Lab 2 – Winter 2017 - EECS 363 Digital Filtering – Due Jan. 20, 2017.

It is first necessary to explain what is meant by "due". In Lab 1 you handed in only printout. The printed output requirement may or may not continue to apply. In most labs from now on I will expect you to give me a demo when you think you are done. You may also want to bring your project to me if you are in trouble somehow, or perhaps you have some reason to want to work in MG18. Most students work at home and find that the demo is most conveniently given by bringing the laptop into MG18 and giving it there. In that case you can use lab instruments instead of the myDAQ software.

An audio loopback program is a program that accepts audio inputs, converts them to digital via the ADC, representing them as ordinary numerical variables, and then sends these variables immediately to the DAC and thence to the audio output of the board. It can be viewed as doing nothing, but it is the basis for DSP that processes the live input signal values and then sends the results of the DSP to the audio output.

We will use TI programs for the eZdsp5505 board, adapted to our eZdsp5535 board.

Create a new project (`smith_lab2`) as you did in Lab 1 but, before exiting the CCS Project window, choose “Advanced settings” and specify as the runtime support library `rts55h.lib`. Delete the existing `main` and then copy/paste the various `.c` and `.h` files from the "loopback" folder that you downloaded earlier as 363_downloads.zip. Replace `C5535.cmd` (delete it) with the downloaded `lnkx.cmd` .

At View> Project Explorer you should see your new `smith_lab2` project listed as the active project, but you will not be able to compile it. Right click on the project name (`smith_lab2`) and then choose Properties at the bottom of the list. Choose Build > C5500 Compiler > Processor Options. Specify memory model huge; also there should be silicon version 5515 specified. Now choose C5500 Compiler > Include Options and add (green + sign) the following paths by browsing the "File system" or by pasting from this document, one at a time:

C:\Code Composer v7\ccsv7\tools\compiler\c5500_4.4.1\c55xx_csl\inc (maybe not 4.4.1)
C:\Code Composer v7\ccsv7\ccs_base\emulation\boards\usbstk5505\include
C:\Code Composer v7\ccsv7\ccs_base\emulation\boards\usbstk5505\misc\include
C:\Code Composer v7\ccsv7\ccs_base\emulation\boards\usbstk5505\misc\src

Note that we are using resources native to the eZdsp5505 board.

Go to C5500 Compiler > Advanced Options > Runtime Model Options and-specify `ptrdiff_size` 32 .

Now go to C5500 Linker > File Search Path and add to the included library files

C:\Code Composer v7\ccsv7\tools\compiler\c5500_4.4.1\c55xx_csl\CSLc55x5h.lib
C:\Code Composer v7\ccsv7\ccs_base\emulation\boards\usbstk5505\lib\usbstk5505bslh.lib

In the same window you should see `rts55h.lib`, which we specified earlier. OK. After a rebuild there should be no errors indicated. There will be one unimportant warning.

We thus using the huge version of the board support library for the eZdsp5505 board. The CSL is the chip support library for the C55xx series of chips. It has its own "`csl_general.h`" but it is not the same as the version we need, which is in the project folder.

Examine the file `main.c` . Note

1. You set your own sample rate by calling the function `set_sampling_frequency_and_gain` and you will find that function in the module `aic3204_init.c` . The given program presets it to 48000
2. The same statement in `main.c` allows you to set the "gain". We are using 0 dB. The original 30 dB assumed a microphone input.
3. The input/output (I/O) is being done by "polling". The single statement `while(!(I2S2_IR & RcvR) );` , having a semicolon at the end, is a complete `while` loop. It says that you just halt on that statement until left and right input values are presented by the A/D converter. These values are being presented (or "received") at the sample rate, which is quite slow compared to the basic cycle frequency of the C5535 DSP chip. Therefore one might say that this program wastes most of its time waiting for input values to be received.
4. You can comment out the useless statement "`mono_input = s…..`"
5. In this program, the received values are immediately sent to the D/A converter ("transmitted"). That is what a loopback program does. We will do digital filtering later by processing (filtering) the values received (inputs) and then transmitting the results (outputs).
6. After a number of I/O cycles equaling the sample rate, an LED is toggled. Therefore an on/off cycle for the LED is 2 seconds (1 sec. on, 1 sec. off).
7. Read/write statements have been allowed to remain in the module `aic3204.c`, though they are not called anymore, since we now do that in `main`. There are a few other remnants of the original program in these files.

You are now going to input sinusoids to the board and observe outputs with the scope (presumably your myDAQ when working at home). It should help to occasionally check the security of the black terminal strip mounted on the side of the myDAQ; its method of mounting is precarious. If you are working at home you can use the Function Generator to input the board. If you are working in MG18 see me on certain details that I can explain to you. The input sinusoids should have a peak value of 200 mV = 0.2 V (400 mv peak-to-peak). If you are working in MG18 you can, use the waveform generator to provide input to the loopback program, using its indicated amplitude 200 mV peak-to-peak. Actually that will give you 400 mV peak-to-peak, i.e. ±200 mV, because the figure given in the window of the instrument, which has a 50Ω output impedance, assumes a 50Ω resistor connected at its output. In many cases you may not get clear traces; I assume the board is sensitive to imperfect ground connections, and MG18 is a noisy room.

In either case you shall input the board from the audio sources using the "STEREO IN" jack on the board. The waveform generator is mono, so when those are the inputs the right and left inputs are the same. In MG18 the cable from the waveform generator to the eZdsp board is via a special board of our construction.

For the output "STEREO OUT" use a Y-connector so that you can connect both the headphones and the scope. In MG18 we have boards to help connect the waveform generator to STEREO IN, and the headphones and scope to STEREO OUT.

Run a frequency response test for frequencies 10 Hz $\leq f \leq$ 16 kHz, using the waveform generator and scope. For the myDAQ scope Trigger choose Edge; adjustment of the “Level” may help sync the myDAQ scope trace. Consider the output at f = 1 kHz to be 0 dB down and record the frequency responses, in dB, for several frequencies in the range using whatever is serving as your scope. Hand in these results. Only the numbers are required; we don't need a curve.

There is not much left to do in this lab except to give me a demo in room MG18.

You could also do the following, but I won't check:

1. **Confirm sample rate**. Use your watch to estimate the LED cycle time, confirming it is what we inferred from the source code.

2. **Listen to real audio**. Input the audio from my Test Audio folder to STEREO IN on the board. You can use things in addition to the Tones, such as the Test Records. Confirm that the left channel is coming to the left headphone and the right to the right. You could use my Test Audio\Test Records\Equalized mp3\HiLoLREq0dB.mp3 file.

1/12/2017