PIC Controlled Two-Band Stereo Audio Equalizer

By

Tim Brown

Senior Project

Electrical Engineering Department

California Polytechnic State University

San Luis Obispo

Fall 2009

Table of Contents

List of Tables

List of Figures

Acknowledgements

First and foremost, I would like to thank Dr. Dennis Derickson for being my advisor for this project. Without Dr. Derickson's input and support, this project would not have been controlled by a PIC; I am greatly indebted to Dr. Derickson's valuable input. I also would like to thank Dr. Gary Granneman for teaching me all about analog filters. In the spring of 2009, I took EE – 425 *Analog Filters*, and this class turned out to be the springboard and backbone for my senior project. Not only did this class teach me all about filters, but the lab portion of the class (EE 455) taught me how to proficiently use a spectrum analyzer, which I relied on heavily in this project.

I also would like to thank Jamie Carmro, who allowed me to check-out and take home test instruments (power supplies and multi-meters), when I needed them most. I also would like to thank Henry C. Ureh (my first E.E. brotha), and all my other numerous friends and classmates who supported me and added advice while I built the project. Last but not least, I would like to thank my parents who installed in me the value of education; without their guidance, I would not be where I am at today.

Tim Brown
San Luis Obispo, Ca
December 2009

Abstract

The purpose of this project is to build a PIC (Peripheral Interface Controlled) controlled two-band stereo audio equalizer. The input audio from each stereo channel is separated into four different analog filters. The PIC has been programmed in BASIC and is used to route the audio signal to one of the four analog filters. Three of the filters have been preset to a certain frequency responses (Jazz, Natural, and Bass), and the fourth filter has the unique ability to be controlled manually. Stereo potentiometers have been installed on the "Manual" filter to allow the user to change the frequency response of the circuit. The manual filter has been divided into two-bands, the "Bass" band, and the "Treble" band. By varying the stereo potentiometer on the manual filter, the manual filter circuit will provide a +/- 12dB of gain for both bands. The "bass" band has a bandwidth of 20-500Hz, while the "Treble" band has a bandwidth of 500hz-20kHz. The PIC Controlled Two-Band Stereo Audio Equalizer is usefully for cutting and boosting certain frequencies, allowing the user to adjust the music to their listening preference. The total material cost is around $150 per unit.

Introduction

Using the knowledge that I have learned while obtaining my Bachelor's Degree in Electrical Engineering from Cal Poly, I have built a PIC (Peripheral Interface Controlled) Stereo Audio Equalizer, with Manual Bass & Treble Controls. Growing up, I've always been fascinated with audio and stereo equipment, and now for the first time in my life, I get to build my very own stereo equipment. A typical setup for my audio equalizer is shown below (Figure 1).

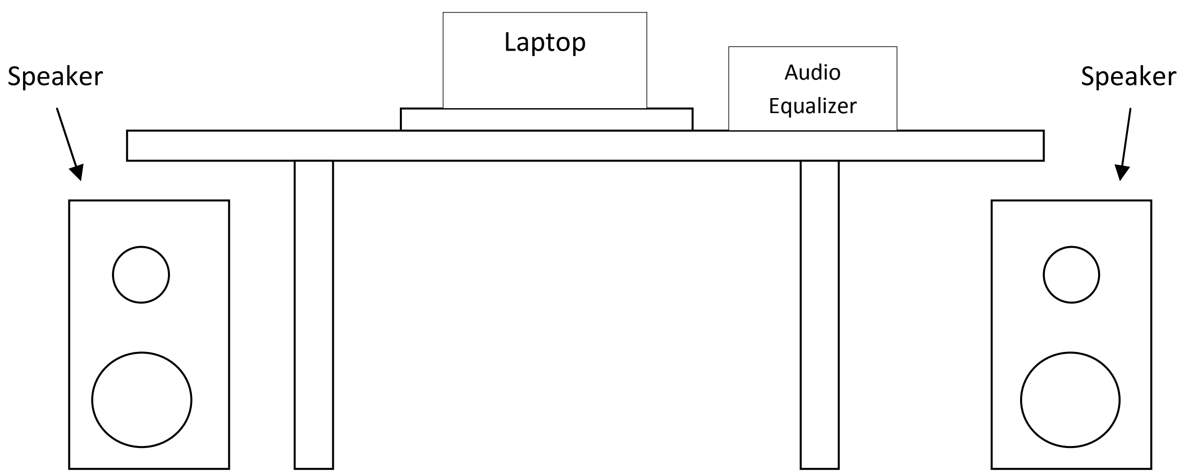


Figure 1: Typical Listening Environment

The size and shape of a room can greatly alter the desired sound of a speaker. Solid objects in a room tend to absorb lower frequencies, while higher frequencies tend to bounce off of object. Being able to attenuate or increase certain frequencies with the turn of a knob, or by the push of a button, is where audio equalizers come in handy in today's music pronged society. With the invention of audio equalizers, music listeners can now adjust the tone of audio to the acoustical likening of their own. This report goes into full detail and operation on how I implemented and designed my very own PIC Controlled Two-Band Stereo Audio Equalizer.

Background

My equalizer will be designed to boost or cut audio frequencies between 20Hz – 20kHz using predesigned filters. I have three different audio settings which are Jazz, Natural, and Bass. The Jazz setting is designed to provide a gain of 12dB across the whole 20Hz-20kHz band, while the Natural setting is designed to span across the same 20Hz-20kHz bandwidth, but instead of providing 12dB of gain, the Natural setting proves 0dB of gain. This 0dB of gain is to allow the user to listen to the music without attenuating of increasing any of the signals for a more “natural” or “un-assisted sound”. The Bass setting has been predesigned to increase the lower bass frequencies (20Hz-500Hz) to a maximum gain of 12dB, while at the same time attenuating the higher frequencies (500hz-20kHz) to a gain of -12dB, for that rich deep bass sound. I also will be able to control the equalizer manually, so I have included one setting which I call the “Manual” setting which can be used if the user prefers to set their own audio equalizer settings.

The Manual setting has the unique ability two divide the 20Hz-20kHz frequency spectrum into two different bands; the two different bands are “Bass” and “Treble.” The Bass band has a frequency bandwidth of 20-500Hz, and the Manual Bass setting has the ability to provide up to +/- 12dB of gain, depending on the users preference. The Treble band has a frequency bandwidth of 500Hz – 20kHz, and once again has the ability to provide up to +/- 12dB of gain, depending once again on the on the users preference. In the Manual setting, the user is able to vary the gain of each band by varying the variable potentiometers hooked up to the analog filter. By varying the bass and treble potentiometers, the user is actually changing the frequency response of the analog filter. By changing the frequency response, the manually filter is cutting or boosting certain frequencies, which increases or decreases certain tones in the music.

I am currently using a PICAXE 08M PIC to control which filters are on, and which filters are off. The PICAXE 08M chip supports 5 inputs/outputs including 3 analog inputs, and requires 4.7V to operate (3 AA batteries). The PIC is programmed in BASIC

and has non-volatile memory, so the user can unplug the PIC from the battery pack, and the PIC will retain the program.

For my project I only require one PIC input, but four different PIC outputs. To control the audio equalizer, I have programmed the PIC to switch on one of the four outputs, each time an input is received (the button is pressed). I have then connected the outputs of the PIC to the control inputs of a SPST quad analog switch (4016BC). On one side of the switch, I have spit the audio from each channel into four different outputs via a 1/8" stereo mini jack. (The 1/8" stereo mini jack is used to spit each channels input, into four different audio input signals.) Then when one of the inputs are "high" on the control side of the quad switch, the audio is passed through the switch to the correct filter. I have also installed an LED on each filer which lights up when the audio passing though a certain filter. The main purpose of this LED is to alert the user which preset audio setting they are listening too; the LEDs are also controlled by the PIC.

After both channels of the audio have passed though the filters, I then re-connect both channels of the filters to a 1/8" stereo mini jack. (The 1/8" stereo mini jack is used as the output connection allowing the user to listen to the music though their headphones or some other device.)

Requirements

The main requirements for this project are listed below (Table 1).

Requirements	Completed
Two-Channels (Stereo)	Yes
Filters to provide +/- 12dB of Gain	Yes
Powered off of 110V AC-DC 12V Transformer	Yes
Four Different Audio Equalizer Settings	Yes
Controlled by PIC	Yes
Frequency Response Plot of each Filter	Yes

Project must be built and in working order by Dec 4th 2009	Yes

Table 1: Project Requirements

Design

I have broken the design down into three different sections to make the whole design process easier to follow. The first design section exposes the user to the System Design (electrical block diagram), the next section is the Filter Design section, and the final section is the Power Supply Design section.

System Design

The block diagram of my PIC Controlled Stereo Audio Equalizer is shown below (Figure 2). As you can see from the diagram below, the design isn't too complicated, simply because everything is symmetrical, due to the fact that the audio equalizer is stereo. I am able to separate CH A from CH B due to a 1/8" stereo mini audio jack. After the audio has been split into two different channels, I then divide one of these audio signals into four different signals, and then connect them to one side of the analog switch (4016BC).

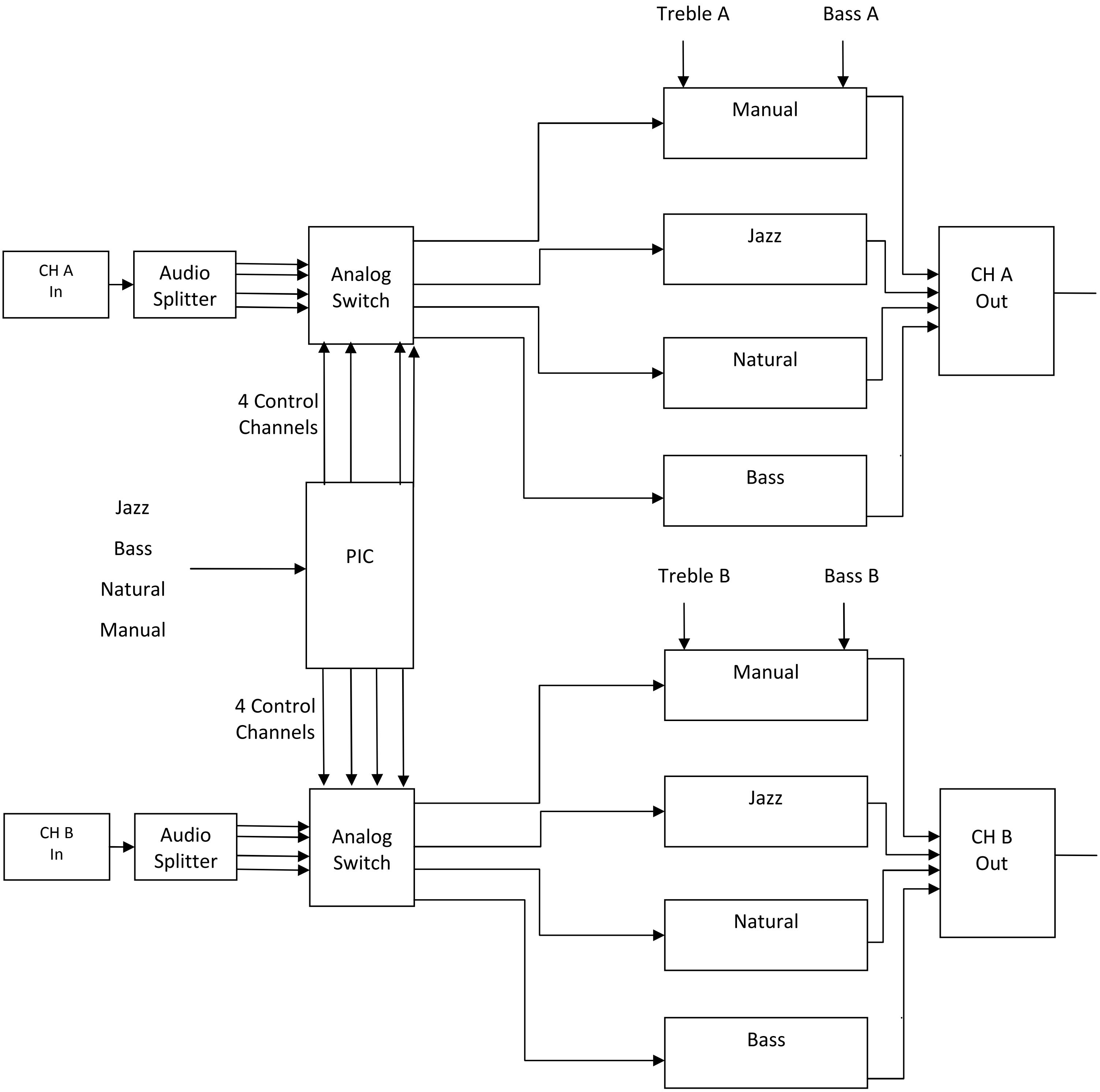


Figure 2: System Block Diagram

As you can see from the bock diagram, the PIC has four different inputs (Jazz, Bass, Natural, and Manual), all stimulating from one input (one button). The PIC has been programmed in BASIC to switch on a different control channel, each time the button is pushed. The PIC is programmed to power up allowing the audio to go to the Manual filter setting, and then each time the button is pressed, the PIC emits a "high"

(4.5V), which routes the audio to the next appropriate filter in the sequence. As the PIC admits a high to the next setting (Jazz in this case), at the same time, the PIC is producing a “low” (0V) on the previous control channels input (Manual setting in this case), turning off the audio routing to the previous filter. As the user keeps pushing the button, the PIC keeps turning-on/cutting-off the appropriate filter each time the button is pressed, and this is how the audio is routed through the audio equalizer. The PIC has been programmed to loop from Manual, to Jazz, to Natural, to Bass, and then the sequence repeats itself from there, each time the button is pressed.

All of the filters have been setup and configured to supply a boot/cut of about +/-12dB of gain (Figure 3). As stated previously, the Jazz, Natural, and Bass settings have all been preset to a certain gain. However, to vary “Treble A’ and “Bass A”, I have hooked each manual filter up to a pair of 100K stereo potentiometers, which allows the user to vary the boost/cut of these two-bands independently, with the turn of a knob.

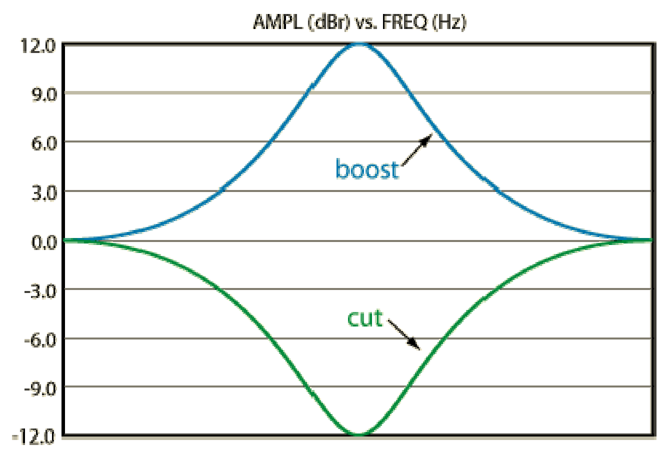


Figure 3: Example of Boots/Cut for each Filter

Since only one filter is on at a time, there is no need to use a summing amplifier to connect all the signals together. I then connect all four outputs of each channel back to another 1/8” stereo mini audio jack, and the system design is complete.

Note: Everything that the PIC does to CH A, it does simultaneously to CH B to provide stereo sound.

Filter Design

Original Filter Design

The graphic equalizer will have two channels (stereo) and a cut/boost operation for two different sections of the 20Hz-20Khz range. This 20Hz-20Khz range frequency is the audibility output of the human ear. The low control (bass control) will boost or cut all of the frequencies from 20Hz to 500Hz. Finally, the treble control will boot or cut the frequencies from 500Hz to 20kHz. The center frequencies for this design will be the center of each of the two audio bands listed above. The selected center frequencies will be chosen to be 250Hz, and 13kHz. The graphic equalizer specifications are listed below (

Table 2).

Original Filter Design Specs	
Number of Channels	2
Bands Per Channel	2
Center Frequencies	250hz, 13khz
Maximum Boost	+12dB
Maximum Cut	-12dB
Normal Gain	0
Q Factor Each Band	1.5
Filter Type	Active (741 Op Amp)
Input Connection	Stereo 1/8" Jack
Output Connection	Stereo 1/8" Jack

Table 2: Original Filter Design Specs

For the audio equalizer, I will use a Q value of 1.5. A Q value of 1.5 is used in this design because it will boost a wider range of frequencies at the center frequency. The filer type will be a band pass filter (Figure 4), and the typical frequency response for this filter is show below (Figure 5)

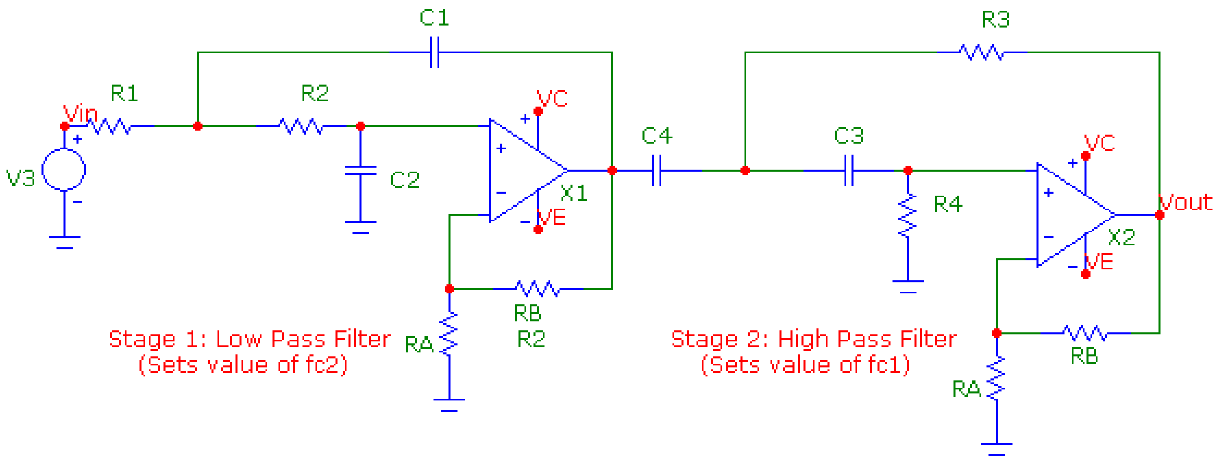


Figure 4: Sallen-Key Bandpass Filter

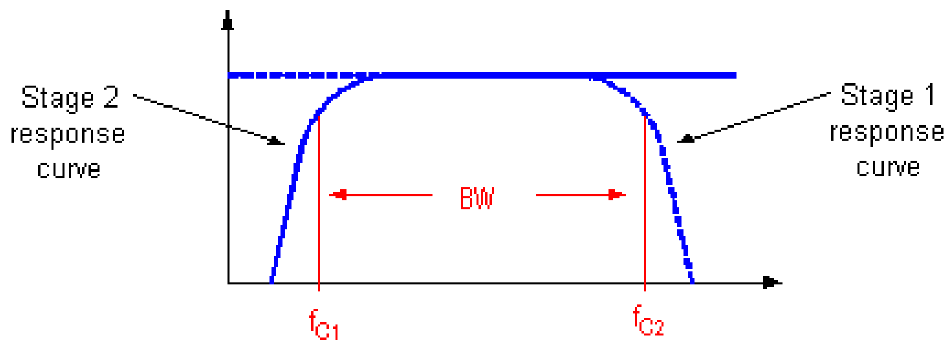


Figure 5: Typical Frequency Response of Filter shown in Figure 4

The circuit shown above in Figure 4 consists of two Sallen-key circuits cascaded together. The first stage is a low pass filter, and the second stage is a high pass filter. After cascading the two stages together, we get a bandpass frequency response (Figure 5). Since each of the two manually controlled filters will consist of a Sallen-Key bandpass filter, I will design the Bass filter here in the document. Then to achieve the other filters, we follow the same principles, except we change the resistor values of the circuit to move the corner frequencies to the appropriate location.

Bass Filter Design

Since we have already defined f0 = 250Hz for the center frequency, and Q= 1.5 (the quality factor), the first thing that we need to determine is KF;

$$Kf = 2\pi fo = (2\pi)250hz$$

Next we determine KZ be letting C1 = 10nF.

Note: We let C1 = 10nF because we have a limited supply of capacitor values when it comes to building circuits.

$$Kz = \frac{1}{Kf\ C1} = \frac{1}{(2\pi)\ 250hz\ (10nF)} = 63.661k\Omega$$

Using 1% resistors values; Kz = 63.4kΩ

Next arbitrary choosing Ra = 1k, we determine Rb for the circuit, Rb helps control the gain of the circuit and is given by the following equation;

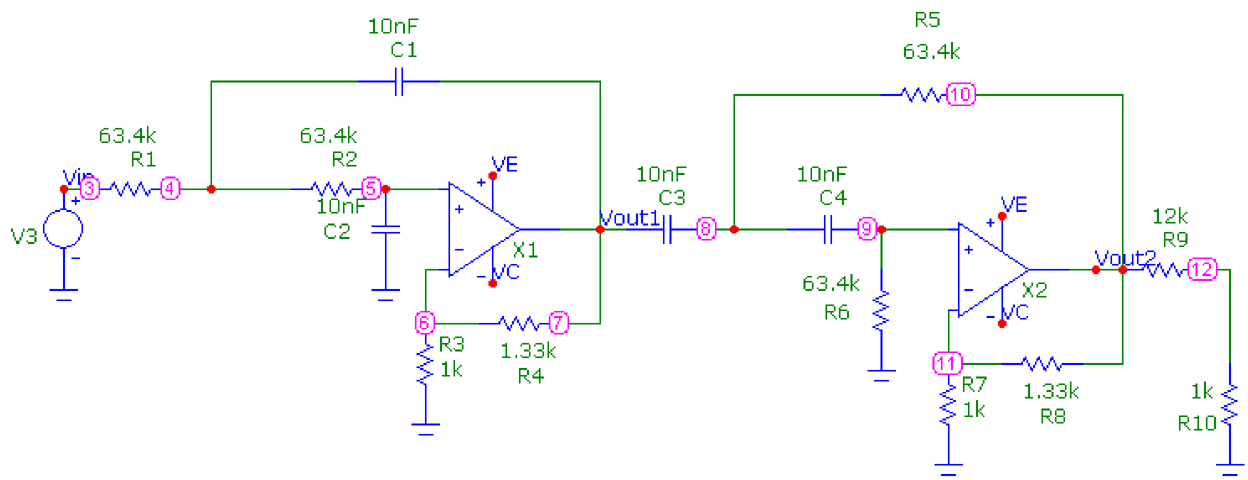
$$Rb = \left(2 - \frac{1}{Q}\right) Ra$$

where Q = quality factor

$$Rb = \left(2 - \frac{1}{1.5}\right) 1k = 1.33k\Omega$$

Using 1% resistors values; Rb = 63.4kΩ

Next using Microcap, I simulated the design in MicroCap (Figure 6 & 7).



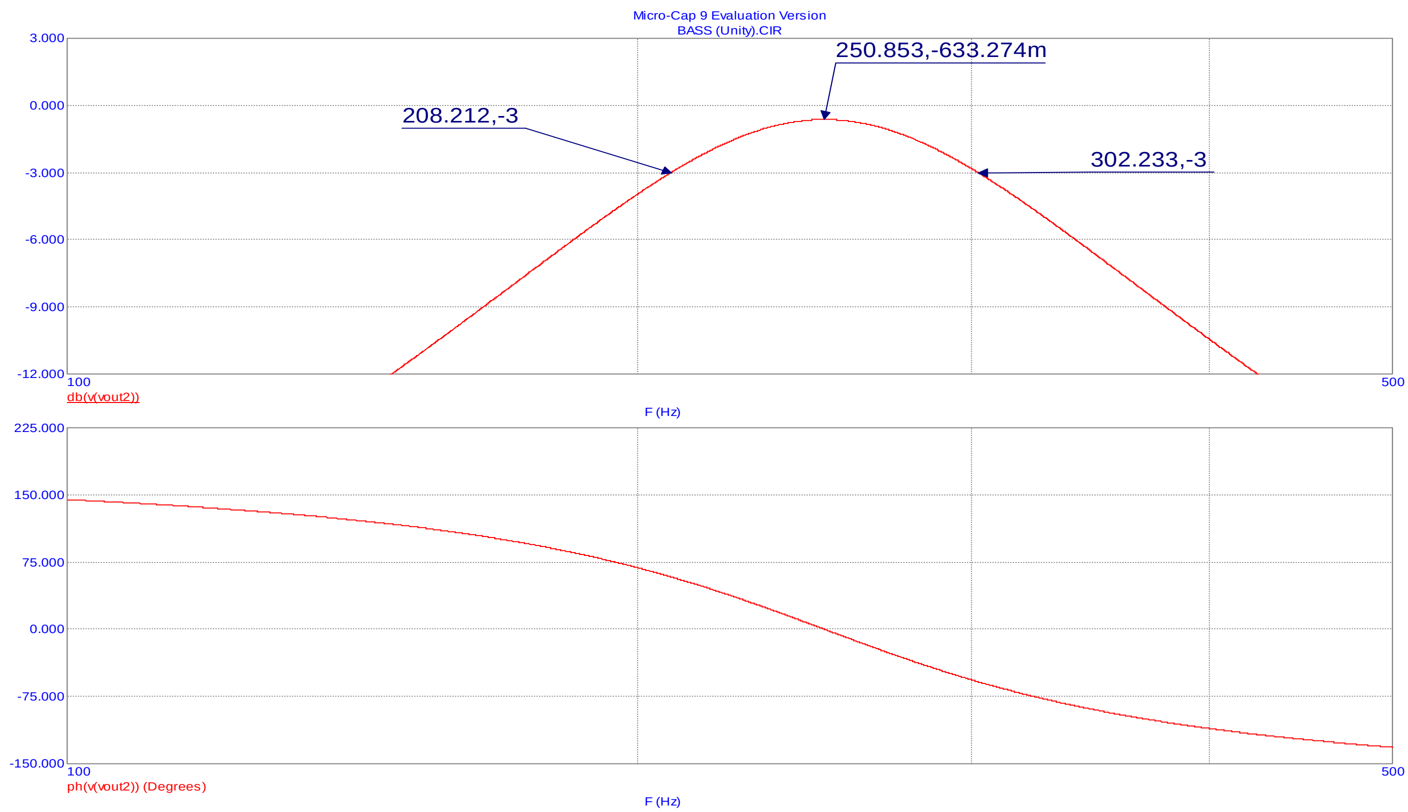


Figure 6: Sallen-key Bass Filter with Unity Gain

Figure 7: Sallen-key Bass Filter Frequency Response with Unity Gain

As you can see from the graph above, I have created a bandpass filter with a center frequency f0 = 250Hz. Also the 3dB corner frequencies for the bass filter are fc1 = 202Hz and fc2 = 302Hz, and the 3dB Bandwidth = 100Hz. Using the design procedure above, I designed the Treble filters.

Initially I had intended to use the Sallen-Key filter design for the Treble and Bass filters, because it's a simple yet productive design. However, after playing around with the design, I was not able to develop a circuit that could produce the +/- 12dB boost that this audio equalizer needs to have. After talking to my professor (Professor Granneman), he suggested that I turn to a Tone Control circuit. After researching numerous Tone Control circuits I decided to use the Baxandall Tone Control circuit.

Baxandall Tone Control Circuit

The Baxandall Tone Control circuit is the most common circuit in audio design applications (Figure 8 & Figure 9). By adjusting X3 in the circuit below, you adjust the treble setting, and by adjusting X2, you adjust the bass setting.

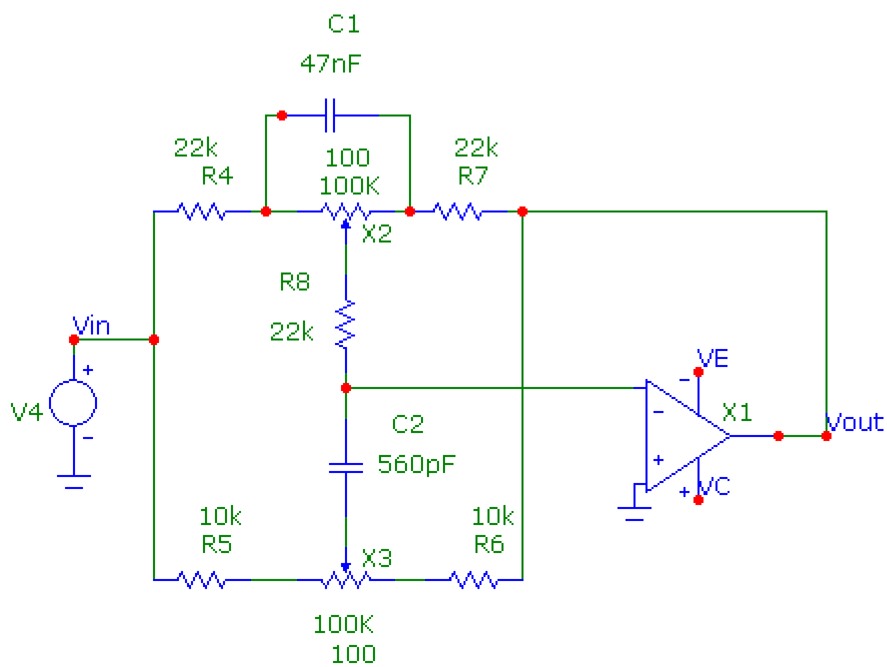


Figure 8: Baxandall Tone Control Circuit

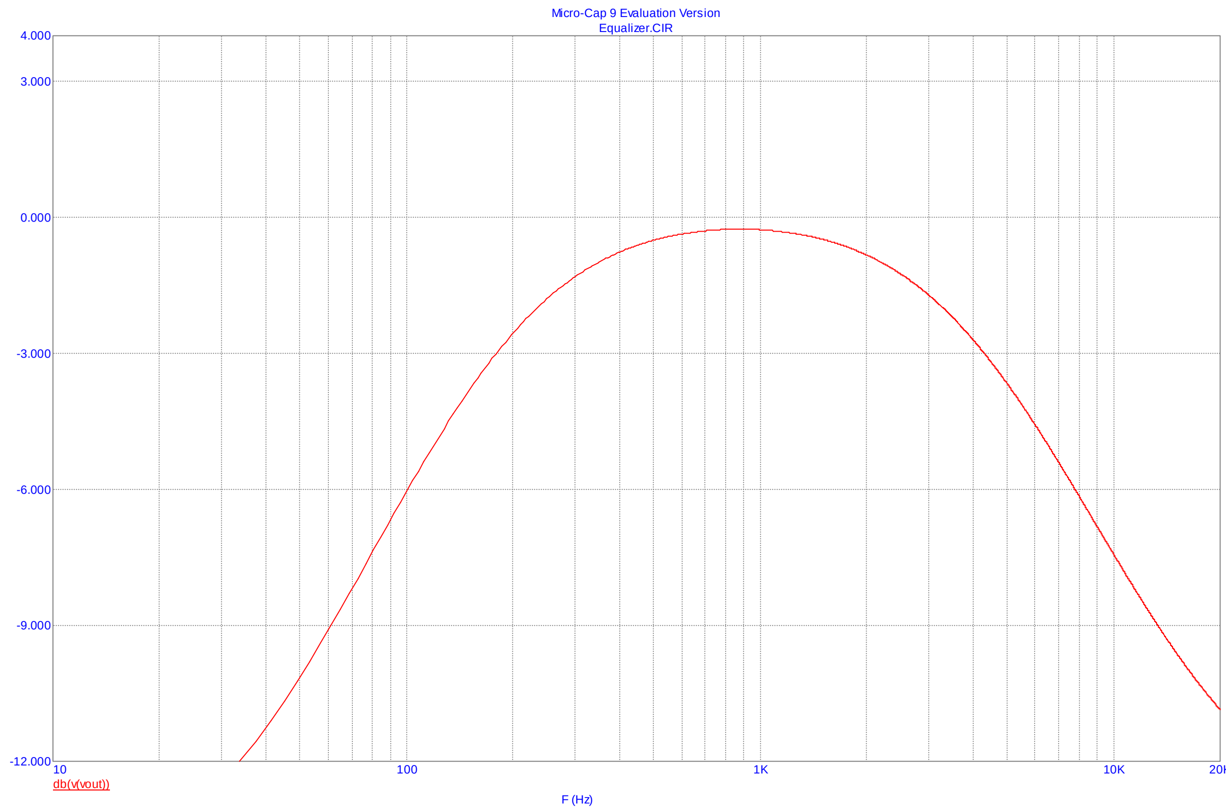


Figure 9: Baxandall Frequency Response

As you can see from the graph above, the Baxandall toned circuit produces the correct response that for my equalizer. This circuit is named after PJ Baxandall who designed and implemented this circuit many years ago. One reason why this circuit is really practical is due to the symmetry involved in the design. The circuit allows no interaction between both control pots, and when both pots are centered, the opamp acts as a buffer and there is no net gain or loss. Or in other words, the frequency response is totally flat.

The circuit is a frequency dependent feedback arrangement, and provides boost and cut for high and low frequencies as you can see from the graph. The maximum boost is about 12dB, and the lower turnover frequency is about 150Hz, and the upper turnover frequency is about 2.5kHz. Once again, these values can be changed by modifying the values of the bass and treble capacitors, and the amount of boost and cut is varied by changing the series resistors for each pot (X2 & X3). One thing to note is that the circuit needs to be driven from a low impedance source in order for it to work.

When the bass control (X2) is turned to the max, essentially you are creating an active low pass filter, and when you turn the treble control (X1) to the max, essentially you are creating a high pass filter. This Tone Control circuit turned out to be the best design for the treble and bass controls for my audio equalizer. I also used this circuit for the three preset audio settings. I was able to do this by making the potentiometers non adjustable. A graph of my Bass, Jazz, and Natural Settings are shown in the Test Results section of this document.

Power Supply Design

Zener Diode Voltage Regulator Circuit

In order to power up the audio equalizer, I needed to design a circuit that could be used as a power module to provide the necessary +/- 5V rails, and virtual ground that the 741 op-amps needed to operate successfully. After purchasing an AC to DC 12V wall adapter power supply, I was faced with the task of creating a power module (dual rail power supply) out of the out of this 12V wall adapter (Figure 10).

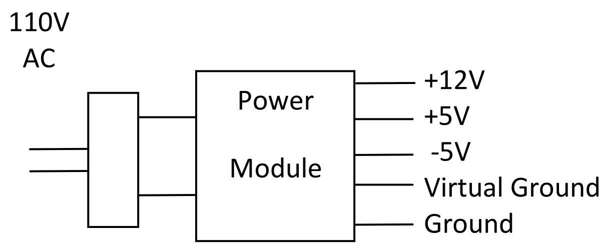


Figure 10: Electrical Block Diagram

After conducting research on dual power supply circuits, I was able to come up with a design to provide the +/- 5V rails, and virtual ground, that I needed for my audio equalizer. The design is simple and is created off of a simple zener diode voltage regulator circuit (Figure 11).

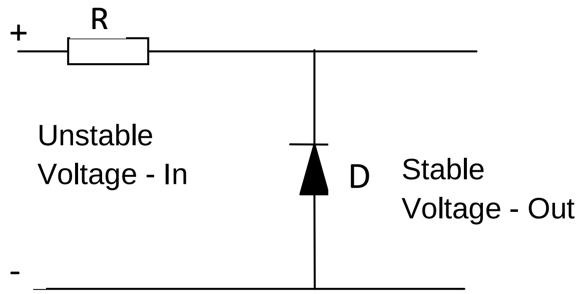


Figure 11: Zener Diode Voltage Regulator Circuit

A zener diode is a special kind of diode that allows the current to flow both ways. The zener diode circuit above allows a fixed stable voltage to be taken from an unstable voltage source. As long as the input voltage is above the desired output voltage, the voltage across the zener diode will be constant. As the input voltage increases or decreases, the current through the diode will also fluctuate, but the voltage drop across the zener diode will always stay the same. The voltage across the resistor will always be the difference between the zener diode voltage, and the input voltage. Since I have a 12V DC supply as my input source, using two (2) 5.1V zener diodes, and a 3.9Ω resistor, I was able to create the power module circuit module.

Power Module Circuit

The power module circuit (Figure 12) is an expansion of the zener diode voltage regulator circuit.

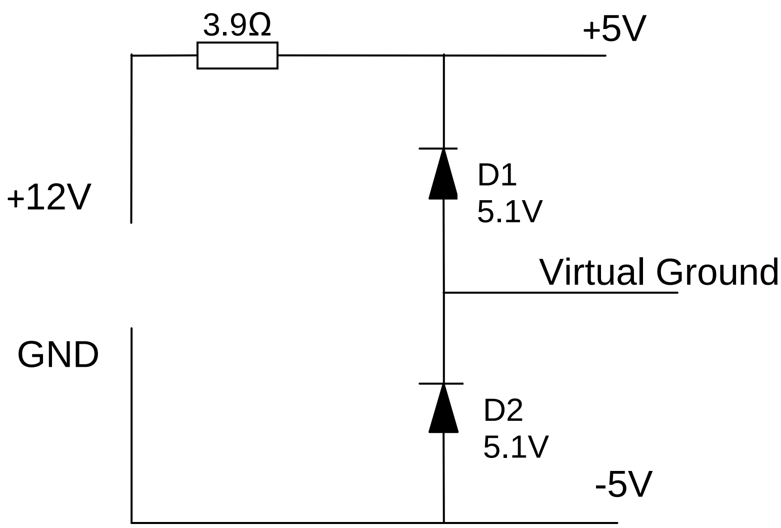


Figure 12: Power Module Circuit

Since the input is 12V, the 3.9Ω resistor, creates about a 10V drop across both 5.1V zener diodes. Since each of these zener diodes must have a 5.1V drop across them, and because there is a 10V drop across both of them, the circuit creates a virtual ground at the middle of the series connection, and this is how the +/- 5V rails are created. At the same time, this circuit provides a constant 12V DC output which can be used to power up other devices.

Major Components Breakdown

Besides the Bandaxall Tone Control circuit, the next two major components of this project were the PIC and Analog Switch.

PIC

The PIC is the mini-brain of the circuit. I used the PIC to control the preset filters. The PIC has been programmed in BASIC and is responsible for switching on and off the four different filters. When the pushbutton is pushed, the PIC has been programmed to

recognize the input, and will route the audio to the correct filter (this is done via and analog switch, which I will explain in the “Analog Switch” section). After researching what kind of PIC I should use to control my circuit, I decided to use a PICAXE 08M Chip (Figure 13)

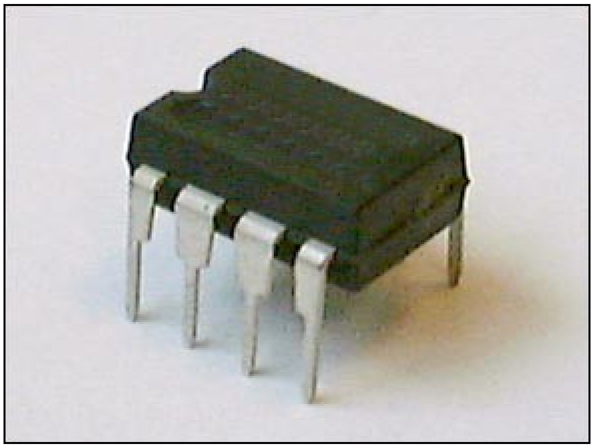


Figure 13: PICAXE 08M Chip

“Microcontrollers are exciting new electronic ‘single chip computers’ that are rapidly being introduced into industry and education. The ‘PICAXE’ system is an extremely powerful, yet low cost, microcontroller programming system designed to simplify educational and hobbyist use of microcontrollers”.

I decided to use this 08M chip because it is an inexpensive entry-level microcontroller, and at the same time, it’s great for the hobbyist, as you can see from the quote above. The specifications for the PIC are listed below (Table 3).

PICAXE 08M Chip	
Power Supply	4.5V or 5V DC is recommended
Outputs	Each output can sink or source 20mA. This is enough to light an LED but will not, for instance, drive a motor. Total maximum current per chip is 90mA.
Inputs	Inputs should be above (0.8 x power supply) to be high, below (0.2 x power supply) to be low. It also recommended to tie all unused inputs low VIA a 10K resistor
ADC	The ADC range is the power supply voltage range. The maximum recommended input impedance is 20k. Unconnected ADC will 'float' giving varying false readings.
Serial Download Pin	The serial download pin must never be left floating. This will give unreliable operation. Always use the 10k/22k resistors as shown below, even if the chip was programmed on a different board.
Reset Pin	The reset pin (if present) must **never** be left floating. This will give unreliable operation. Always tie high (ie to the positive supply) via a 4k7 or 10k resistor

Table 3: PICAXE 08M Specs

A picture of the download circuit is shown below (Figure 14), and the Pin Locations is shown also shown below (Figure 13).

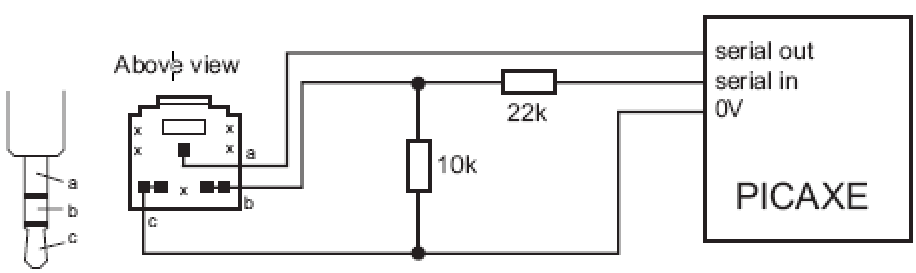


Figure 14: PICAXE Download Circuit

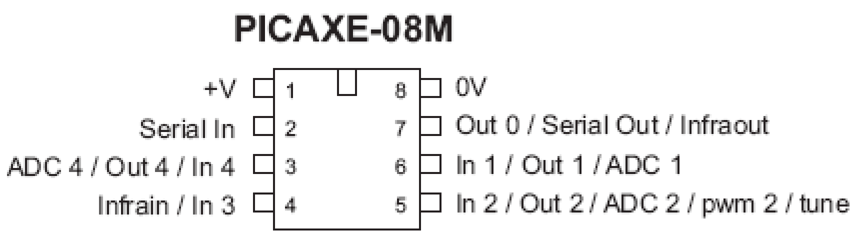


Figure 15: PICAXE 08M Pinout

Since my design only calls for four outputs (Jazz, Natural, Bass, and Manual), I am using PIN 3 as the input, and PIN 7,6,5, and 4 as outputs going to the analog switch. A simple diagram explaining my circuit design is shown below, but instead of having the output be a motor, speaker, or LED, I will be outputting to the analog switch (Figure 16).

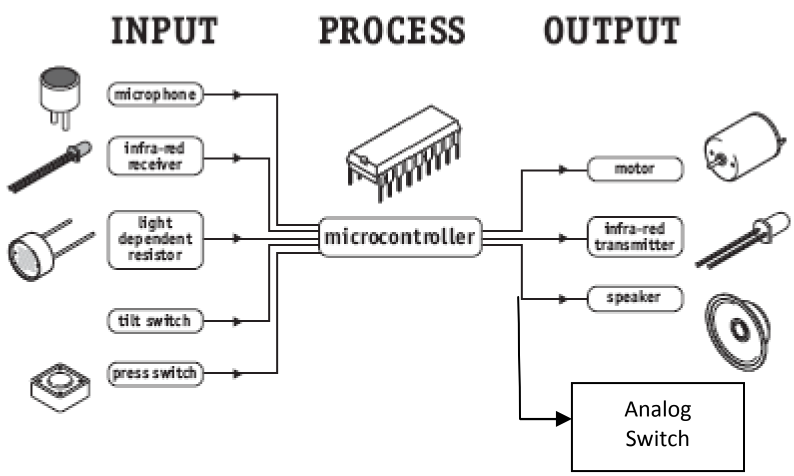


Figure 16: Sample PIC Diagram

In order to use the PICACE 08M Pin PIC, I had to obtain the PICAXE 08M Starter kit (Figure 17). The starter kit is inexpensive and includes everything that I needed to program the PIC, except for the PICAXE programming cable. The programming cable is rather expensive, but it has the FT232R embedded into the USB (Figure 18) .

Figure 17: PICAXE Starter Kit

The kit above includes the following components (Table 4).

PICAXE Starter Kit Components
PICAXE 14M IC
14-Pin Standard Project Board and Parts
Battery Holder
Software CD

Table 4: PICAXE Starter Kit Components

Figure 18: PICAXE USB Programming Cable

The PICAXE Starter Kit comes with its own development software. Unlike most microcontrollers, this PIC does not use C-coding, instead you program the chip in BASIC. One reason why I chose this PIC was because the programming software is free, low cost, and easy to use. The microcontroller also has non-volatile memory, so the PIC retains the program, even if the power source is removed. To control my audio equalizer, I am simply running “loops” in BASIC, and I have attached the BASIC code in Appendix B.

Analog Switch

I have used an Analog Switch (4016BC) to activate one of the four filters when the button is pressed (Figure 19).

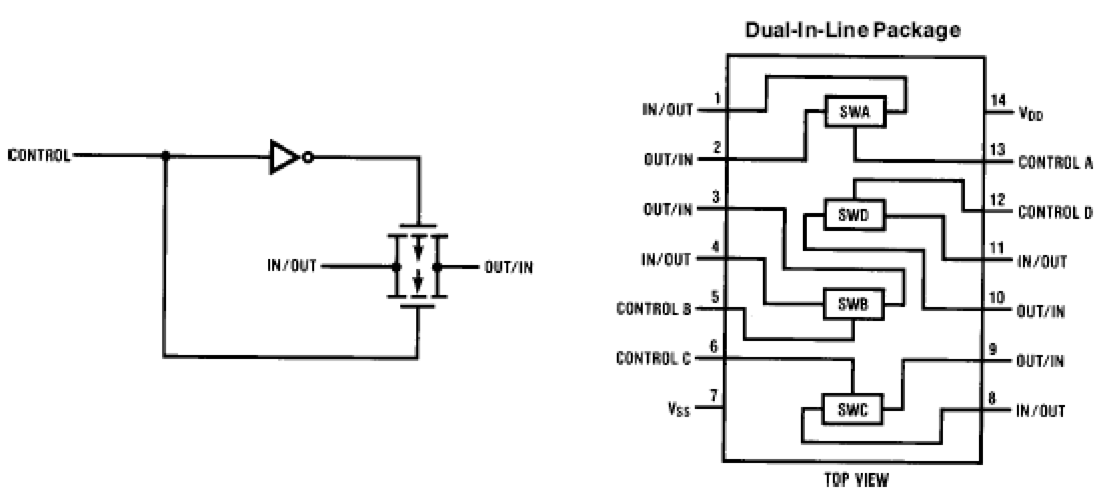


Figure 19: 4016BC Quad Bilateral Switch

The four outputs of the PIC will connect to the four “control channels” of the analog switch. On the “in” side of the switch, I have connected the four CH A audio inputs, to the four different “in” connections. I have then routed each “out” connection of the switch, as inputs to one of the four “analog filters”, and this is how the audio is routed to each filter. Each time the button is pressed, a different control channel is being cut on, which in turn routes the audio to the appropriate filter.

Note: Everything that the PIC does to CH A, it does simultaneously to CH B to provide stereo sound.

Test Results

Using the spectrum analyzer, I hooked each filter up to this device to measure the frequency response of the filter. (This measurement is done by graphing the “transfer function” of the filter.) After successfully hooking up the filter to the spectrum analyzer and measuring its response (Figure 20 & Figure 21), using Microsoft Excel I duplicated the graph by recording the gain in dB of the circuit, as the frequency changed. Using the variable potentiometers, I was able to manually adjust the filter to the preset setting that I wanted. The frequency response plots (Jazz [Figure 22], Natural [Figure 23], and Bass [Figure 27]) are shown below. I have also attached the Manual frequency response plot (Figure 26 & Figure 27), showing the maximum and minimum gain of this filter.

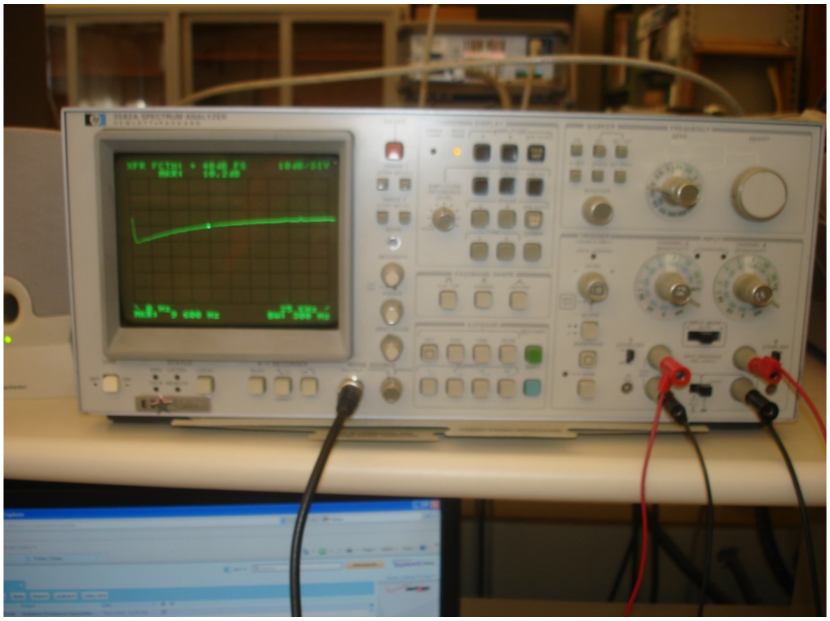


Figure 20: Sample "Manual Treble" Frequency Response Plot

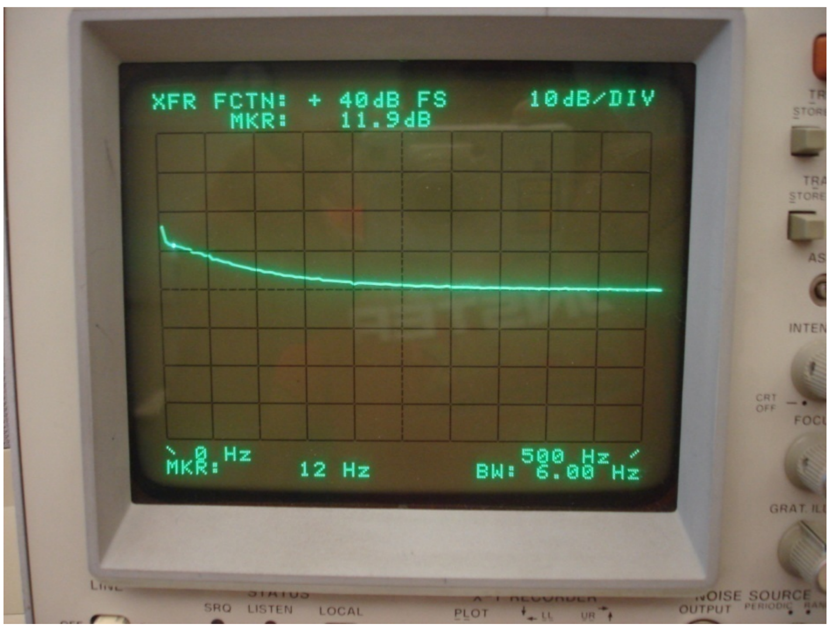


Figure 21: Sample "Bass Filter" Frequency Response Plot

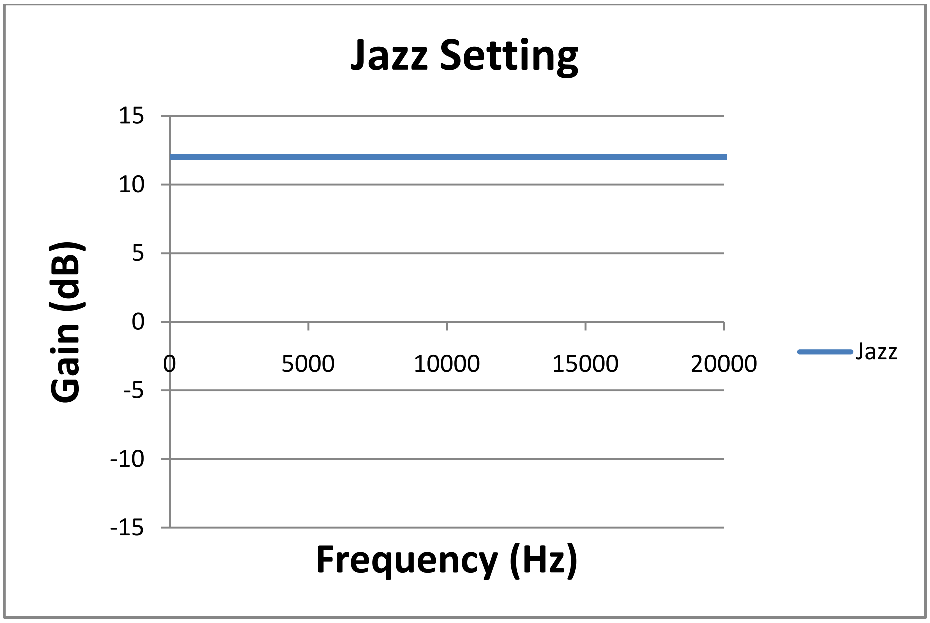


Figure 22: Jazz Setting Frequency Response Plot

The Jazz filter is designed to provide 12dB of gain, across the 20Hz-20kHZ bandwidth.

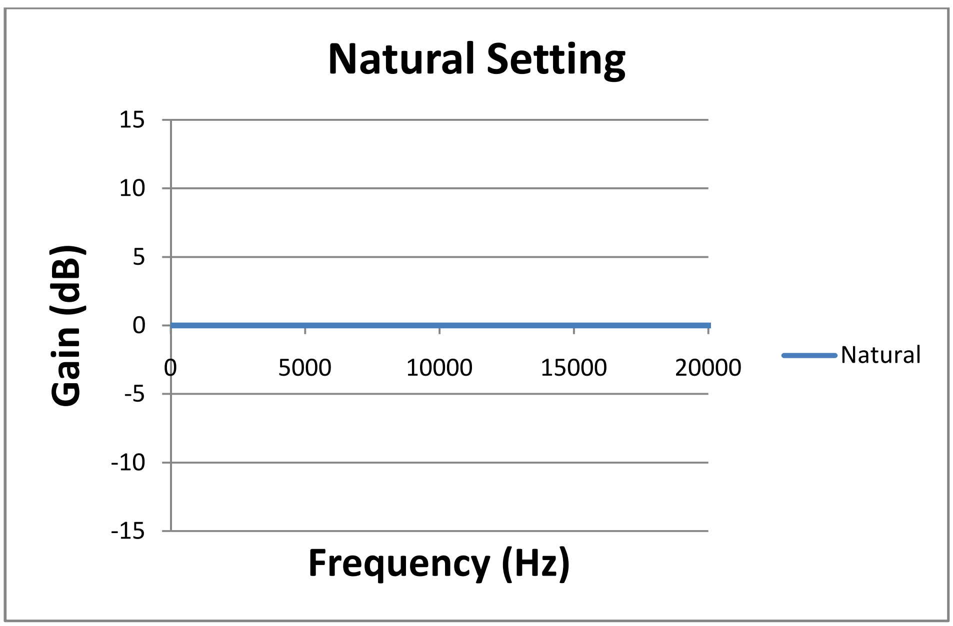


Figure 23: Natural Setting Frequency Response Plot

The Natural filter is designed to provide 0dB of gain, across the 20Hz-20kHZ bandwidth.

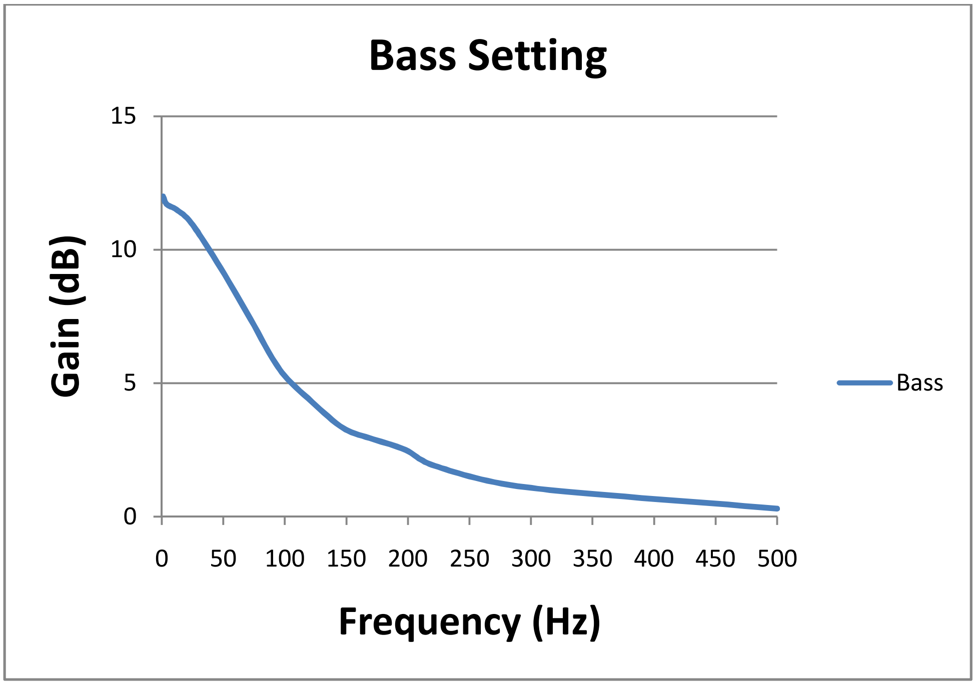


Figure 24: Bass Setting Frequency Response Plot (0-500Hz)

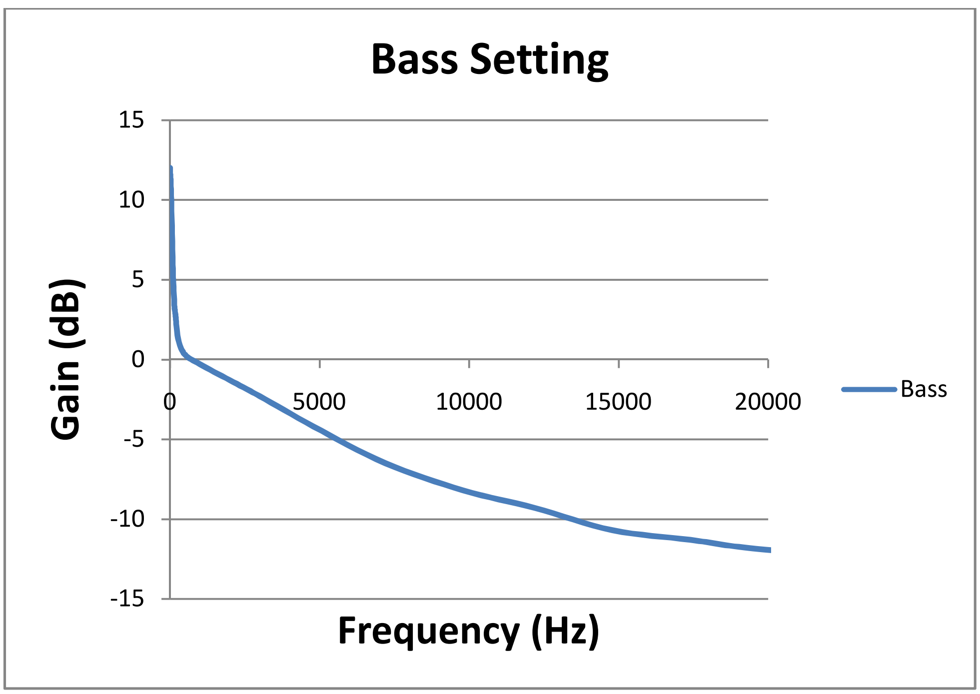


Figure 25: Bass Setting Frequency Response Plot (0-20kHz)

The Bass filter is designed to provide 12dB of gain for the lower frequencies, while at the same time attenuating the higher frequencies, for the deep rich bass response.

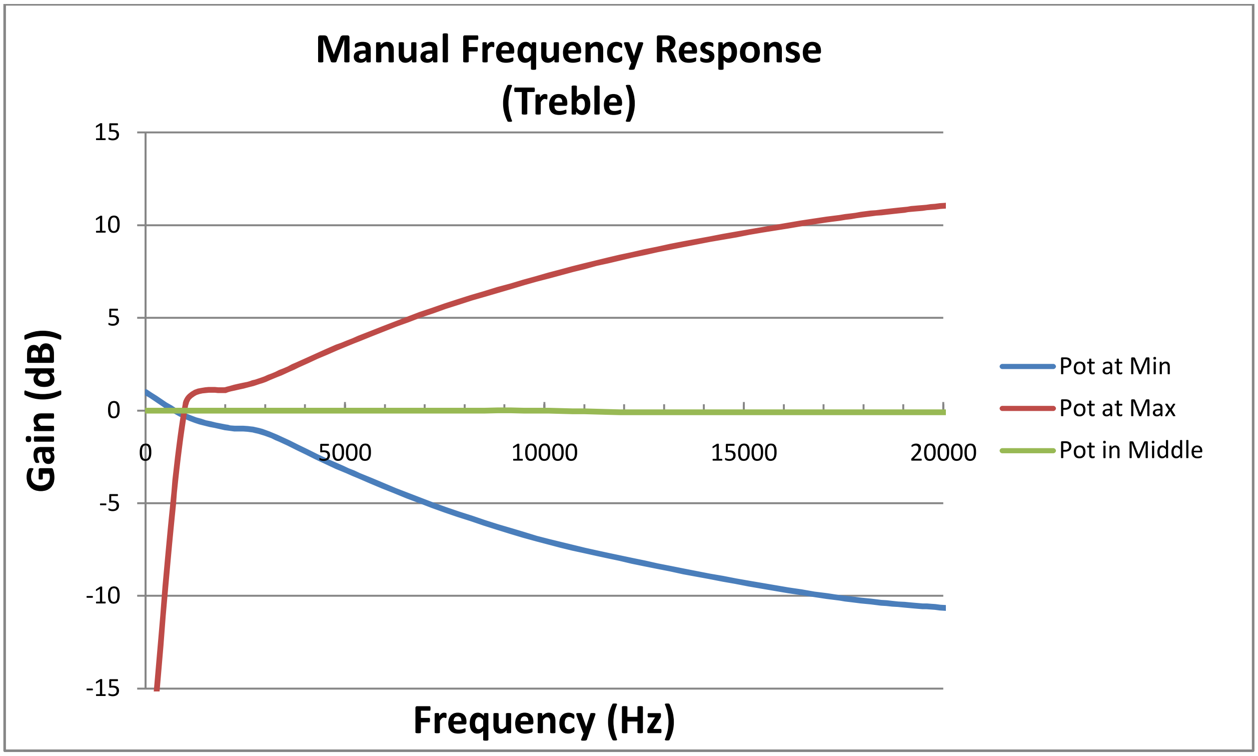


Figure 26: Manual (Treble) Frequency Response Plot

The Manual Treble filter is designed to provide +/-12dB of gain with most of the gain being at the higher frequencies.

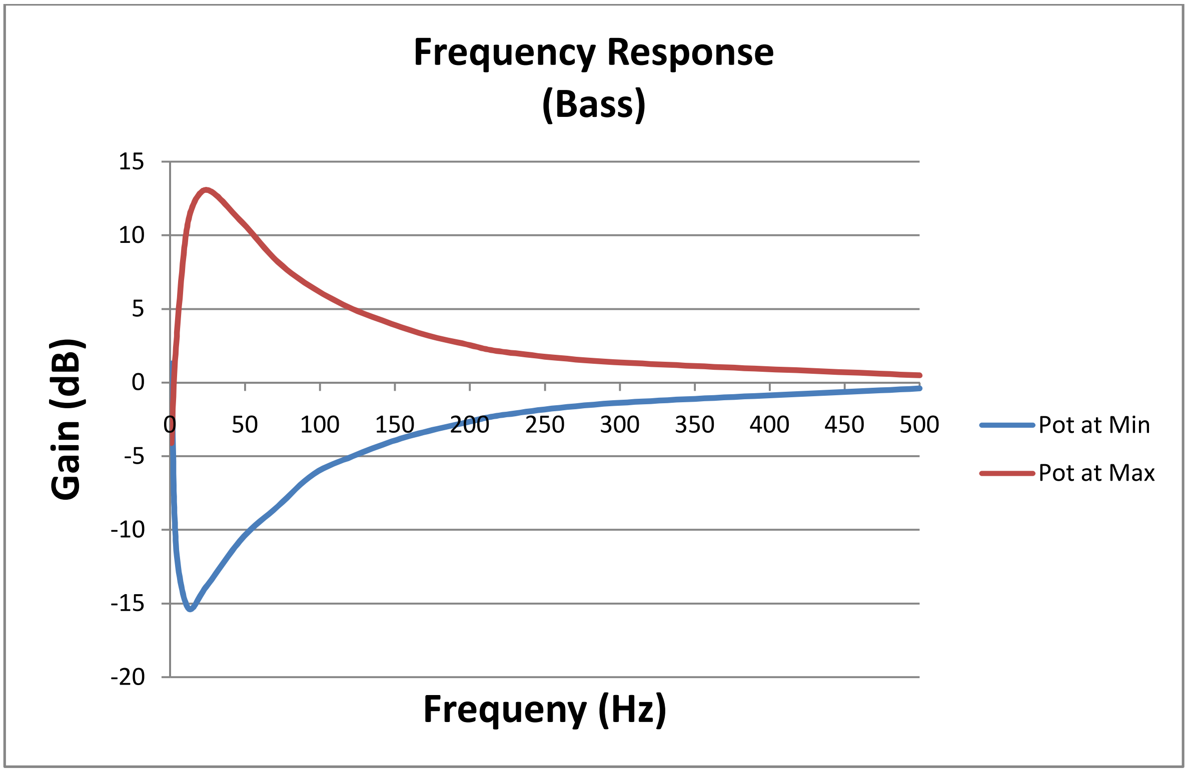


Figure 27: Manual (Bass) Frequency Response Plot

The Manual bass filter is designed to provide +/-12dB of gain with most of the gain being at the lower frequencies.

Conclusion

My PIC Controlled Two-Band Stereo Audio Equalizer has been successfully built, and works beyond my expectations; a picture of the final project is shown below (Figure 28).

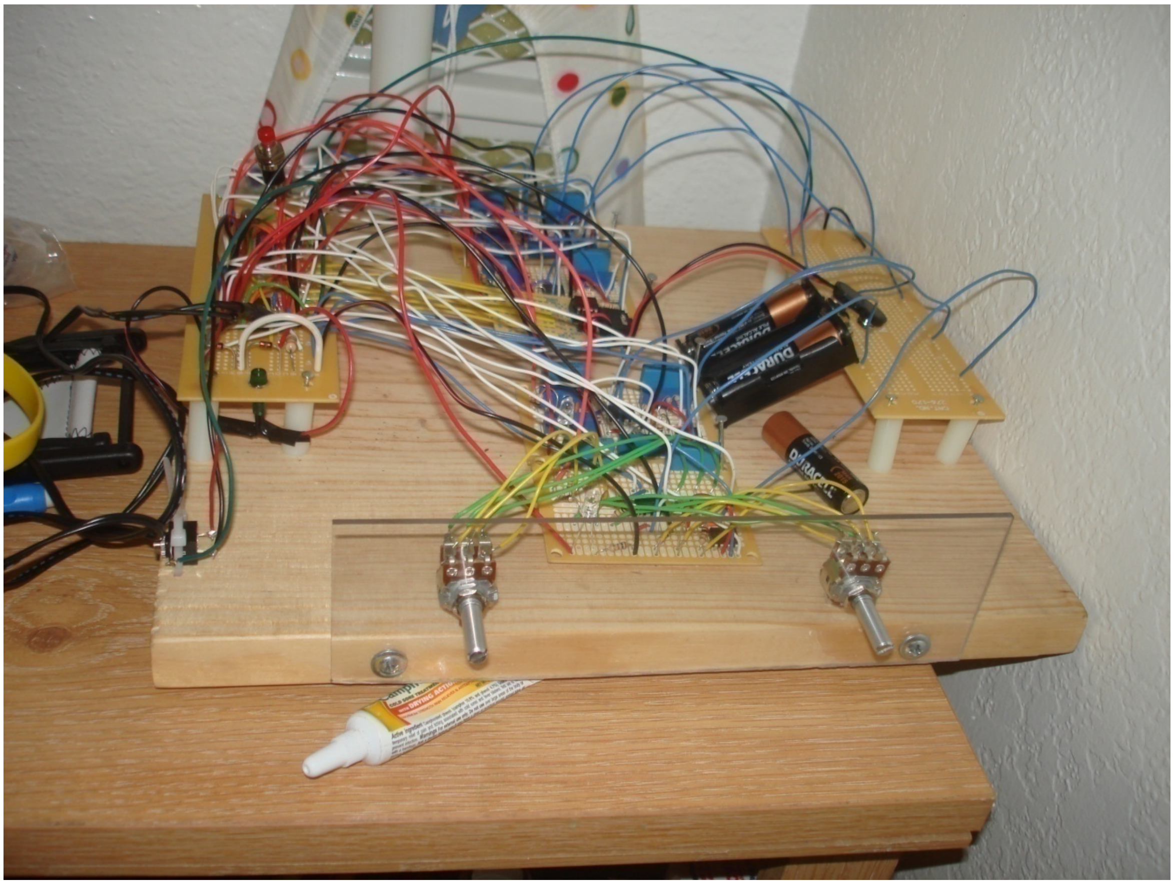


Figure 28: Final Project

As I stated earlier, the total cost of the project was around $150, and besides ordering the PIC from overseas, I was able to get most of the parts either at my local Radio Shack, or from Jameco electronics. I feel as if this product can compete with other audio equalizers on the market today simply because this product works superb. The audio is crystal clear, and you can clearly hear how the audio changes as you listen to the music through the different filters.

One possible improvement on the unit is to physically lay it on a PCB. Because I built the product from scratch, I elected not to layout the circuit on a PCB. However, now that I have completed the unit, if I were to build another one, I would definitely lay the circuit out on a PCB, to give it more of a professional appearance. Basically the design went together pretty smoothly. The one major setback that I faced was the fact that I had unwanted audio bleeding though the filters, when the input was off on a certain filter. I was able to correct this problem by adding a metal film capacitor between the analog switch and the audio input signal, and the problem went away.

In conclusion, I am very glad that I built this audio equalizer. Not only did I get to apply my engineering knowledge to it by testing, building, and troubleshooting the audio equalizer, but I also learned a lot about audio filters and how they work,. Since music and DJ equipment is a passion of mine, this was the correct senior project for me.

Bibliography

[1] Schauman, Rolf. *Design of Analog Filters*,USA: Oxforfd University Press, Jan 2001

[2] Lancaster, Donald. *Active-Filter Cookbook*, First Edition: Macmillan Pub Co, July 1975

Appendix A: Parts List & Cost

Parts Description	Qty	Price Each	Total Cost
12V DC Power Supply	1	$9.95	$9.95
5.1V Zener Diode	2	$0.35	$0.75
47nF Capacitor	8	$0.20	$1.60
100K Pot	6	$1.50	$9.00
560pF Ceramic Capacitor	8	$0.30	$2.40
22K Resistor	24	$0.15	$3.60
10K Resistor	16	$0.15	$1.40
741 Op Amp	8	$0.15	$1.20
100K Dual-Ganged Stereo Volume Control	2	$3.49	$6.98
Multipurpose PC Board with 417 Holes	4	$1.99	$7.96
Printed Circuit Board with 550 Holes	2	$2.99	$5.98
500-Piece 1/4-Watt Carbon-Film Resistor Assortment	1	$12.99	$12.99
4016BC Quad Bi-Lateral Switch	2	$0.09	$0.18
Mini SPST 0.5-Amp Momentary Switch	1	$3.49	$3.49
0.1uF Metalized Polyester Film Capacitor	4	$1.49	$5.96
Low Voltage Red LED	4	$1.49	$5.96
12-Volt Hi-Brightness Green Lamp Assembly (2-Pack)	2	$2.49	$4.98
15-Watt Soldering Iron with Grounded Tip	1	$8.49	$8.49
Standard Rosin-Core Solder (0.5 Oz.)	1	$3.49	$3.49
De-soldering Braid	1	$3.49	$3.49
1/8" Stereo Panel-Mount Phone (2-Pack)	1	$2.49	$2.49
Misc Wires	1	$15.99	$15.99
PICAXE Microcontroller Starter Kit	1	$49.99	$49.99
Total Cost			$158.37

Appendix B: BASIC Code Controlling the PIC

```
init: let pin3 = 0

buttonpress1: 'Switches on output "0" and cuts off output "4"
pause 100   'Waits for next button to be pressed to turn on output "1"
high 0
low 4
if pin3 = 1 then
goto turnon1
else
goto buttonpress1
endif
turnon1:  'Switches on output "1", and cuts off output "0".
pause 100 'Waits for next button to be pressed to turn on output "2".
pin3 = 0
low 0
high 1
if pin3 = 1 then
goto buttonpress2
else
goto turnon1
endif

buttonpress2: 'Switches on output "2" and cuts off output "1"
pause 100     'Waits for next button to be pressed to turn on output 3.
low 1
high 2
pin3 = 0
if pin3 = 1 then
goto turnon3
else
goto buttonpress2
endif
turnon3:  'Switches on output "4" and cuts off output "2"
pause 100 'Waits for next button to be pressed to turn on output 0.
high 4
low 2
pin1 = 0
if pin3 = 1 then
goto buttonpress1 'Loop repeats to the top.
else
goto turnon3
endif
```