**ECE 671**

**Project: The Stereo AV Equalizer**

**Prof. S. Ganesan**

**Oakland University**

**Summer 2011**

**Prepared by**

* **GERALD JOCHUM**
* **BHEGIN NTAGAZWA**
* **MAURICE FARAH**

Webpage: <http://www.oakland.edu/~mfarah/DSP>

**Table of Contents**

Abstract \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ page 3

Introduction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ page 3 – 6

Procedures \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ page 6 - 19

Test Results \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ page 19

Conclusion \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ page 19-20

References \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ page 20

**Abstract:**

This project combines the implementation of (six) finite impulse response (FIR) filters, delay line, and echo effect subprojects using DSP tool TMS320C6713 DSK. The FIR filters are tools used to provide three groups of frequency components per channel where the gain adjustment gave control to amount of each frequency group was present in output.

**Introduction:**

This project demonstrates channel routing among left and right, signal delay, echo, muting select channels, or application of Low pass, Band pass, and High pass filters to signal provided Line input and output Lineout/Head phones on TMS320C6713 DSK. The use of FIR filters requires two MatLab modules to produce the filter COF files. The fdatool function offers the type of filter and design parameters related to chosen filter. Exporting the resulting array variable in MatLab where the dsk\_fir67.m program formats the output to a COF file. The COF files are the coefficients used to implement the FIR filter. The FIR model function works on sampling of an ideal analog signal x (t) and applying the mathematical model

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (1)

where ( *t* - *kT* ) is the impulse (delta) function delayed by *kT* and *T* = 1/ *F s* is the sampling period. The function is zero everywhere except at *t = kT.*

In order to understand how digital filter is working it better to look into difference equations “a recurrence equation”*.* To solve the difference equation z-transform of expression such as x (n-k) must be found. This expression corresponds to the *k*th derivative of an analog signal x (t). The order of the difference equation is determined by the largest value of *k*. For example, k =2 represents a second order derivative

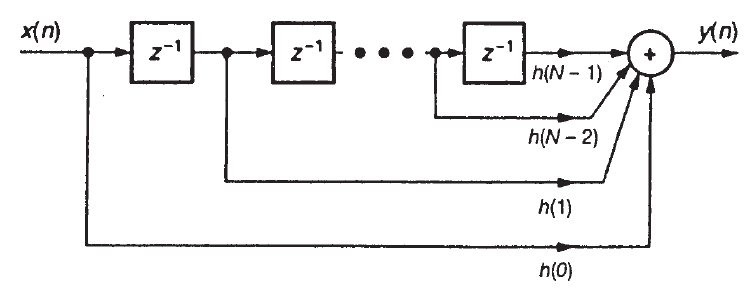
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (2)

Then the z-transform of x(n-1) with respect to first order derivatives dx/dt,

ZT [x(n-1)]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (3)

Where by x (-1) represents the initial condition.



**Figure 1** FIR filter structure showing delays

The input signal to the TI DSK 6713 board is sampled in the AIC23 module. The AIC23 was an ADC section and a DAC section. The DSP reads a unsigned integer 32 (Unit32) value of A/D via the McBsp providing the value present at line input jack. The DSP writes a Uint32 to AIC23 where it contains values for left and right channels. The DSP is designed to do the FIR math very fast, allowing real-time computations to live signals to line input.



**Figure 2**  DSK TMS320C6713 circuit board



**Figure 3** Block diagram of DSK TMS320C6713

The DSK –board device used in this project comes with a wide variety of

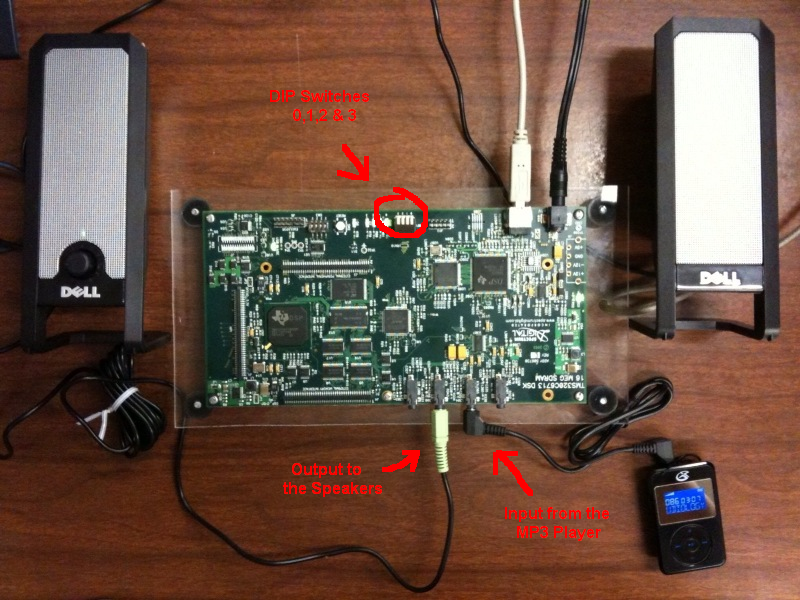
application environments. Key features include:

* A Texas Instruments TMS320C6713 DSP operating at 225 MHz.
* An AIC23 stereo codec
* 16 Mbytes of synchronous DRAM
* 512 Kbytes of non-volatile Flash memory (256 Kbytes usable in default
* configuration)
* 4 user accessible LEDs and DIP switches
* Software board configuration through registers implemented in CPLD
* Configurable boot options
* Standard expansion connectors for daughter card use
* JTAG emulation through on-board JTAG emulator with USB host
* interface or external emulator
* Single voltage power supply (+5V)

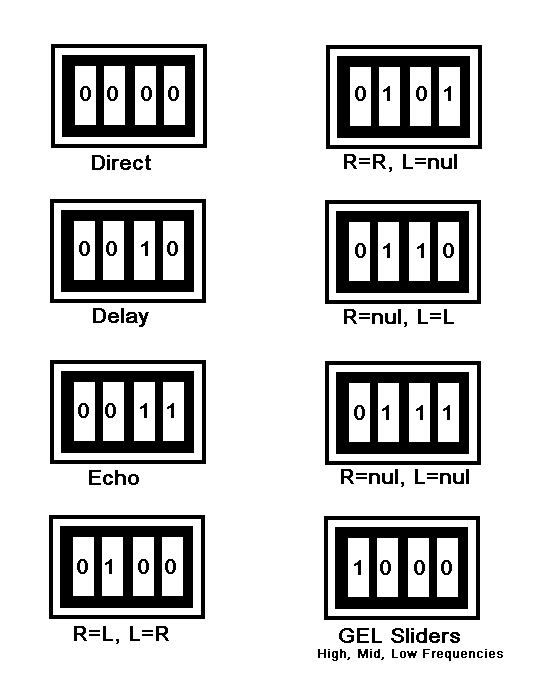
In this report DIP switches are used to send signal for each event based on the C code. The C code is also attached in this report. The switch combination application explains what the C code will do to the signal input.

**Procedures:**

This project started by design of FIR using Matlab tool fdatool as used in lab projects. Refer to example 4.4 from the text book (“*Digital* *signal processing and applications with C6713 and C6714 DSK”* by Rulph Chassaing) and inputting the data shown on the text book. The internet search to obtain starting filter frequency boundaries for audio mixer yielded Low Pass 300Hz, Band pass (LP to HP), and High pass using Frequency cutoff of 5000Hz. Once all the data are input in the fdatool and processed. Matlab generates a data file which is loaded in DSP tool and processed in conjunction with the C code. The C code design combined functions of delay.c, echo.c, and FIR filter sited as text example 4.4. Also integrated some runtime controls such as several parameter controls via GEL extensions and use of the four switches and LEDS to select run mode shown on board below.



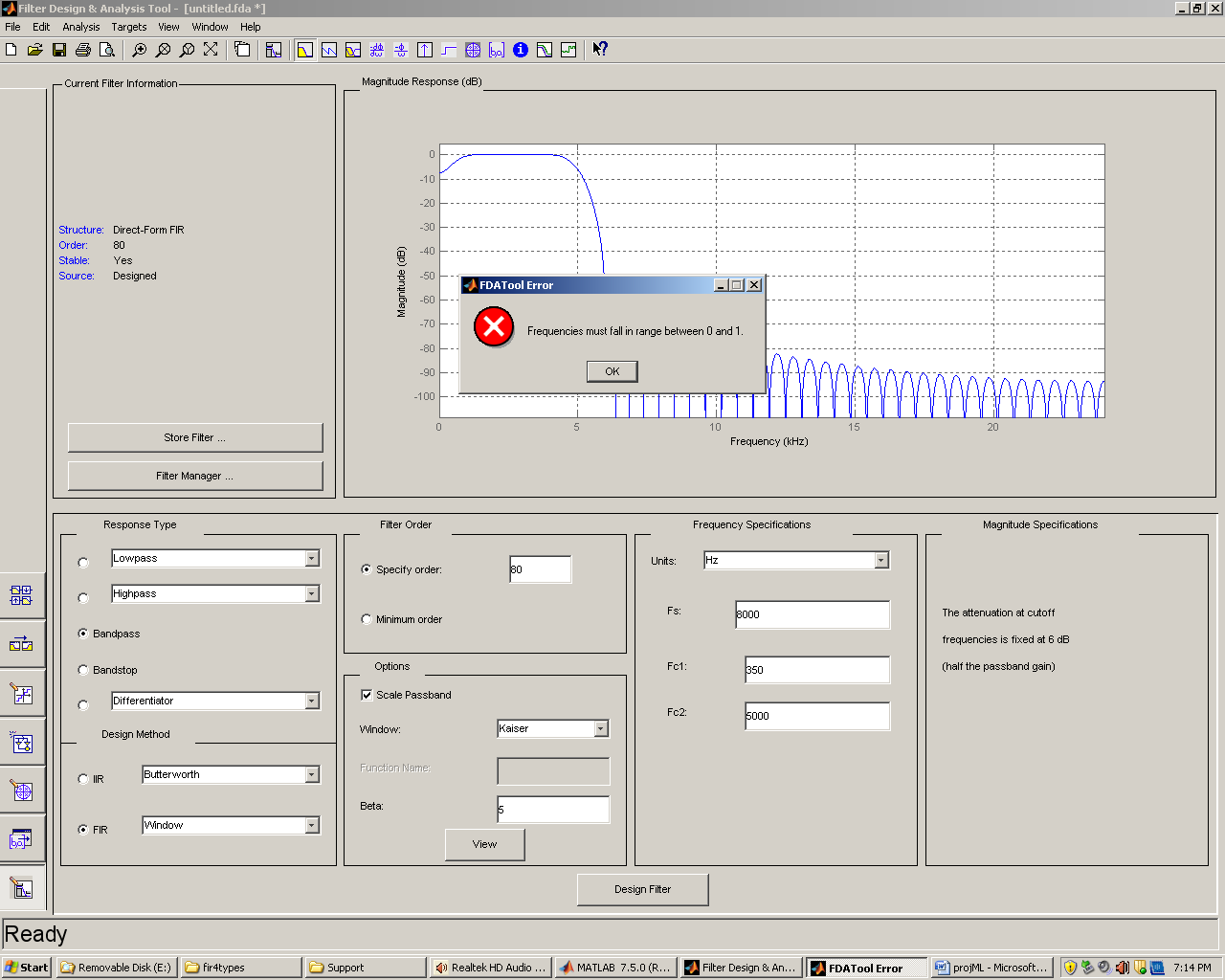
**Figure 4** Project Hardware



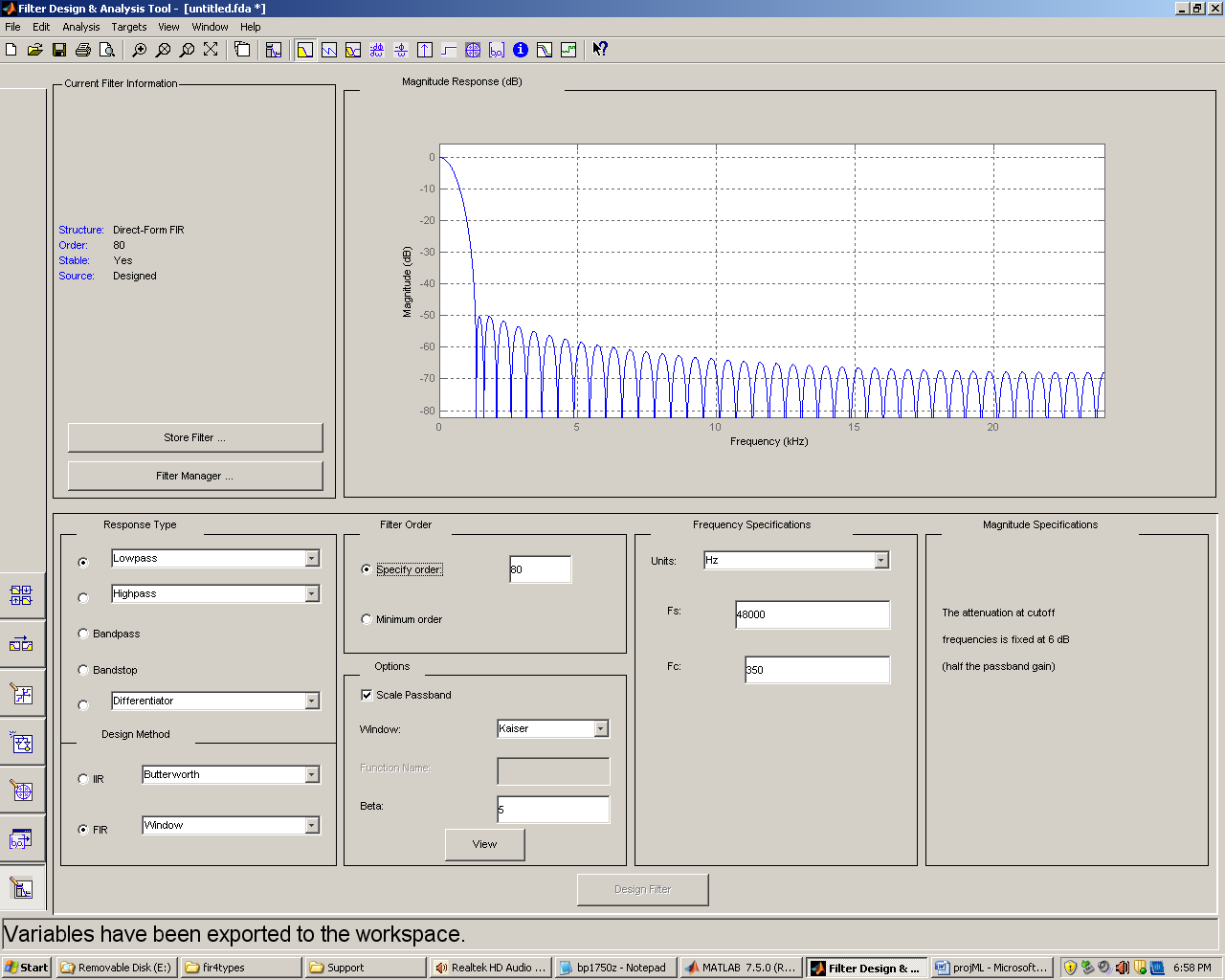
**Figure 5** Switches Combination

**Figures below shows a step by step process used in this project FIR design:**

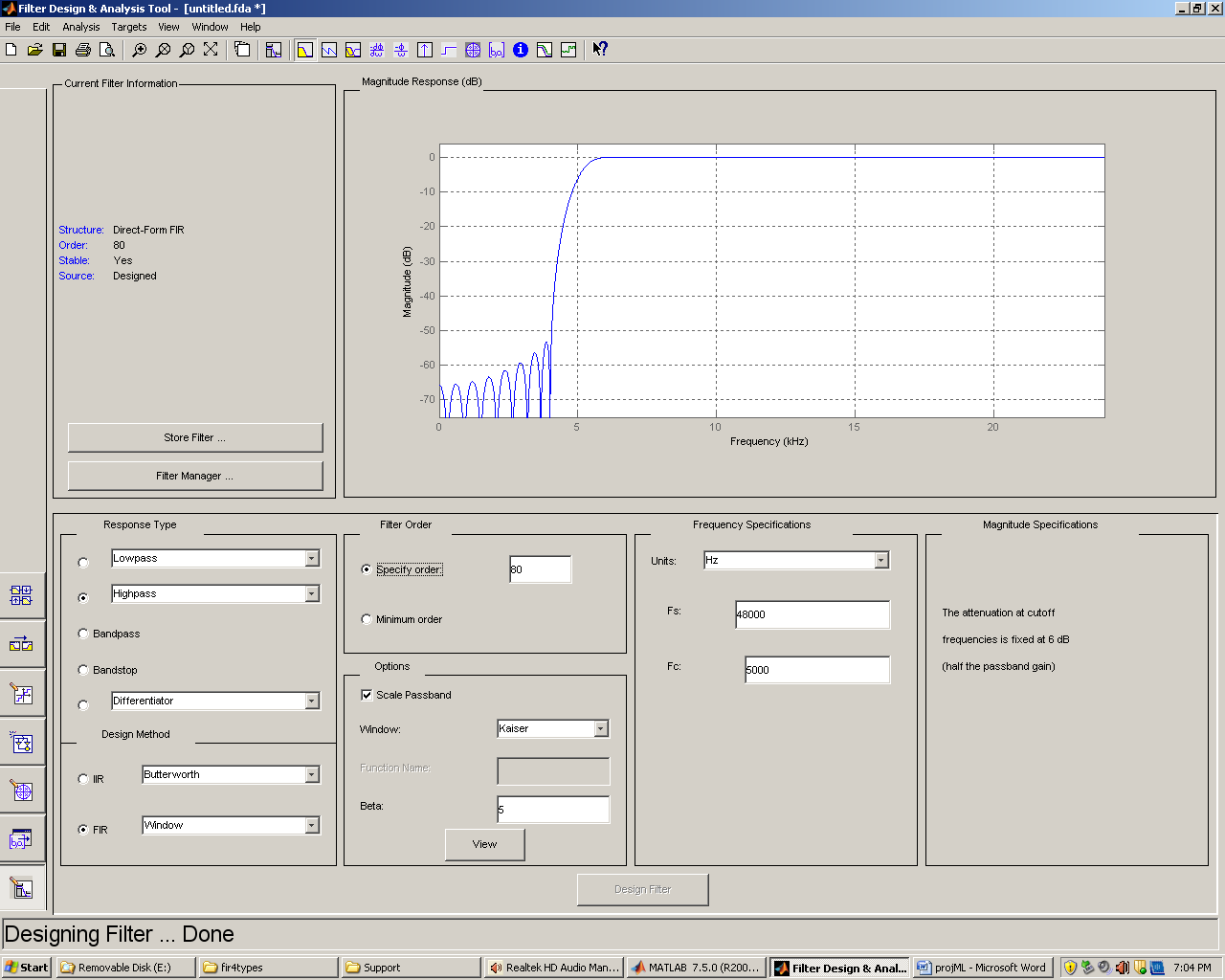
Attempting to use 8Khz sample clock rate shown to be too slow:



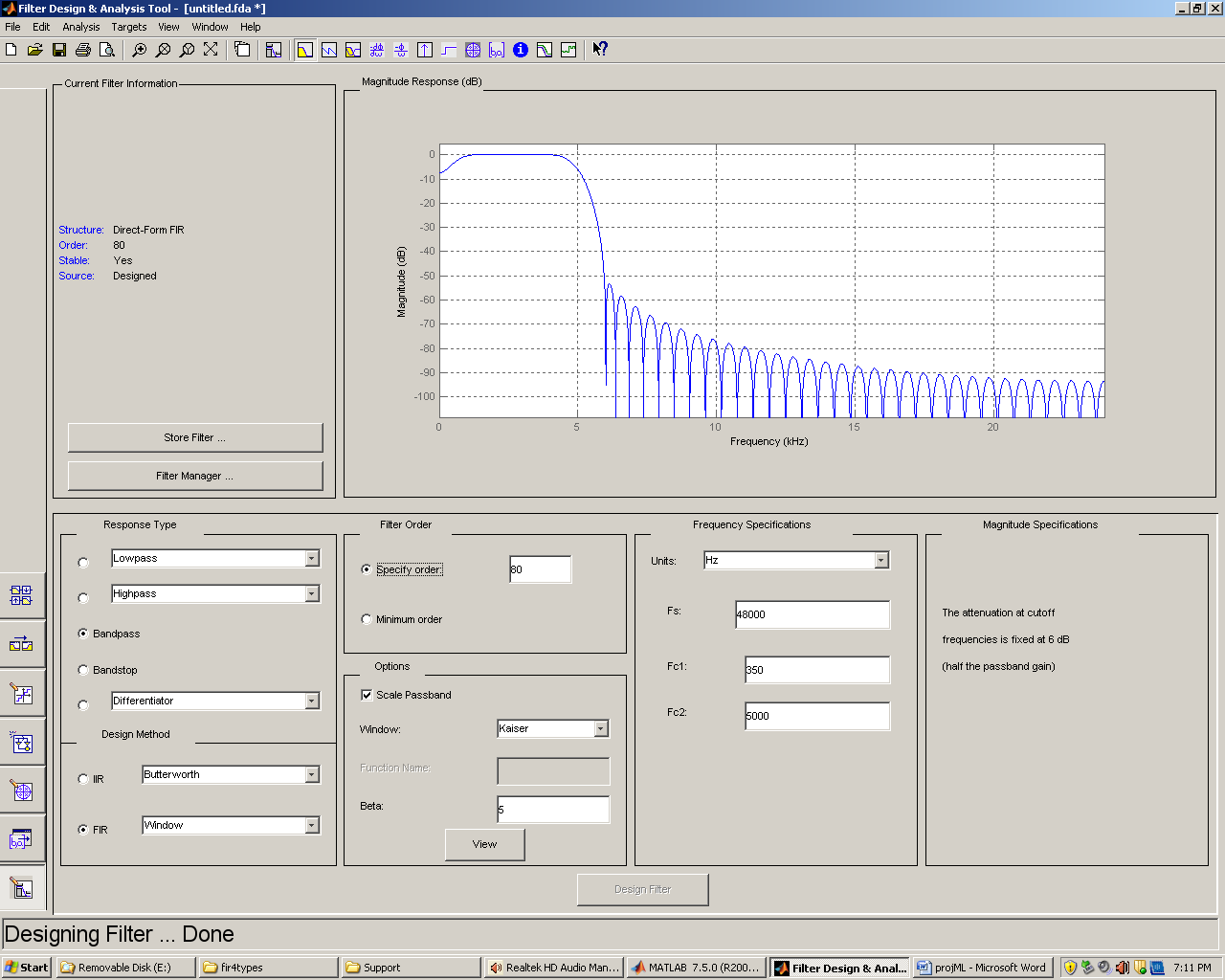
**Figure 6** Matlab tool fdatool generating bandpass signal frequency (F) at 300 to 5000 Hz.  
.hbp350t5kf8 ------ not possible.



**Figure 7** Matlab tool fdatool generating lowpass signal frequency (Fc) at 350 Hz.  
Hlp350f48 -----



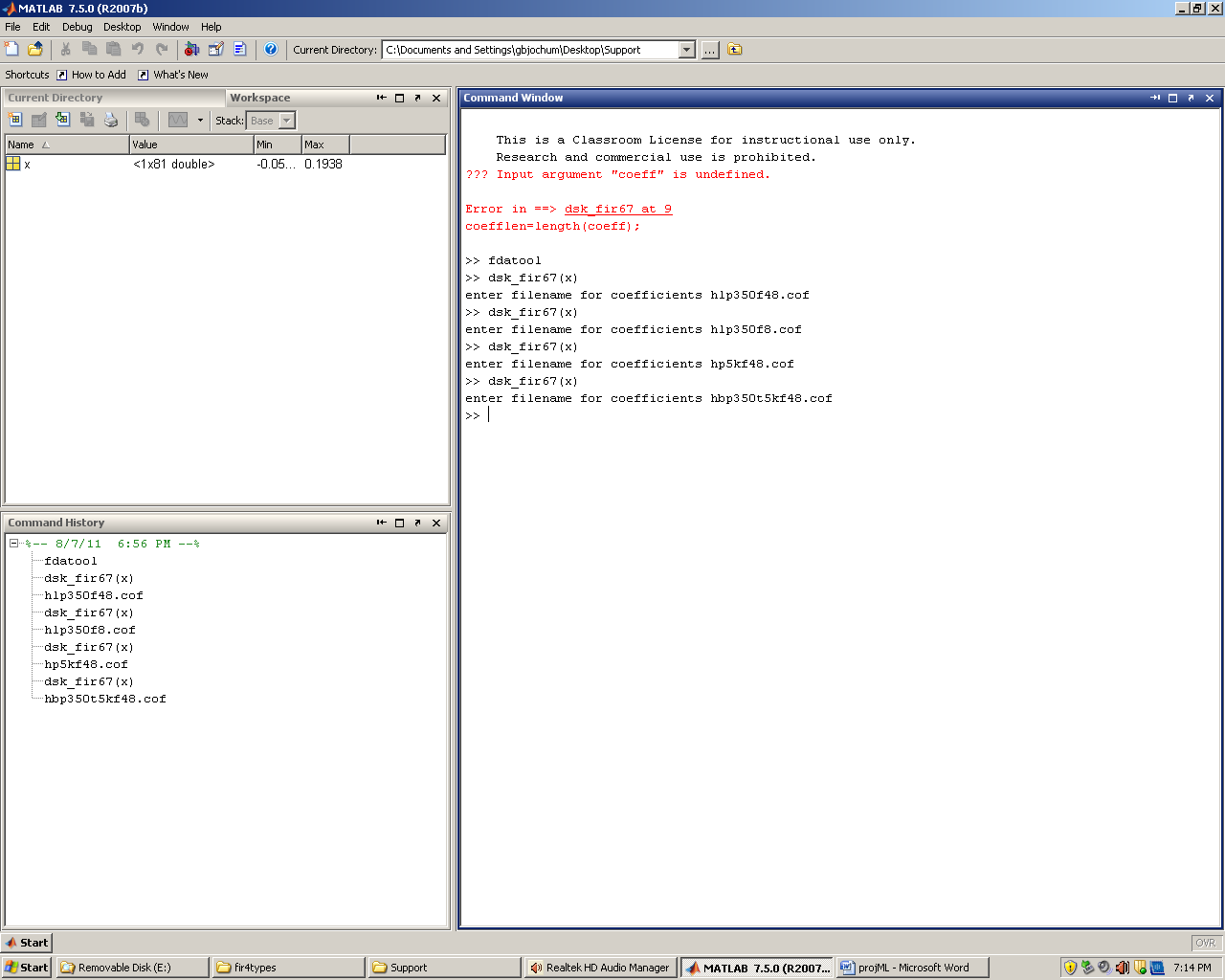
**Figure 8** Matlab tool fdatool generating highpass signal frequency (Fc) at 5000 Hz.  
Hhp5kf48 ------



**Figure 9** Matlab tool fdatool generating bandpass signal frequency (F) at 300 to 5000 Hz.  
.hbp350t5kf48 ------

This is where you can see the window tool is used to generate the signal. The window tool which has it mathematics calculation defined on section 4.6 of the text book helps to transform the rectangular window function

The transform of the rectangular window function yields a sinc function in the frequency domain. These windows help reduce the amplitude oscillations; they provide a more gradual truncation to the infinite series expansion.



**Figure 10** Matlab tool dsk\_fir67(x) to generate COF files each filter designed.

Matlab data generated from the fdatool for three filter groups.

// hbp350t5kf48.cof

// this file was generated automatically using function dsk\_fir67.m

#define N 81

float hbp[N] = {

-2.9197E-005,-2.3742E-004,-6.5416E-004,-1.2003E-003,-1.6797E-003,-1.8546E-003,

-1.5789E-003,-9.2530E-004,-2.2912E-004,5.7745E-006,-6.6975E-004,-2.3840E-003,

-4.7700E-003,-7.0052E-003,-8.1056E-003,-7.3881E-003,-4.9052E-003,-1.6188E-003,

8.4072E-004,8.1276E-004,-2.5823E-003,-8.8179E-003,-1.5871E-002,-2.0834E-002,

-2.1089E-002,-1.5612E-002,-5.8289E-003,4.4701E-003,1.0322E-002,7.5690E-003,

-5.0715E-003,-2.4725E-002,-4.4538E-002,-5.5489E-002,-4.9379E-002,-2.1992E-002,

2.4738E-002,8.2482E-002,1.3845E-001,1.7903E-001,1.9384E-001,1.7903E-001,

1.3845E-001,8.2482E-002,2.4738E-002,-2.1992E-002,-4.9379E-002,-5.5489E-002,

-4.4538E-002,-2.4725E-002,-5.0715E-003,7.5690E-003,1.0322E-002,4.4701E-003,

-5.8289E-003,-1.5612E-002,-2.1089E-002,-2.0834E-002,-1.5871E-002,-8.8179E-003,

-2.5823E-003,8.1276E-004,8.4072E-004,-1.6188E-003,-4.9052E-003,-7.3881E-003,

-8.1056E-003,-7.0052E-003,-4.7700E-003,-2.3840E-003,-6.6975E-004,5.7745E-006,

-2.2912E-004,-9.2530E-004,-1.5789E-003,-1.8546E-003,-1.6797E-003,-1.2003E-003,

-6.5416E-004,-2.3742E-004,-2.9197E-005

};

// hlp350f48.cof

// this file was generated automatically using function dsk\_fir67.m

#define N 81

float hlp[N] = {

4.8441E-004,6.6983E-004,8.8901E-004,1.1445E-003,1.4389E-003,1.7741E-003,

2.1522E-003,2.5748E-003,3.0429E-003,3.5574E-003,4.1186E-003,4.7262E-003,

5.3796E-003,6.0775E-003,6.8180E-003,7.5988E-003,8.4169E-003,9.2688E-003,

1.0150E-002,1.1057E-002,1.1983E-002,1.2924E-002,1.3874E-002,1.4826E-002,

1.5773E-002,1.6709E-002,1.7627E-002,1.8520E-002,1.9380E-002,2.0202E-002,

2.0978E-002,2.1702E-002,2.2368E-002,2.2970E-002,2.3503E-002,2.3962E-002,

2.4344E-002,2.4644E-002,2.4861E-002,2.4991E-002,2.5035E-002,2.4991E-002,

2.4861E-002,2.4644E-002,2.4344E-002,2.3962E-002,2.3503E-002,2.2970E-002,

2.2368E-002,2.1702E-002,2.0978E-002,2.0202E-002,1.9380E-002,1.8520E-002,

1.7627E-002,1.6709E-002,1.5773E-002,1.4826E-002,1.3874E-002,1.2924E-002,

1.1983E-002,1.1057E-002,1.0150E-002,9.2688E-003,8.4169E-003,7.5988E-003,

6.8180E-003,6.0775E-003,5.3796E-003,4.7262E-003,4.1186E-003,3.5574E-003,

3.0429E-003,2.5748E-003,2.1522E-003,1.7741E-003,1.4389E-003,1.1445E-003,

8.8901E-004,6.6983E-004,4.8441E-004

};

// hp5kf48.cof

// this file was generated automatically using function dsk\_fir67.m

#define N 81

float hhp[N] = {

-2.5306E-004,-1.5291E-004,1.3603E-004,5.3320E-004,8.4097E-004,8.2055E-004,

3.2460E-004,-5.7511E-004,-1.5439E-003,-2.0786E-003,-1.7301E-003,-3.7025E-004,

1.6346E-003,3.4627E-003,4.1314E-003,2.9591E-003,-1.9548E-017,-3.7821E-003,

-6.7549E-003,-7.2551E-003,-4.4006E-003,1.2855E-003,7.7843E-003,1.2192E-002,

1.1895E-002,5.8729E-003,-4.4428E-003,-1.5260E-002,-2.1612E-002,-1.9339E-002,

-7.1529E-003,1.2075E-002,3.1496E-002,4.2094E-002,3.5675E-002,8.0257E-003,

-3.8918E-002,-9.6826E-002,-1.5291E-001,-1.9355E-001,7.9187E-001,-1.9355E-001,

-1.5291E-001,-9.6826E-002,-3.8918E-002,8.0257E-003,3.5675E-002,4.2094E-002,

3.1496E-002,1.2075E-002,-7.1529E-003,-1.9339E-002,-2.1612E-002,-1.5260E-002,

-4.4428E-003,5.8729E-003,1.1895E-002,1.2192E-002,7.7843E-003,1.2855E-003,

-4.4006E-003,-7.2551E-003,-6.7549E-003,-3.7821E-003,-1.9548E-017,2.9591E-003,

4.1314E-003,3.4627E-003,1.6346E-003,-3.7025E-004,-1.7301E-003,-2.0786E-003,

-1.5439E-003,-5.7511E-004,3.2460E-004,8.2055E-004,8.4097E-004,5.3320E-004,

1.3603E-004,-1.5291E-004,-2.5306E-004

};

**The above data is from matlab. Note that variable names must be edited to** hhp[N], hlp[N], and hbp[N]

**/\*GJBN\_proj.gel Gel file for 3 different filters: LP,HP,BP\*/**

menuitem "Bhegan & Gerald Filter Characteristics"

slider TestMode(0,1,1,1,testparameter) /\* 0 = normal mode , active effects on both channels (modes 0-7 only)\*/

{

lockright = testparameter; /\* 1 = use copy of right input directly to right channel out \*/

}

slider delay(1,20,1,1,delay\_parameter) /\* can adjust reference delay pointer max of 2000 bytes (modes 2, 3 only)\*/

{

buflength = delay\_parameter\*100;

}

slider gain(0,18,1,1,gain\_parameter) /\* this gain is for amount of echo feedback effect (mode 3) \*/

{

gain = gain\_parameter\*0.05;

}

slider gain\_H(0,18,1,1,gainH\_parameter) /\* amount of this filter output summed into output stream \*/

{

gain\_H = gainH\_parameter\*0.05;

}

slider gain\_M(0,18,1,1,gainM\_parameter)

{

gain\_M = gainM\_parameter\*0.05;

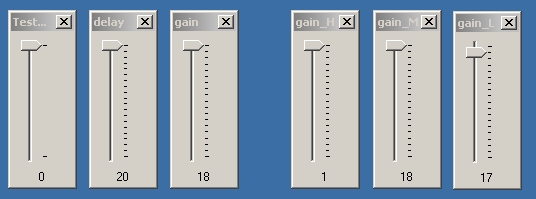
}

slider gain\_L(0,18,1,1,gainL\_parameter)

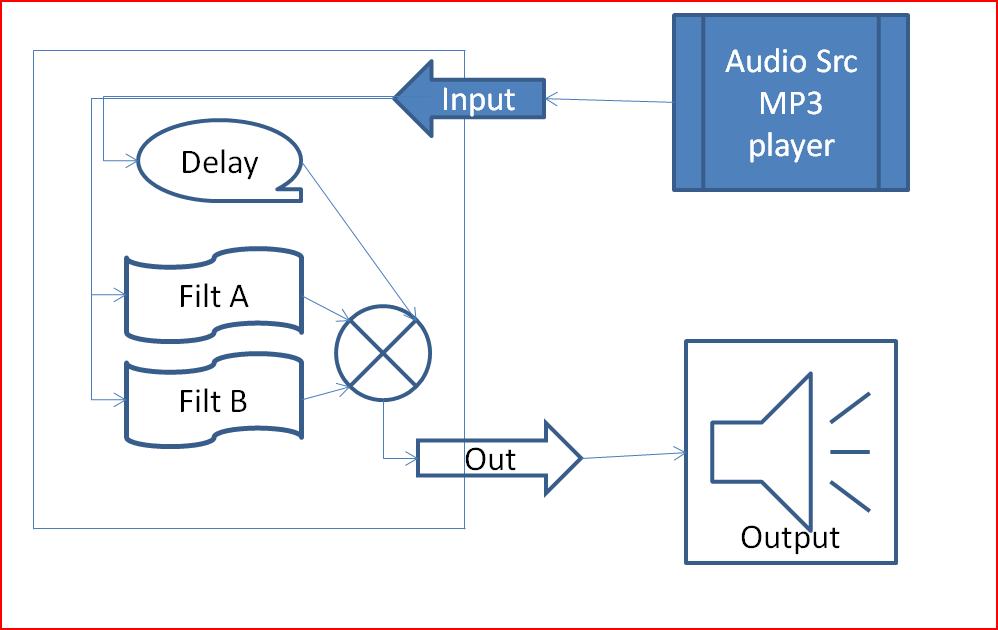
{

gain\_L = gainL\_parameter\*0.05;

}



**Figure 11** Code Composer Studio GEL Controls

**Figure 12** Diagram of Project

Note hhp[N], hlp[N], and hbp[N]array s (cof files) are referenced in C program following.

**Source code used dsk**

**//GJBN\_proj.c 3 FIR filters: Lowpass, Highpass, bandpass**

**#include "DSK6713\_AIC23.h"**

**Uint32 fs=DSK6713\_AIC23\_FREQ\_48KHZ;**

**#define DSK6713\_AIC23\_INPUT\_MIC 0x0015**

**#define DSK6713\_AIC23\_INPUT\_LINE 0x0011**

**Uint16 inputsource=DSK6713\_AIC23\_INPUT\_LINE; // select line in**

**#define LEFT 1 //data structure for union of 32-bit data**

**#define RIGHT 0 //into two 16-bit data**

**union {**

**Uint32 uint;**

**short channel[2];**

**} AIC23\_data;**

**#include "hp5kf48.cof" //coeff file HP @ hp5000 coeff**

**#include "hbp350t5kf48.cof" //coeff file BP @ bp350-5k coeff**

**#include "hlp350f48.cof" //coeff file LP @ lp350 coeff**

**int effect=0;**

**// 0000 = direct , all sw up, group of delay, echo, channel swap or mute selected channels group**

**// 0010 = delay , sw1 down**

**// 0011 = echo , sw1 down & sw2 down**

**// 0100 = R=L, L=R , sw3 down group only does filter routine outputs past here.**

**// 0101 = R=R, L=nul**

**// 0110 = R=nul, L=L**

**// 0111 = R=nul, L=nul (mute)**

**// 1000 = using High. Mid, Low filters, See GEL sliders to adjust**

**#define BUF\_SIZE\_D 8000 // this sets maximum length of delay**

**float gain = 0.5; // fraction (0 - 1) of output fed back**

**#define BUF\_SIZE 2000 // this sets length of delay**

**short hold, L\_input, R\_input,L\_output,R\_output,L\_delayed,R\_delayed;**

**short buflength = 1000;**

**short L\_buffer[BUF\_SIZE\_D ]; // storage for previous samples Left channel**

**short R\_buffer[BUF\_SIZE\_D ]; // storage for previous samples Right channel**

**int j = 0; // index into buffer**

**// cof tables/matrixes sets for left and right**

**short sL, sR; // place holder for values from L & R inputs**

**float gain\_H = 0.5; // fraction (0 - 1) of output fed back (High range)**

**float gain\_M = 0.5; // fraction (0 - 1) of output fed back (Mid range)**

**float gain\_L = 0.5; // fraction (0 - 1) of output fed back (Low range)**

**short lockright=0; // if lockright=0 right=computed, lockright=1 then right=r\_input**

**int yn\_L, yn\_LH, yn\_LM, yn\_LL, yn\_R, yn\_RH, yn\_RM, yn\_RL; //initialize filter's output**

**short LH\_dly[N], LM\_dly[N], LL\_dly[N]; //delay samples per filter (LEFT)**

**short RH\_dly[N], RM\_dly[N], RL\_dly[N]; //delay samples per filter (RIGHT)**

**short dly[N]; //delay samples**

**interrupt void c\_int11() //ISR**

**{**

**short i;**

**i=0;**

**effect =0; // use switch as 0-15 binary number selection integer**

**if(DSK6713\_DIP\_get(0)==0 ) effect +=1; //if SW0 pressed +1**

**if(DSK6713\_DIP\_get(1)==0 ) effect +=2; //if SW1 pressed +2**

**if(DSK6713\_DIP\_get(2)==0 ) effect +=4; //if SW2 pressed +4**

**if(DSK6713\_DIP\_get(3)==0 ) effect +=8; //if SW3 pressed +8**

**AIC23\_data.uint = input\_sample(); //read new input sample**

**L\_input = AIC23\_data.channel[LEFT]; // put in simple L\_input**

**R\_input = AIC23\_data.channel[RIGHT]; // put in simple R\_input**

**if (effect <= 7) {**

**DSK6713\_LED\_on(0); // led 0 on = non filter**

**DSK6713\_LED\_off(1);**

**DSK6713\_LED\_off(2);**

**DSK6713\_LED\_off(3);**

**L\_delayed = L\_buffer[j]; //read output of delay line**

**R\_delayed = R\_buffer[j]; //read output of delay line**

**L\_output = L\_input + L\_delayed; //output sum of new and delayed samples**

**R\_output = R\_input + R\_delayed; //output sum of new and delayed samples**

**if (effect == 0) { //direct pass through**

**L\_output = L\_input; //direct pass through**

**R\_output = R\_input; //direct pass through**

**}**

**if (effect == 2) { //delay.c Basic time delay**

**L\_buffer[j] = L\_input; // update delayed sample**

**R\_buffer[j] = R\_input;**

**}**

**if (effect == 3) { //echo\_control.c echo with variable delay and feedback**

**L\_buffer[j] = L\_input + L\_delayed\*gain; // store new input and a fraction of the delayed value in buffer**

**R\_buffer[j] = R\_input + R\_delayed\*gain; // use GEL controls at runtime**

**}**

**if (effect == 2) { //delay.c Basic time delay end of buffer testing**

**if(++j >= BUF\_SIZE) // test for end of buffer mode=2**

**j = 0;**

**} else {**

**if(++j >= BUF\_SIZE\_D) // test for end of buffer mode=3**

**j = BUF\_SIZE\_D - buflength;**

**}**

**if (effect == 4) {**

**hold = L\_output; // swap left & right channels**

**L\_output = R\_output;**

**R\_output = hold;**

**}**

**if (effect == 5) L\_output=0; // muting options**

**if (effect == 6) R\_output=0;**

**if (effect == 7) {**

**L\_output=0;**

**R\_output=0;**

**}**

**if (lockright==1)R\_output = R\_input; // case =1, pass input directly to output**

**AIC23\_data.channel[LEFT] = L\_output; //post value to use**

**AIC23\_data.channel[RIGHT] = R\_output;**

**output\_sample(AIC23\_data.uint); //output to both channels**

**// 0000 = direct**

**// 0010 = delay**

**// 0011 = echo**

**// 0100 = R=L, L=R**

**// 0101 = R=R, L=nul**

**// 0110 = R=nul, L=L**

**// 0111 = R=nul, L=nul (mute)**

**} // if non filter**

**else DSK6713\_LED\_off(0); // led 0 off = filter**

**if (effect >= 8) { // Three FIR filters: Lowpass, Highpass, bandpass**

**DSK6713\_LED\_on(3); // led 3 on = filter**

**LH\_dly[0] = L\_input; //newest input @ top of buffer[each is independent fn]**

**LM\_dly[0] = L\_input; //**

**LL\_dly[0] = L\_input; //Left side done**

**RH\_dly[0] = R\_input; //**

**RM\_dly[0] = R\_input; //**

**RL\_dly[0] = R\_input; //Right side is done**

**yn\_L =0; //initialize filter's working output (8 places)**

**yn\_R =0; // summing var 2 places**

**yn\_LH =0; // Left side**

**yn\_LM =0;**

**yn\_LL =0;**

**yn\_RH =0; // Right side**

**yn\_RM =0;**

**yn\_RL =0; //initialize filter working output (section done)**

**for (i = 0; i< N; i++)**

**{**

**yn\_LH +=( hhp[i]\*LH\_dly[i]); //y(n) += h(LP#,i)\*x(n-i) [repeats 6 times]**

**yn\_LM +=( hbp[i]\*LM\_dly[i]);**

**yn\_LL +=( hlp[i]\*LL\_dly[i]); // Left done**

**yn\_RH +=( hhp[i]\*LH\_dly[i]); // Right begin**

**yn\_RM +=( hbp[i]\*LM\_dly[i]);**

**yn\_RL +=( hlp[i]\*LL\_dly[i]); // Right done**

**}**

**yn\_L = (yn\_LH \*gain\_H) + (yn\_LM \*gain\_M) + (yn\_LL \*gain\_L); // build sum of 3 Inp**

**yn\_R = (yn\_RH \*gain\_H) + (yn\_RM \*gain\_M) + (yn\_RL \*gain\_L);**

**for (i = N-1; i > 0; i--) //starting @ bottom of buffer**

**{**

**LH\_dly[i] = LH\_dly[i-1]; //update delays with data move [6 places begin]**

**LM\_dly[i] = LM\_dly[i-1];**

**LL\_dly[i] = LL\_dly[i-1];**

**RH\_dly[i] = RH\_dly[i-1];**

**RM\_dly[i] = RM\_dly[i-1];**

**RL\_dly[i] = RL\_dly[i-1]; //update delays with data move [6 places done]**

**}**

**sL=yn\_L; // move to cast short , maybe fix error of audio output**

**sR=yn\_R;**

**AIC23\_data.channel[LEFT] = sL;**

**AIC23\_data.channel[RIGHT] = sR; // R\_input; //**

**output\_sample(AIC23\_data.uint); //output to both channels**

**return; //return from interrupt**

**} // was fir filter**

**}**

**void main()**

**{**

**short i;**

**for(i=0 ; i<BUF\_SIZE\_D ; i++) {**

**L\_buffer[i] = 0; // zero buf at beginning**

**R\_buffer[i] = 0;**

**}**

**for (i=0; i<N; i++)**

**{**

**LH\_dly[i] = 0; //newest input @ top of buffer[each is independent fn]**

**LM\_dly[i] = 0; //**

**LL\_dly[i] = 0; //Left side done**

**RH\_dly[i] = 0; //**

**RM\_dly[i] = 0; //**

**RL\_dly[i] = 0; //Right side is done**

**}**

**comm\_intr(); //init DSK, codec, McBSP**

**while(1); //infinite loop**

**}**

**Testing results**

Setup test environment as shown in diagram prior, used an MP3 player as source to Line input and amplified stereo speakers we could hear our modified audio when program loaded and running.

Reading switches 1-4 as a binary mode selection allowed easy change of AV application operational mode during testing and operation. Early tests were verifying could read both channels and mute any combination presented to line output jack. Verified we still got output when either delay or echo modes set active. Testing modes greater than 7 where activating the LP, BP, and HP filters bought challenges to learn why there was no output.

When changing from prebuilt COF sample to ones made in MatLab, experienced no output.  
This was fixed over time. The GEL sliders are also very beneficial way to vary operational parameters. The Gel sliders used most were the gain\_L, gain\_M, and gain\_H to cause changes to audio output by frequency groups.

**Conclusion.**

This project implementation taught some first hands-on of the challenges of taking a concept implementation and problem solve short comings. The concept depicted in diagram seemed easy to achieve in some ways. Class labs and text examples were valuable to define this project in a shorter amount of time. The use of nearly all lab assignments and more combined into one product required thought to regroup each sub task so when assembled, each module did not adversely affect other modules. The most challenging problem to solve was not the design implementation but not expecting the variable type changed from MatLab process. The example files were included, so the fact that the prior COF files contained arrays of integers, the new 48 KHz FIR COF files containing floats took a few lab runs before the awareness the code was not the issue but the float/integer type change. Answer was simple; however most helpers said fault was in code hence spent most debug time there. The overall learning experience of hands on implementing each of these modules and being empowered to problem solve our issues, was priceless to the participants of the project.

**References**

ECE 671 text book *Digital* *signal processing and applications with C6713 and C6714 DSK* by Rulph Chassaing

<http://www.comlab.uni-kassel.de/cms/fileadmin/Diplom/ProjectKhan.pdf>

<http://ethesis.nitrkl.ac.in/71/1/10407030.pdf>