EC581 DSP Projects: Lab Project #5

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

Digital Audio Average Volume Intensity Meter & DSP-to-PC Communication

Last Modified on April 6, 2002 (KEH)
Adapted to 6711DSK 19-Apr-2004, may

1. Adding the VU Meter Feature

A volume-unit (VU) meter is an AC voltmeter that indicates the average magnitude of an audio waveform over a specified time interval, such as 30 ms. (The same time frame was used in the previous lab.) Furthermore, recall that the average magnitude of the frame was already calculated by the AGC program, as the variable `avgMagOfFrame`. The VU meter typically uses a logarithmic (dB) scale. This is because the human ear has a logarithmic response. In our case, we shall use the average frame magnitude as the VU meter reading.

This week's project is to modify your digital AGC program so that it will continue to act as a real-time AGC, but at the same time, it must periodically display the VU meter value. This value must be displayed on the PC screen at specified intervals of time.

You could use the "LOG*_printf( )*" function call from within the DSP program to print the average frame magnitude value. Try it. See page 9-23 of the workshop notes for an example.

Run your program and speak into the microphone or use the signal generator. Verify that the VU meter value appears to be responding properly.

Demonstrate this program to your lab instructor, and obtain his signature on your program listing. Include this signed listing in your lab report as **Attachment A.**

2. DSP-to-PC communication and Visual Basic Application

In the previous section, you verified that the VU meter value (average frame magnitude) is correctly calculated by the modified AGC program running on the C6711 DSK board. Now you are ready to add statements that will send this data down to a companion Visual Basic program that is running on your PC. The Visual Basic program must receive the VU meter value from the DSP program and display it on the PC screen.

Look at the files in *Lab05* in the ece581 directory. In particular look at *ISRs.c* and *rtdxVUmeter.xls*. In the Excel file go to `Tools:Macro:Macros`…. Select `h_read` and click `Edit`. Spend a few minutes studying these files. Compile and run them. Once you have everything working, demonstrate it to your instructor.

Now, study *IRSs.c* and *rtdxVUmeter.xls* again. 15 to 20 minutes might be needed. Have one person study *IRSs.c* and the other r*tdxVUmeter.xls*. Explain your file to your partner. How is data

sent between the host PC and the DSK? How can you modify *your* AGC program to pass the VU information from the DSK to the PC? (Don't just use my program.)

1) Now write a short Visual Basic application that functions as the VU meter display program. First, let it display the average magnitude of a 30 ms segment as an integer value.

2) After you get this to work, change it to display an ASCII-character-oriented bar graph display (row of asterisks.) Include your final listing as ***Attachment B***.

3) Include a printout of your Excel worksheet as ***Attachment C*.**

Take a look in *C:\ti\examples\hostapps\rtdx* for more examples.

Measuring Performance

So far we have used the *CPU Load Graph* to measure how much of the CPU time is being used. We can make more detailed measurements using the DSP/BIOS instrumentation. In `ISRs.c` you should have seen calls to `STS_set()` and `STS_delta()`. These two routines allow you to measure how much time has passed from the time `STS_set` is called until `STS_delta` is called. To make these work in your code:

1. Go to the configuration manager and click on the + by **Instrumentation**.
2. Expand **STS – Statistics Object Manger**.
3. Right-click on **STS** and select **Insert STS**.
4. Rename the newly created **STS0** to **sysProcess**.
5. Save and close the configuration.
6. Go to the DSP/BIOS menu and select **View Statistics**.

In your DSK code, insert the statement:

```
STS_set(&stsProcess, CLK_gethtime());
```

to start timing and

```
STS_delta(&stsProcess, CLK_gethtime());
```

to stop. When you run your code the statistics should appear in the statistics window. If not, right-click on the statistics window and select **Property Page…** Check **sysProcess**.
You can have multiple statistic objects, each monitoring different parts of your code.

Conclusions

The skills learned in this lab will probably be very relevant to your term DSP project. Most of the term projects will require writing two programs, one for the DSP side and one for the PC side, which communicate data between them.

The following is taken from the end of *C:\ti\examples\hostapps\rtd\readme.htm*.

How to Write A C++ RTDX Client

If you wish to create your own RTDX COM client using Microsoft Visual C++, follow these steps:

1. Import the RTDX Server type library by entering the following lines in your file:

```
#import
using namespace RTDXINTLib;
// or
// #import no_namespace
```

2. Declare a variable of type IRtdxExpPtr by entering the following line of code in your file:

```
IRtdxExpPtr rtdx;
```

3. Initialize COM by entering the following line of code in your file:

```
::CoInitialize(NULL);
```

4. Instantiate the RTDX COM Object by entering the following line of code in your file:

```
HRESULT hr = rtdx.CreateInstance( L"RTDX" );
```

5. Set the desired board and processor

```
BSTR board=::SysAllocString(L"C6xxx EVM (Texas Instruments)");
BSTR processor=::SysAllocString(L"TMS320C6201EVM");
status = rtdx->SetProcessor(board,processor);
```

You are now free to call the methods of the RTDX Host COM API.
Example:

```
rtdx->Open("ochan", "R");
rtdx->ReadI4(&data);
```

6. When you are finished calling the methods of the RTDX Host COM API, release the reference to the COM object by entering the following line of code in your file:

```
rtdx.Release();
```

7. Uninitialize COM by entering the following line of code in your file:

```
::CoUninitialize();
```

The example above communicates with the RTDX Host COM api using early-binding. The source code for The Console Event Display (eventprog.exe) is an example that demostrates this algorithm.

How To Write A Visual Basic RTDX Client

If you wish to create your own RTDX COM client using Microsoft Visual Basic, follow these steps:

1. Declare a variable of type Object by entering the following line of code in your file:

```
Dim rtdx As Object
```

2. Create an instance of the RTDX COM object by entering the following line of code in your file:

```
Set rtdx = CreateObject("RTDX")
```

3. Set the desired board and processor

```
Const board As String = "C6xxx EVM (Texas Instruments)"
Const processor As String = "TMS320C6201EVM"
status = rtdx.SetProcessor(board,processor)
```

You are now free to call the methods of the RTDX Host COM API.
Example:

```
rtdx.Open("ochan", "R");
rtdx.ReadI4(data);
```

4. When you are finished calling the methods of the RTDX Host COM API, release the reference to the COM object by entering the the following line of code in your file:

```
Set rtdx = Nothing
```

The example above communicates with the RTDX Host COM api using late-binding. The source code for The General-Purpose Display (rtdx.exe) is an example that demostrates this algorithm.