

Experiment 3: Implementing filters using DSP Board

1 Introduction

This lab will provide an introduction to the Analog Devices EZ-KIT LITE DSP board: a few simple demos on the board will be covered, and implementation of different kinds of FIR filters such as low pass, high pass, band pass, and band stop will be done. A brief explanation of FIR filters will be given. These filters will be designed using the Remez exchange algorithm function available in the MATLAB signal processing toolbox. The filter coefficients generated by the MATLAB file will be written to a separate file. This output file in turn will be inserted into a C program which will be compiled and downloaded to the DSP board. Audio signals and signals generated by a function generator will be filtered through the board and the output will be sent to a speaker and observed on a spectrum analyzer.

2 Analog Devices DSP board

The DSP board has a SHARC processor manufactured by Analog Devices and can sample up to 48 kHz. The following steps will guide you through the setup and testing of the board.

- First, provide power to the board using the appropriate DC transformer.
- Look for three small yellow buttons in the center of the board. One of them is labeled reset. Press it.
- Connect the stereo output of the board to a set of speakers. If the previous steps were successful, you should here the infamous Peter Gunn Theme.
- Now, ensure that the serial cable is connected from the board to the computer. Using the computer go to the Start Menu. Then go to Programs > SHARC EZ-KIT Lite and click on EZ-KIT Lite Host.
- Once you are in the EZ-KIT Lite Host window, go to File and then Demos. Choose the Bandpass filter demo.
- Pop an audio CD into the computer's CD-ROM. (Hopefully, there will be some CD's in the lab.) Next Connect the computer's CD player to the input terminal of the board using a 1/8 inch stereo headphone jack.

- Choose either Noise or Codec. The Codec option will allow you to listen to the audio input where as the Noise option will allow you to listen to internally generated noise. You can choose different pass bands and listen to the changes in the sound.
- You're welcome to try any of the other fabulous demos.

3 FIR filters

One class of filters that do not use output feedback are nonrecursive and hence have a finite impulse response (FIR). One form of FIR structure is obtained by implementation of the discrete convolution:

$$y(n) = \sum_{m=-\infty}^{\infty} h(m)x(n-m). \quad (1)$$

where $x(n)$ is the input signal, and $h(n)$ is the FIR filter impulse response.

If the impulse response $h(m) = 0$ for $m < 0$ and $m \geq N$, then equation (1) becomes

$$y(n) = \sum_{m=0}^{N-1} h(m)x(n-m). \quad (2)$$

This is a difference equation, but it is nonrecursive as $y(n-m)$ does not appear on the right. The Z transform of equation (2) has all its poles at the origin; so it must be a stable system. The length of the impulse response is N samples.

The Z transform of any filter has the form

$$H(z) = \frac{B(z)}{A(z)}. \quad (3)$$

For an FIR filter, $A(z) = 1$, since it does not have poles in the Z-plane other than at the origin. However it has N-1 zeros. So the equation for the Z transform of the impulse response can be written as

$$y(n) = \sum_{m=0}^{N-1} h(m)z^{-m}. \quad (4)$$

We can design the FIR filter such that $H(z)$ is a lowpass, highpass, bandpass, or bandstop filter.

4 MATLAB file

The required MATLAB file is already written for you. It is in C:\EZ-KIT\EZSHARC\tt. The name of the file is dsp_filter.m. Open the file and read the program. As you will see, the first line of the program is **function dsp_filter(N,F,A,file_name)**. When you are in MATLAB under C:\EZ-KIT\EZSHARC\tt, call the function by typing **dsp_filter(...)**. The information in the parentheses has to be entered by you.

The first parameter, **N**, is the order of the filter. This can be entered as a number. The next two parameters, **F** and **A**, are the frequency and the amplitude arrays respectively of the Remez exchange algorithm. They both need to have the same number of elements. The last parameter, **file_name**, can be called whatever you want. This **must** be entered in single quotes. This will produce the required output file with that name in the same directory.

If you want to construct a low pass or a high pass filter, the arrays need to have four elements each, where as for a band pass or band stop filter, the arrays need to have six elements each.

4.1 Low Pass

- For a low pass filter, the F array needs 4 numbers:

[**0, start of transition band, stop of transition band, 1**]

The start and stop of the transition band are in percentages of the Nyquist frequency which is denoted by the 1. So these numbers have to be represented as fractions. For example, entering [**0 .5 .6 1**] for the F array causes a transition band that starts at 50% of the Nyquist frequency and ends at 60% of the Nyquist frequency.

- The A array also needs 4 numbers. For a low pass filter, it should be [**1 1 0 0**]. This denotes the magnitude of the low pass filter at zero frequency and the starting frequency of the transition band as 1, and the magnitude of the low pass filter at the stopping frequency of the transition band and the Nyquist frequency as 0.
- So the appropriate command to type in MATLAB for implementing the above filter (see Figure 1 on page 8) as a 50th order filter and output filename 'lowpass' would be
dsp_filter(50, [0 .5 .6 1], [1 1 0 0], 'lowpass')

4.2 High Pass

- For a high pass filter, the F array is the same as it is for the low pass filter.
- The only difference in the A array is that the magnitudes are reversed. It should be [**0 0 1 1**]. The first two magnitudes are 0 and the last two are 1 which is quite obvious. The filter should not pass any low frequencies and pass all frequencies above the cutoff.
- So the appropriate command to type in MATLAB for implementing a 50th order high pass filter (see Figure 2 on page 8) and output filename 'highpass' would be
dsp_filter(50, [0 .5 .6 1], [0 0 1 1], 'highpass')

4.3 Band Pass

- For a band pass filter, the F array needs 6 numbers:

[0, start of tr_band1, stop of tr_band1, start of tr_band2, stop of tr_band2, 1]

As you know, the band pass filter has two transition bands. The four start and stop frequencies are entered as fractions of the Nyquist. For example, entering [0 .2 .3 .7 .8 1] will produce the first transition band from 20% to 30% of the Nyquist, and the second transition band from 70% to 80% of the Nyquist.

- Similarly the A array has 6 numbers: [0 0 1 1 0 0]. For a band pass, the magnitude should be 1 at the stop of the first transition band and at the start of the second transition band, and 0 everywhere else.
- So the appropriate command to type in MATLAB for implementing a 50th order band pass filter with the above characteristics (see Figure 3 on page 9) and output filename 'bandpass' would be

```
dsp_filter(50, [0 .2 .3 .7 .8 1], [0 0 1 1 0 0], 'bandpass')
```

4.4 Band Stop

- For a band stop filter, the F array is the same as it is for a band pass filter.
- The A array has 6 numbers: [1 1 0 0 1 1]. For a band stop, the magnitude should be 0 at the stop of the first transition band and at the start of the second transition band, and 1 everywhere else.
- So the appropriate command to type in MATLAB for implementing a 50th order band pass filter with the above characteristics (see Figure 4 on page 9) and output filename 'bandstop' would be

```
dsp_filter(50, [0 .2 .3 .7 .8 1], [1 1 0 0 1 1], 'bandstop')
```

After the function has been called, the program will compile and two plots will appear. One will have the magnitude of the filter in dB and linear scale. The other shows the coefficients of the impulse response of the filter. Study these plots carefully.

Look at the coefficients of the impulse response of the low pass filter. What is the shape of the impulse response? Can you see the relationship between the filter and impulse response? Are the impulses centered around the origin? If yes or no, why do you think this is?

Re-run the MATLAB function for any kind of filter, but make the transition bands finer. For example, for a low pass filter, enter [0 .5 .55 1] as the F array instead of [0 .5 .6 1]. Do you notice a change in the magnitude plots? Why do you think this happens? How can you fix this problem?

5 C code and compilation

After running the MATLAB function, make sure an output file with a '.h' extension of the filename that you denoted in the function call has been created in the directory. You can view this file in Notepad, and you will see a list of numbers created. These are the required coefficients of your filter that are going to be used by the C program. Close this file **without** modifying anything. This is very important. If you do modify something by mistake, make sure you do not save changes as you will have to re-run your MATLAB function `dsp_filter`.

Under the same directory, open the file `tt.c`. After scrolling down a little you will come to the section where you will see the following text

```
#define NUM_TAPS 129  
  
float pm coeffs[NUM_TAPS] =  
{  
#include "lpf.h"  
};  
  
float dm state[NUM_TAPS+1];
```

Change "lpf.h" to "filename.h". So basically change the code to include the name of your filename that you denoted in the function call in MATLAB instead of lpf.h. Also change the NUM_TAPS to 51. The NUM_TAPS is defining the number of coefficients and it always has to be one more than the order you defined in the MATLAB function. Why? Save the file `tt.c` and close it.

In `C:\...\tt`, you will see a file called `Mk.bat`. Double click on this file. This will invoke a DOS window and compile the `tt.c` file. Close the DOS window after it has finished compiling.

Now, just go to the EZ-KIT Lite Host window. Go to File and then Demos. Choose the talkthru demo. You should hear the filtered sound from your audio source. You can play around with different values of the sample rate and filter gain and listen to the difference in the sound with different sampling rates. The filter gain will just increase the amplitude of the sound.

After you have done the procedure with a certain type of filter, go back and re-run the MATLAB function with a different filter with different characteristics. Make sure you change the filename every time. Go through the rest of the steps mentioned above and you will hear distinct differences in the sound with the different types of filters.

6 Function Generator

Instead of using an audio input, we are now going to use a sinusoid generated by the function generator. Choose a sinusoid of 200mV amplitude and 12 kHz frequency. Connect the output of the function generator to the input of the DSP board. The input to the board has a 1/8" stereo headphone jack connection where as the function generator output has a BNC connection. You have to use the appropriate conversion device, which should be already provided to you. Ask if you do not have this.

Change the filename in the `tt.c` file to `bandpass.h` and change the `NUM_TAPS` to 129. Re-run the MATLAB function as a band pass filter that has a pass band from about 40% to 60% of the Nyquist. So in MATLAB, type the following command:

```
dsp_filter(128, [0 .4 .45 .6 .65 1], [0 0 1 1 0 0], 'bandpass')
```

Double click on the `Mk.bat` file and run the Talkthru demo in the EZ-KIT Lite Host window. You should be able to hear the 12kHz sinusoid as a constant frequency tone. If you cannot, then increase the input gain from 0 dB to something higher until you hear the tone.

Notice that the sampling frequency on the talkthru demo GUI is 48 kHz. What do you think the Nyquist frequency is? Before you proceed, make sure you find out what it is. Slowly change the frequency of the sinusoid from the function generator. Increase it to beyond 15 kHz and you should not be able to hear the tone anymore. Now, decrease the frequency to below 9 kHz and the sound should again fade away. Is this correct? Repeat this procedure with different combinations of filters and Nyquist frequencies. To change the Nyquist frequency, simply click on a different value of the sampling frequency on the GUI. To change the filter characteristics, you will have to go through all the necessary steps mentioned above. You should try all four types of filters (LP, HP, BP, BS).

7 Spectrum Analyzer

We will use the SR770 spectrum analyzer to see the magnitude spectrum of these filters.

- Before you set up the spectrum analyzer, create a 128th order low pass filter in MATLAB with the cutoff frequency set to about 50% of the Nyquist. Change the relevant information in the `tt.c` file and recompile the `Mk.bat` file.
- Disconnect the output of the DSP board from the speakers. Connect the output of the DSP board to the **A** input of the spectrum analyzer using a BNC - 1/8" stereo headphone jack connection. Using another BNC - 1/8" stereo headphone jack connection, connect the input of the DSP board to the **Source Out** of the SR770.

- Turn the SR770 on holding the backspace [←] key until the Calibrating offset sign appears on screen.
- Hit the **Span** soft key and lower it to **25 kHz** using the knob.
- Hit the **SOURCE** button under the **MENU** list. Hit the **Noise** soft key. This turns on the internally generated noise in the SR770 which contains frequencies from 0 to 100 kHz. Hit the **FREQ** button under the **MENU** list to return to the frequency menu.
- In the computer run the talkthru demo.
- Under the **ENTRY** list, hit the **AUTO RANGE** button followed by the **AUTO SCALE** button. You should be able to see the magnitude spectrum of the filter now.
- Now, hit the **AVERAGE** button under the **MENU** list. Click on the **Number Averages** soft key and change this value to **600**. Right above the **Number averages** option, there should be the **Averaging** option and this should be Off. Turn this **on** by hitting that soft key. It will take a few seconds to do this.
- Press the **MAX\MIN** button under the **MARKER** list. Using the knob, measure the frequency at which the filter falls off. It should be around 12 kHz as the Nyquist frequency is 24 kHz and you designed the filter with the cutoff frequency set to about 50% of the Nyquist.
- Repeat the above steps for a band pass, band stop, and high pass filter and compare the characteristics of the filters you see in the SR770 with the actual MATLAB plots of the magnitude spectrums. Every time you want to re-average, you do not need to go to the **AVERAGE** menu. Simply hit the **START** button under the **CONTROL** list.
- When you generate a high pass or a band stop filter, you will see that the spectrum drops off at high frequencies even though ideally this is not supposed to happen. What is the reason for this? You may not be able to find the answer unless you know a little about the hardware of the DSP board. Make sure you find out.

8 Design Work

Design and implement a filter (on the DSP board) that will pass only the third harmonic of a 1kHz square wave (i.e. the harmonic at 3kHz). Justify your choices for the sampling rate, filter order, pass band, transition band, and stop band. Include all relevant Matlab plots and data from tests.

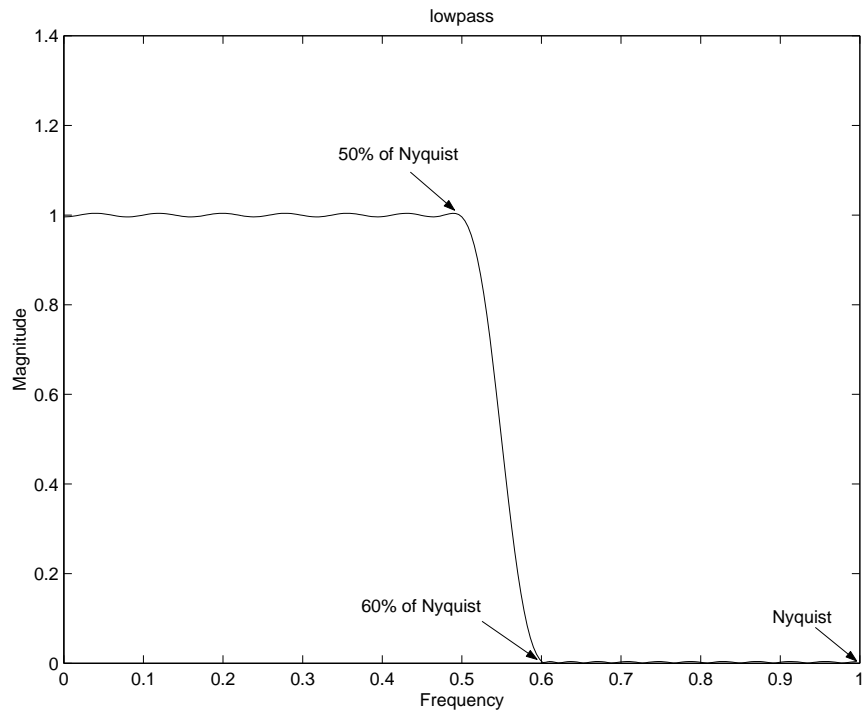


Figure 1: Low Pass Filter

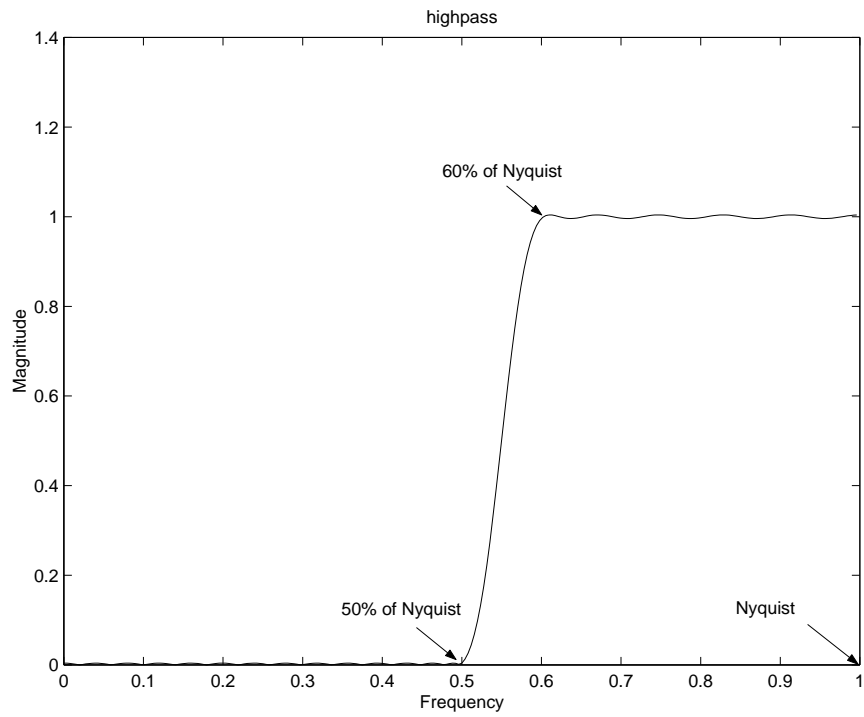


Figure 2: High Pass Filter

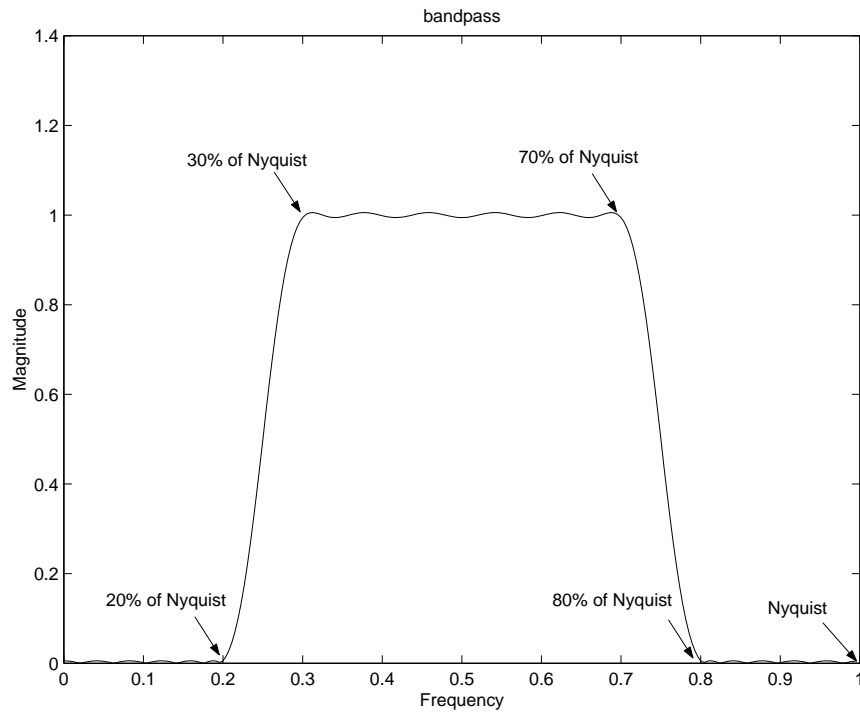


Figure 3: Band Pass Filter

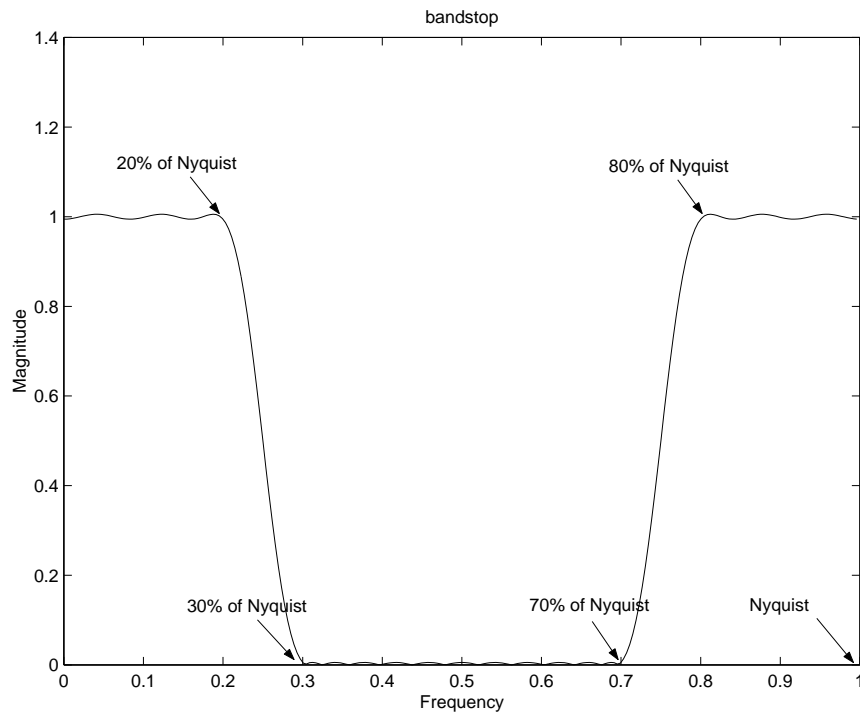


Figure 4: Band Stop Filter