**EE 445S Real-Time Digital Signal Processing Laboratory – Prof. Brian L. Evans**

**Lab 3 Instructions – Part 3**

**1. winDSK demonstration**

Connect your DSP board with the PC through the serial cable. Run winDSK8 on the PC, and then follow the instruction on page 58-59 in the real-time DSP book to run the “Notch Filter” applications. Tune the parameter “r” and see how it affects the attenuation around the center frequency.

**2. IIR filter design**

Run fdatool in Matlab to design a bandpass IIR filter:

* Design method: Butterworth
* Order: 6
* Sampling frequency: 48000Hz
* Passband: 5000Hz to 15000Hz

Go to menu “Edit”🡪”Convert to Single Section”, then export the numerator and denominator coefficients into the workspace.

**3. Theoretical magnitude response measurement**

Go back to fdatool. Measure the magnitude response (dB) for the FIR filter that you just designed. You need to measure the magnitude response for 1 kHz, 2 kHz … 24 kHz. Save these values in an Excel file. You will need to compare them with the experimental values later.

**4. DSK implementation using DF-I**

Create a new project as before. After adding the files in “common code” folder, add “IIRmono\_ISR.c” and “StartUp.c” in “code\chapter\_04\ccs\IIRrevA”. Finally add the target configuration files as usual.

Read and understand the code in “IIRmono\_ISR.c” based on the explanation on page 80 in the real-time DSP book. Once you have understood the code, you need to modify it based on following requirements:

(1) Define N as 6

(2) Change the numerator (B) and denominator (A) coefficients based on what you get from MATLAB. The initialization of x and y need to be changed accordingly as N has been changed.

(3) Delete the code between ‘x[0] = CodecDataIn.Channel[LEFT];’ and ‘CodecDataOut.Channel[LEFT] = y[0];’, then insert ‘iir\_df\_one();’.

(4) Create the new function ‘iir\_df\_one()’ in “IIRmono\_ISR.c” file to implement IIR filtering using direct form I. In this new function you need to compute the output value y[0] and also update all the state variables in x and y.

**5. Experimental magnitude response measurement**

Run the program with your own IIR function. Measure the experimental magnitude response and compare it with the theoretical one.

**6.** **Compare number of clock cycles**

Compare the number of clock cycles cost by ‘iir\_df\_one()’ with different optimization levels.

**7.** **DSK implementation using SOS**

Implement the IIR filter using SOS and measure the experimental magnitude response. To get the coefficients please go back to fdatool and choose “Edit”🡪”Convert to Second-Order Sections”.

Here are some hints:

(1) The code for variable declaration should look like:

#define N 2 // IIR filter order

#define M 3 // number of biquads

float B[M][N+1] = {{},{},{}}; // numerator coefficients

float A[M][N+1] = {{},{},{}}; // denominator coefficients

float G[M+1] = {}; // scale factors

float x[M][N+1] = {{},{},{}}; // input value (buffered)

float y[M][N+1] = {{},{},{}}; // output values (buffered)

(2) The in function “Codec\_ISR()” should look like:

x[0][0] = CodecDataIn.Channel[LEFT]; // current input value

for (i=0;i<M;i++){

 biquad(i); // implement the i\_th biquad

}

CodecDataOut.Channel[LEFT] = y[M-1][0]; // setup the LEFT value

WriteCodecData(CodecDataOut.UINT); // send output data to port

(3) Note that in function ‘biquad(i)’ you need to compute the output using x[i][] and y[i][]; update the state variables; and then assign x[i+1][0]=y[i][0].

(4) Run the program; connect the board with the signal generator and the oscilloscope. Tune the frequency in the signal generator from 1 kHz to 24 kHz and record the experimental magnitude response. Compare this with the theoretical magnitude response. (You may draw two lines in the same graph)

**8. Do not forget to answer the lab questions on the webpage.**