Lab 07 Frame Based Filtering Tutorial

Read chapter 7. It won’t take long. The book and the code from the book you will use are for implementing an FIR filter using frames of data. There are a few gotchas that are explained in the book.

**1. First Assignment:** Turn in the whole project you generated for this task.

The first task is to implement a **Low Pass FIR filter**. Use the code from Code/Chapter\_07/CCS/FiltFrm\_6748 as a start. Make sure not to put the existed filter coefficient files(COEFF.C, COEFF.H) into your project. You will generate your own.

**A. Use the fdatool in the MATLAB to design a low-pass FIR filter**

(1) Filter order: 32nd

(2) Cutoff frequency: 3KHz

(3) Sample frequency of 48KHz

(4) Export a workspace: Num

(5) Use the MATLAB function FIR2C.m (This MATLAB code is available on the course website from Chapter 3) to generate your c. and .h files, which will be used and added to your new CCS project

**B. Create your CCS project**

(1) Add all necessary files to your project

(2) Add c. and h. files you just created to your project

(3) Go to Book3rdEdition/Code/Chapter\_07/CCS/FiltFrm\_6748, add ISRS.c, main.c, and frames.h to your project (Don’t forget to include your own .h file at the beginning of the ISRs.c)

(4) Add configuration files to your project

**C. Modify the code**

Implement the FIR filter on the left channel and talk through on the right channel.

(1) Use a buffer length of 256 samples.

**2. Second** **Assignment**: Turn in the whole project you generated for this task.

The second task is to implement an IIR Low Pass filter. There is no code for the IIR filter in chapter 7.

**A. Use the fdatool to design a 4th order Butterworth IIR filter.**

(1) Filter order: 4th order

(2) Cutoff frequency: 3KHz

(3) Sample frequency of 48KHz.

(4) Structure: DFIIT SOS

(5) Export workspaces: SOS and G

(6) Use the MATLAB function SOS2C.m (This MATLAB script is available on the course website from Chapter 4) to generate your c. and .h files, which will be used and added to your new CCS project

Since this is a frame based filter, you won’t use the code from Chapter 4. In fact you are in luck. There is a subroutine written for the Biquad function you used in Chapter 4. It implements a Direct Form II Transform second order section for frames of data.

The name of the subroutine is DSPF\_sp\_biquad.h. It is described in the help files you will download. Do that now. The “Required TI DSPLib 3.4.0.0 files …” link on the course website contains everything required as the name implies to use the filter as well as the help files. Download and unzip this file to a folder of your choice.

You can’t install the library on a CAE computer so I extracted the files you need and put them in the zip file. The library installation file is supplied in case you use a computer you have access to. If you install the library you can use all of the functions.

I suggest you look at the pseudo C code version of the subroutine. It is available on the course website. (Take a look at the IIR filter function, it might be helpful.) The subroutine will be used directly, thus you will not need the #pragma lines for your own C code. It is shown just to show how it works and how the data needs to be setup before calling it.

This subroutine works on a frame of input data. It generates a frame of output samples. You have to supply the b0, b1 and b2 coefficients in a vector and the a1 and a2 coefficients in a vector. The state variables are in a vector too. The subroutine refers to them a delay. The final parameter that you call with the subroutine is the number of points in the frame. The important thing that is not given in the help file is that all of the vectors need to be double word aligned. Since you probably don’t know what that means or how to do it I am going to tell you now.

Place these commands in the C file just before you define the vectors:

#pragma DATA\_ALIGN(x, 8);

#pragma DATA\_ALIGN(y, 8);

#pragma DATA\_ALIGN(b, 8);

#pragma DATA\_ALIGN(a, 8);

#pragma DATA\_ALIGN(states, 8);

All pragma statements in one place works fine.

So we’re ready to go right. Not yet.

**B. Create your CCS project**

(1) Add all necessary files to your project

(2) Add c. and h. files you just created for your IIR filter to your project

(3) Add configuration files to your project

Since we are not using a C file for the subroutine but rather a library we need to tell CCS to load the library. This is easy. If the compiler doesn’t find the routine in the project proper it will look in libraries. The names of libraries that are installed are available in the properties of the project.

(4) Go to the project properties.

Under the File Search Path you will find the library and path lists that you can add your info too. The library you want to add is dsplib.lib which will be in the directory you unzipped the downloaded files too.

a. In the “Include library file or command file as input” window, add your library:

click add button, browse the file path, find Biquad\dsplib.lib, click OK then.

b. In the “Add <dir> to library search path” window, add the directory to your library:

click add button, browse the file path, find the directory to Biquad folder, click OK then.

c. Click “Apply and Close”

(5) Now you are ready to use the DSPF\_sp\_biquad library. Write the function that is similar to the IIR biquad code from chapter 4 with the appropriate changes made for the frames.

a. Add the ISRs.c, main.c, and frams.h to your project

Go to Book3rdEdition\code\chapter07\ccs\FiltFrm\_6748 and add those files.

b. Implement both left and right channels

Check any **stereo**\_ISRs.c files, compare with **mono**\_ISRs.c, see how to implement both left and right channels.

c. Implement the IIR filter on the left channel and talk through on the right channel.

d. Use a frame buffer length of 12 samples. One for each of the left and the right channels.

e. Don’t forget to add All pragma statements to your C code.

f. Write the function that is similar to the IIR biquad code from chapter 4 with the appropriate changes made for the frames.

**Zip the whole projects for two assignments into a single file as the submission.**

**Only one submission needed per group. Clearly demark your zip file (zip file should contain full names of both the team members).**