EE442/EE592 Real-Time Digital Signal Processing
Project #3: TI DSP-Based Sound Field Simulator
Due: 5:00pm, Friday, April 1

Assignment

The goal of this project is to implement Moorer's sound field simulator (SFS) on the TI6416DSK. Moorer's SFS differs from Schroeder's, implemented in Project #1, in several ways as described in class and in the textbook.

Sound Field Simulator

Students will code Moorer's SFS described in Section 5.6 of the text. The simulator will consist of a tapped delay line (19 taps) along with the reverberator [parallel bank of six comb filters (each with LPFs in the feedback loop) followed by an allpass filter]. A MATLAB implementation of Moorer's SFS can be found at

`http://www.ece.nmsu.edu/~pdeleon/EE592/TI_6416_Code.html`

EE442

Students registered for EE442 will code Moorer's SFS and process the right channel.

EE592

Students registered for EE592 will code Moorer's SFS and process the right and left channels.

Testing

Although listening to the output of the DSK can give some indication of whether the code is working or not, to be sure we must turn to analytic techniques. One way to rigorously test is to capture the impulse response using the sound card and examine in MATLAB. A more preferred method, is use the DSK's considerable amount of external memory to store the SFS output, download onto the host, and examine in MATLAB (see Ivan Mecimore's procedure at the end of this assignment).

You may compare your captured impulse response with the simulated response. In the DSPFunctionsFixedPoint folder you will find **`impulse.c`** which will replace the incoming input samples on the TI DSK with an internally-generated unit pulse train. Directions are included in the comment section for how to include it in your program. During the official evaluation and grading, we will measure the delays and gains in your actual impulse response and compare with the original specifications.

Submitted Items

You will be required to submit several items for this project, described below.

Code Printout

Please turn in printouts of your code including, **`user_data.h`**, **`util.h`**, **`initialize_program.c`**, **`process_signal.c`**, and any new files you write. Your codes should be fully documented in the header and completely commented.

Code in Electronic Form

Turn in a USB flash drive (with your name clearly labeled on it) with two directories: one which contains code to process audio samples and one which processes the impulse. Each directory should contain **ALL** code and support files necessary to build the program.

Grading

We will evaluate, test, and grade the codes after the due date. A correct impulse response will yield +95 points; an incorrect impulse response will scale the point total accordingly. In addition, good code design, readability, and commenting will be evaluated and scored up to an additional +5 points. If we have questions or problems with the project, we will contact you.

Bonuses

You may wish to earn bonus points with the following enhancements to the project. Note that we will only grade one program be it the basic project or enhanced project. It is far better (point-wise) to have a working basic project than a non-working enhanced project.

EE442

Code Moorer's SFS to also process the left channel. (+10 points). Maximum grade for EE442, Project #3 is 110 points.

Evaluation

EE442

Baseline: Moorer's SFS on right channel +95 points. Code design, readability, and commenting adds +1 to + 5 additional points for a maximum of +100 points. Average readability, commenting, and documentation will be worth +3 points.

EE592

Baseline: Moorer's SFS on right and left channels +95 points. Code design, readability, and commenting adds +1 to + 5 additional points for a maximum of +100 points. Average readability, commenting, and documentation will be worth +3 points.

Deductions:

–3	No initial clearing of buffers
–5	Tap gains in TDL are incorrect
–5	Tap spacings in TDL are incorrect
–10	TDL impulse response missing last tap
–10	Single CF does not have proper response
–10	Single APF does not have proper response
–40	No build

From: Ivan Mecimore <imecimor@nmsu.edu>
Date: April 5, 2007 2:10:02 PM MDT
To: pdeleon@nmsu.edu
Subject: Checking outputs in Matlab

Dr. De Leon: I found that testing by recording the output into MatLab would not give me a good idea of the magnitude of a response. This code will add an array which will hold all of the output samples of the dsp and can be reviewed in MatLab.

Insert this code inside initialize_program.c under global variable initializations:

```
// For storing samples into memory.
     int Monster[MonsterSize+1]; % Bug fix (+1) by P. Davis 2011-03-16
     int *MonsterPtr;
```

```
inside void initialize_program()
// For storing samples into memory.
     int j;
     MonsterPtr = Monster;
     for(j=0;j<=MonsterSize;j++)
          Monster[j] = 0;
```

Insert this code inside user_data.h

```
// For storing samples into memory.
     #define MonsterSize 96000
     extern int Monster[];
     extern int *MonsterPtr;
```

Insert this code inside process_signal.c

```
     // For storing samples into memory.
     if(MonsterPtr <= MonsterSize+Monster)
     {
     *MonsterPtr = (int)FUNCTION();
     MonsterPtr++;
     }
```

Substitute FUNCTION with the function call you wish to test, i.e.
tdl(TDLD, TDL, &TDL_OldestStatePtr, NUMTAPS, TDL_Gains, TDL_Spacing, inputRight)

Now, run the code and let it run for at least 5 seconds. Stop the code, and highlight the word Monster in process_signal. Right click and choose "Open Memory Window".

Go to File -> Data -> Save... and choose a filename.
In "Address:" put the address of Monster.
In "Length:" put in 96000 and save.

Next open the file and remove the top line using notepad. After that import the file using Matlab and type plot(varname/2^15);

The response will be scaled very nicely showing the actual magnitudes.

To comment the code back out, just search all files for "For storing samples into memory."