EC581 DSP Projects: lab6

Dept. of Electrical and Computer Engineering
Rose-Hulman Institute of Technology

Audio Pass Thru

Last Modified on 18-Mar-2004 (may)

Introduction

With this lab we part from the TI notes. The original lab 6 went through 19 pages of details instructions which showed how to set up the McBSP, EDMA and the audio codec. Rather than take you through such agony, I'm going to give you some code that works and have you explain what is going on.

I. Copy lab6 to your work area and play with it some..

You will find it at the usual place: `iw6000\labs\lab6`.

1) Start Code Composer and open the project `Frame_EDMA\Frame_EDMA.pjt`. Compile and run it. Plug a microphone into the IN jack and speakers to the OUT jack on the main DSK board. Play a bit. Which channel (left or right) is being passed through? Why is there a delay? How long is the delay?

2) The DSK has a AD535 which is a low-end codec. It can sample mono at 8K. The DSP also has an Audio Daughter card with a high-end coded. It can sample 16-bit stereo at up to 48K. It also has two built in microphones. The software I'm giving you can work with either card. Open `DSK_Config.h` (it's under Include). You can switch to the better codec by commenting out the line with `TLC320AD535`, and uncommenting the line with `TI_PCM3003_16bit`. Do it, and recompile **everything**. Switch the speakers to the daughter card and make sure JP1 and JP2 are in place. Does it work? Can you hear the stereo? Has the delay changed? Why? How long is it now?

II. Study the code to learn how it works.

Lab Report Part 1.

Now that you know how the code works (we talked about it in class), explain why the delay is the length it is.

`ISRs.c` has several chunks of code commented out. Try uncommented some and see what they do.

III. Add the Sine Wave to the Audio Stream

Add a sine wave to the incoming audio stream

In lab3, we used code to create a sine wave. We will now add this to the incoming audio stream. When played, you should hear the audio along a sine wave tone.

1. Inside the ProcessBuffer () routine in ISRs.c, look for the commented out code labeled "add a sinusoid". Uncomment the code and see (hear?) what happens.
2. We used a better sine generator in lab 2 (Why is it better?). Let's use it. Make a copy of the `for` loop you presently have to generate the sine and comment out the first copy.
3. Up with the includes add:

```
#include "..\sine.h"
```

4. Just below the declarations in ProcessBuffer() add the following.

```
static SINE_Obj sineObj;
static int first=1;

if(first) {
      SINE_init(&sineObj,  440.0, 48 * 1024);
      first = 0;
 }
```

   What does this code do?

5. In your sine for loop add:

```
for(i=0;i < BUFFER_COUNT;i++){
   *pL = *pL + (float)sineGen(&sineObjLeft);
   pL++;
 }
```

   What does this code do?

6. Compile and run. It's too loud. Change your code to fix the problem.
7. Modify your code to produce an 880Hz tone on the right channel and a 440 Hz on the left.

IV Add a switch.

Finally, add code to read a DIP switch. Have this code turn the sine wave on and off. Here is how you add the switch:

1. Add `#include "bsl.h"` to the file using the switch.
2. Go to **Project:Build Options**., Select **Compiler** tab, select Category: **Preprocessor**.
3. Enter `../../bsl/include` as the **Include Search Path (-i)**
4. Add `_DEBUG;BOARD_6711DSK` in **Define Symbols (-d).**
5. Go to **Project:Add Files to Project…**
6. Select `bsl/lib/bsl6711dsk.lib`.

All the above is needed so you can use the following code to test the switch:

```
if(DIP_get(DIP_1) == 0) {
     /* Stuff to do if switch is down */
}
```

You can also test `DIP_2` and `DIP_3`. Can you figure out how to turn on the LEDs?