Left RG6
//LED 2 Middle RE7
int ADC_Readout;
int counts = 0;
int counts_between_breaths;
int detect_first_breath=0;
int silence_after_breath = 0;
double secs_between_breaths;
double breaths_per_sec=0;
int silence_counts;
/*
*
*/
void __attribute__((__interrupt__,no_auto_psv)) _ADCP0Interrupt ()
{ IFS6bits.ADCP0IF=0;
counts = counts + 1;
ADC_Readout = ADCBUF0;
if (ADC_Readout > 655)
{
red1LED = 1;
red2LED = 1;
red3LED = 1;
yellow1LED = 1;
yellow2LED = 1;
yellow3LED = 1;
green1LED = 1;
green2LED = 1;
green3LED = 1;
}
else if (ADC_Readout > 650)
{
red1LED = 1;
red2LED = 1;
```

```
red3LED = 1;
yellow1LED = 1;
yellow2LED = 1;
yellow3LED = 1;
green1LED = 1;
green2LED = 1;
green3LED = 0;
}
else if (ADC_Readout > 645)
{
red1LED = 1;
red2LED = 1;
red3LED = 1;
yellow1LED = 1;
yellow2LED = 1;
yellow3LED = 1;
green1LED = 1;
green2LED = 0;
green3LED = 0;
}else if (ADC_Readout > 636)
{
red1LED = 1;
red2LED = 1;
red3LED = 1;
yellow1LED = 1;
yellow2LED = 1;
yellow3LED = 1;
green1LED = 0;
green2LED = 0;
green3LED = 0;
}
else if (ADC_Readout > 627)
{
red1LED = 1;
red2LED = 1;
red3LED = 1;
yellow1LED = 1;
yellow2LED = 1;
yellow3LED = 0;
green1LED = 0;
green2LED = 0;
green3LED = 0;
}
else if (ADC_Readout > 618)
{
red1LED = 1;
red2LED = 1;
red3LED = 1;
yellow1LED = 1;
```

```
yellow2LED = 0;
yellow3LED = 0;
green1LED = 0;
green2LED = 0;
green3LED = 0;
}
else if (ADC_Readout > 609)
{
red1LED = 1;
red2LED = 1;
red3LED = 1;
yellow1LED = 0;
yellow2LED = 0;
yellow3LED = 0;
green1LED = 0;
green2LED = 0;
green3LED = 0;
}
else if (ADC_Readout > 600)
{
red1LED = 1;
red2LED = 1;
red3LED = 0;
yellow1LED = 0;
yellow2LED = 0;
yellow3LED = 0;green1LED = 0;
green2LED = 0;
green3LED = 0;
}
else
{
red1LED = 1;
red2LED = 0;
red3LED = 0;
yellow1LED = 0;
yellow2LED =0;
yellow3LED = 0;
green1LED = 0;
green2LED =0;
green3LED = 0;
if (detect_first_breath == 1){
silence_counts = silence_counts + 1;
if (silence_counts > 450)
{
silence_after_breath = 1;
}
}
}
if (ADC_Readout > 618) {
```

```
if (detect_first_breath ==0){
counts = 0;
detect_first_breath = 1;
}
if (silence_after_breath == 1){
counts_between_breaths = counts;
secs_between_breaths = 0.0022 * counts_between_breaths;
breaths_per_sec = 60/secs_between_breaths;
detect_first_breath = 0;
silence_counts = 0;
silence_after_breath = 0;
//breaths_per_sec = 13;
if (breaths_per_sec > 24){
LED1T = 0;
LED1TL = 1;
LED1TR = 0;
LED1BR = 1;
LED1BL = 0;
LED1B = 0;
LED1M=0;
LED2T = 1;
LED2TL = 0;
LED2TR = 0;
LED2BR = 0;
LED2BL = 1;
LED2B = 1;
LED2M = 0;
}
else if (breaths_per_sec >= 22){LED1T = 0;
LED1TL = 1;
LED1TR = 0;
LED1BR = 1;
LED1BL = 0;
LED1B = 0;
LED1M=0;
LED2T = 0;
LED2TL = 1;
LED2TR = 0;
LED2BR = 1;
LED2BL = 0;
LED2B = 0;
LED2M = 0;
}
else if (breaths_per_sec >= 20){
LED1T = 0;
LED1TL = 1;
LED1TR = 0;
LED1BR = 1;
LED1BL = 0;
```

```
LED1B = 0;
LED1M=0;
LED2T = 0;
LED2TL = 0;
LED2TR = 0;
LED2BR = 0;
LED2BL = 0;
LED2B = 0;
LED2M = 1;
}
else if (breaths_per_sec >=18){
LED1T = 1;
LED1TL = 1;
LED1TR = 0;
LED1BR = 0;
LED1BL = 1;
LED1B = 1;
LED1M=1;
LED2T = 0;
LED2TL = 0;
LED2TR = 0;
LED2BR = 0;
LED2BL = 0;
LED2B = 0;
LED2M = 0;
}
else if (breaths_per_sec >=16){
LED1T = 1;
LED1TL = 1;
LED1TR = 0;
LED1BR = 0;LED1BL = 1;
LED1B = 1;
LED1M=1;
LED2T = 0;
LED2TL = 0;
LED2TR = 1;
LED2BR = 0;
LED2BL = 0;
LED2B = 0;
LED2M = 0;
}
else if (breaths_per_sec >=14){
LED1T = 1;
LED1TL = 1;
LED1TR = 0;
LED1BR = 0;
LED1BL = 1;
LED1B = 1;
LED1M=1;
```

```
LED2T = 1;
LED2TL = 0;
LED2TR = 0;
LED2BR = 0;
LED2BL = 1;
LED2B = 1;
LED2M = 0;
}
else if (breaths_per_sec >=12){
LED1T = 1;
LED1TL = 1;
LED1TR = 0;
LED1BR = 0;
LED1BL = 1;
LED1B = 1;
LED1M=1;
LED2T = 0;
LED2TL = 1;
LED2TR = 0;
LED2BR = 1;
LED2BL = 0;
LED2B = 0;
LED2M = 0;
}
else if (breaths_per_sec >=10){
LED1T = 1;
LED1TL = 1;
LED1TR = 0;
LED1BR = 0;
LED1BL = 1;
LED1B = 1;LED1M=1;
LED2T = 0;
LED2TL = 0;
LED2TR = 0;
LED2BR = 0;
LED2BL = 0;
LED2B = 0;
LED2M = 1;
}
else if (breaths_per_sec >=8) {
LED1T = 0;
LED1TL = 0;
LED1TR = 0;
LED1BR = 0;
LED1BL = 0;
LED1B = 0;
LED1M=1;
LED2T = 0;
LED2TL = 0;
```

```
LED2TR = 0;
LED2BR = 0;
LED2BL = 0;
LED2B = 0;
LED2M = 0;
}
else
{
LED1T = 0;
LED1TL = 0;
LED1TR = 0;
LED1BR = 0;
LED1BL = 0;
LED1B = 0;
LED1M=1;
LED2T = 0;
LED2TL = 0;
LED2TR = 1;
LED2BR = 0;
LED2BL = 0;
LED2B = 0;
LED2M = 0;
}
}
}
}
void init_PWM(void)
{
PTCON2bits.PCLKDIV=0b110; //PWM input clk devider =64
PHASE1 = 0xFFFF; // Timer1 period=max, this is just
convenient for validationPDC1 = 0x7FFF; // Duty Cycle is 50%
IOCON1bits.PENH = 1; // PWM1H is controlled by PWM module
IOCON1bits.PENL = 1; // PWM1H is controlled by PWM module
IOCON1bits.POLH = 0; // Drive signals are active-high
IOCON1bits.PMOD=1; // PWM pair works in Independent Mode
PWMCON1bits.ITB = 1; // Select Independent Timebase mode
}
void init_ADC(void)
{
ADCONbits.FORM = 0; //Integer data format
ADCONbits.EIE = 0; //Early Interrupt disabled
ADCONbits.ORDER = 0; //Convert even channel first
ADCONbits.SEQSAMP = 0; //Select simultaneous sampling
ADCONbits.ADCS = 5; //ADC clock = FADC/6 = 120MHz / 6 = 20MHz
ADPCFGbits.PCFG0 = 0; //select CH0 as analog pin
ADPCFGbits.PCFG1 = 0; //select CH1 as analog pin
IFS6bits.ADCP0IF = 0; //Clear ADC Pair 0 interrupt flag
IPC27bits.ADCP0IP = 5; //Set ADC Pair 0 interrupt priority
IEC6bits.ADCP0IE = 1; //Enable the ADC Pair 0 interrupt
```

```
ADSTATbits.P0RDY = 0; //Clear Pair 0 data ready bit
ADCPC0bits.IRQEN0 = 1; //Enable ADC Interrupt pair 0
ADCPC0bits.TRGSRC0 = 4; //ADC Pair 0 triggered by PWM1 Trigger
TRGCON1bits.DTM=1; //dual trigger mode
TRIG1bits.TRGCMP=0; //Primary trig compare value
STRIG1bits.STRGCMP=0xFFF; //secondary trig compare value
TRGCON1bits.TRGDIV = 0; // Trigger generated every PWM cycle
TRGCON1bits.TRGSTRT = 0; // enable Trigger generated after 0 PWM cycles
TRIG1 = 1; // Trigger compare value
}
int main(int argc, char** argv) {
/*Configure Oscillator to operate the device at 40MHz.
* Fosc= Fin*M/(N1*N2), Fcy=Fosc/2
* Fosc= 7.37M*43/(2*2)=80Mhz for 7.37M input clock */
// Configure PLL prescaler, PLL postscaler, PLL divisor
PLLFBD=41; // M = PLLFBD + 2
CLKDIVbits.PLLPOST=0; // N1 = 2
CLKDIVbits.PLLPRE=0; // N2 = 2
__builtin_write_OSCCONH(0x01); // New Oscillator FRC w/ PLL
__builtin_write_OSCCONL(0x01); // Enable Switch
while(OSCCONbits.COSC != 0b001);
while(OSCCONbits.LOCK != 1) {};
ACLKCONbits.FRCSEL = 1; // FRC provides input for Auxiliary PLL (x16)
ACLKCONbits.SELACLK = 1; // Auxiliary Ocillator provides clock source for PWM
& ADC
ACLKCONbits.APSTSCLR = 7; // Divide Auxiliary clock by 1
ACLKCONbits.ENAPLL = 1; // Enable Auxiliary PLLwhile(ACLKCONbits.APLLCK != 1);
/*The FRC clock runs at 7.37 MHz. While setting the aux clock,
there is an automatic 16x multiplication factor, which would
result in a 120 MHz aux clock, so we divided that by 8 to
get an aux clock of 15 MHz*/
TRISD = 0;
TRISBbits.TRISB1 = 0;
TRISBbits.TRISB2 = 0;
TRISBbits.TRISB3 = 0;
TRISBbits.TRISB4 = 0;
TRISBbits.TRISB5 = 0;
TRISBbits.TRISB6 = 0;
TRISBbits.TRISB7 = 0;
TRISBbits.TRISB9 = 0;
TRISE = 0;
TRISG = 0;
init_PWM(); // PWM Setup
init_ADC();
PTCONbits.PTEN = 1; // Enable the PWM
ADCONbits.ADON = 1; // Enable the ADC
while(1);
}
```