Technical Notes on using Analog Devices' DSP components and development tools
Phone: (800) ANALOG-D, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com, FTP: ftp.analog.com, WEB: www.analog.com/dsp

Digital Delay Demo for the 2181 EZ-Kit Lite

Contributed by: Greg G. and Steve R.

This DSP EZ-Note describes a simple digital delay demonstration using the ADSP-2181 EZ-Kit Lite development board (ADDS-21XX-EZ-LITE.) This board is a low-cost evaluation/development board based on the ADSP-2181 digital signal processor (DSP.) It is assumed that a "default" installation of the EZ-Kit Lite software (included with the development kit) has been performed. This software includes the assembler, linker, simulator and prom splitter programs as well as a Windows based monitor program and various sample applications.

One of the included demonstration applications is MIC2OUT. This program takes an input from a microphone connected to the EZ-Kit Lite board, digitizes the sample through the A/D converter on the AD1847 codec, and then immediately outputs the sample back through the D/A on the codec.

The source code for the digital delay application is contained in the directory:

C:\ADI-DSP\21xx\EZKITL\2181\DSP

The DSP source file is MIC2OUT.DSP. For this exercise, we will first run the application to see how it operates. Then we will modify the program to add a digital delay. By changing delay parameters, we will change the length of the delay.

The following steps outline the procedure for observing and modifying the digital delay:

1) Create a "DOS box" by double-clicking on the MS-DOS prompt icon from the program manager window.

2) Change to the \DSP directory (an exercise in typing in long paths.) This can be done by typing:

```
CD  \ADI-DSP\21xx\EZKITL\2181\DSP
```

3) Look at the source code file by typing:

```
EDIT MIC2OUT.DSP
```

4) Using the PAGEDOWN key, move down near the end of the file to the INTERRUPT SERVICE ROUTINES section. All of the code in the earlier portion of the application is used to configure the codec and initialize the program.

5) Note just above the INTERRUPT SERVICE ROUTINES section, a line of code:

```
talkthru:     idle;
              jump talkthru;
```

This is the main routine of the MIC2OUT application. The first instruction places the DSP in idle mode and the second loops back indefinitely. The processor performs this loop until it is interrupted by the Receive Interrupt Service Routine, indicating that a data sample from the input is ready to be processed.

6) Next, look at the Receive Interrupt Service Routine. This is the section of the application where the input data is read from the A/D converter with the following instructions:

```
ax0 = dm(rx_buf + 1)
ay0 = dm(rx_buf + 2)
```

This reads the left and right channels from the receive buffer in data memory. The data has been placed there from the codec using the autobuffering capability of the serial ports.

The next two lines take the data and immediately write it to the transmit buffer in data memory to be sent back to the codec for output. This is done by placing the output data in the transmit buffer.

```
dm(tx_buf + 1) = ax0
dm(tx_buf + 2) = ay0
```

7) Exit the editor and return to the DOS command line by typing ALT-FX.

8) Press ALT-TAB to get back to the PROGRAM MANAGER window. Double click on EZ-KIT Lite folder and then start the EZ-Kit Lite monitor by double clicking on the EZ-Kit Lite program icon.

9) Press ALT-L to bring up the Loading menu and choose the Download user program and Go option. Click on the MIC2OUT.EXE program in the File Name listing window to select the MIC2OUT application. Click on OK to load and run the application.

10) Make sure that you have a microphone connected to the input jack of the EZ-Kit Lite board and a set of amplified speakers connected to the output jack. Talk into the microphone and observe the immediate output. This is the talkthru mode working.

11) Next, we will modify the MIC2OUT.DSP source file to add a digital delay.

12) Press ALT-TAB to return to the DOS window. From the *\adi_dsp\21xx\ezkitl\2181\dsp* subdirectory , bring the source file into the editor by typing:

EDIT MIC2OUT.DSP

13) Pagedown to the DATA BUFFER DECLARATIONS section.

14) What we want to do is to add a digital delay to the program. This will mean that rather than take the immediate input sample and just write it to the output, we must store the current sample temporarily and output a sample from a previous input. We do this through the use of circular buffers.

Below is an example of a circular buffer of length N. Data which had been placed into the buffer N samples ago, is read from the current location. New sample data is placed into the buffer at the next location and will be fetched N reads later. When the current location pointer reaches N and increments, it wraps around and starts at location 0 again. Hence the term circular buffer.

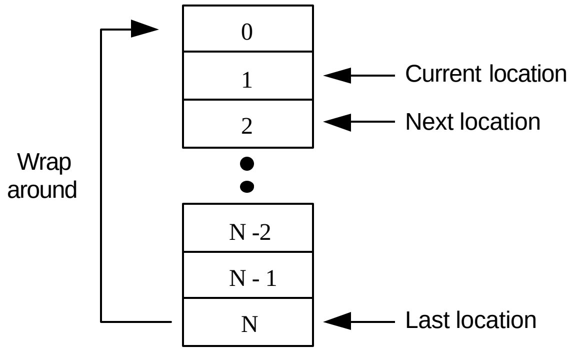

Fig. 1 - Circular Buffer of length N

For our purposes, N will be the total delay time. The codec has a sample rate of 8Ksps so each sample will be 125 µS.

15) On the first line of the section, add the following line to the DATA BUFFER DECLARATIONS section:

.var/dm/ram/circ old_buf[1024];

This declares a circular buffer called old_buf of length 1024 bytes in data memory . This will contain our old , delayed samples.

16) Page down to the ADSP-2181 INITIALIZATION section and add the following lines to the end of the Data Address Generator Initialization section:

i2 = ^old_buf;
L2 = %old_buf;

This will initialize the pointer i2 to point to the start of old_buf and set the length of the buffer to the number of locations we indicated in the Data Buffer Declarations.

17) Page down to the INTERRUPT SERVICE ROUTINES section. Replace the line:

ax0 = dm(rx_buf + 1);

with

ax0 = dm(i2,m2);

This loads the variable ax0 with the delayed sample from old_buf.

18) Following the line:

ay0 = dm(rx_buf + 2);

add the line:

dm(i2,m1) = ay0;

This stores the current sample in the next location in the delay buffer.

19) Save the changes to the file by typing ALT-FS and then exit back to DOS by typing ALT-FX.

20) Compile the new source code by typing:

MIC2OUT

21) Using ALT-TAB, get back to the PROGRAM MANAGER window. Press ALT-L to bring up the Loading menu and select the Download user program and Go option. From the file list, select the MIC2OUT.DSP file and hit Enter. The new code will be running on the EZ-Kit Lite. Talk into the microphone and listen for the delay. The left channel will output the real time signal while the right channel will output the delayed signal.

22) As another exercise, try editing the source code to increase the delay time by lengthening the buffer size. This can be done by repeating steps 15 and 19 - 22.

Hint: At 8Ksps sampling rate, how many samples will there be in 1/2 a second?

This simple example demonstrates some of the capabilities of the 21XX development tools and the EZ-Kit Lite board. By making simple changes to the source code, many different effects and variations are possible.

The source code for the DSP program (MIC2OUT.DSP) could also be modified to change sampling rates, length of the delay and direction of the playback. For more information refer to the 21XX Applications manuals.

Last Modified: 9/9/96