

**Developing a Digital Filter to Process WAV Files**  
**Dr. Fred DePiero – EE Department – CalPoly State University**

The development tools for WAV file processing in this package are intended to mimic the style of signal processing used on a real-time DSP board. In real-time systems, the difference equation must be evaluated sample-by-sample, as new inputs  $x(n)$  are acquired. As such a 'history'  $x(n-k)$  of recent inputs, and of recent outputs,  $y(n-k)$ , must be maintained in order to compute each  $y(n)$  output value. FYI, digital signal *processors* feature circular addressing modes, which efficiently maintain the signal history. With the WAV file programs, the history can be updated simply via scalar variables with appropriate assignments. (Or you can use arrays, which are better).

The following diagram provides an analogy relating WAV file processing to the A/D and D/A hardware of a real-time system.



Use microphone and a sound recorder program to save WAV file to disk.	Drag & drop input WAV file onto C program. Output saved in another WAV file.	Playback output WAV file through sound card.
-----------------------------------------------------------------------	------------------------------------------------------------------------------	----------------------------------------------

*Comparison of block diagram for a real-time DSP system versus WAV file processing.*

The template C program in the development package includes the logic necessary to respond to the drag & drop operation and to open & create WAV files. See the 'WAV Project'. Note the programming required just involves the signal processing aspects (implementing the difference equation & history update) not any WAV file input/output.

An excerpt from the main program follows. This shows the WAV input (analogous to A/D) and the WAV output (D/A). The assignment 'out = in;' is the difference equation, in this case simply  $y(n) = x(n)$ .

```
// process all samples in the input wav file
for(n=0;n<num_samples;n++)
{
    // read an input sample from wav file
    in = read_wav(wav_in);

    // do processing
    // (dsp students: get to work here!)
    out = in;

    // save result for output wav file
    write_wav(wav_out,out);
}
```

*Excerpt of C program implementing the difference equation  $y(n)=x(n)$ . No update of the history of recent inputs or outputs is provided in this example. See 'WAV Project'.*

## Development Cycle for Digital Filter

### Installing the Software Development Environment

Go to DePiero's web site; follow the link to the software development tools. Download & install the Dev-C++ package as described on the web page. Open the 'WAV Project' under the 'C:\Dev-C++\CP\CP-Projects' folder, and double click on the 'Wav Project.dev' file. This starts the development environment. More information on these steps is provided on the web site. You may wish to setup a shortcut to the 'CP-Projects' folder. See additional notes on the web site regarding installation and use.

### Design & Analyze the Digital Filter

- *On Paper*, Complete the design calculations for the  $A_k$ ,  $B_k$  filter coefficients.
- *On Paper*, Find the magnitude and phase of the frequency response.
- *In SciLab*, Use the following commands to find the magnitude and phase of the frequency response. Replace the 'b = [B0 B1 B2...]' and 'a = [A0 A1 A2...]' with your  $A_k$  and  $B_k$  values. Additional notes available on web, see 'Filter Analysis'.

```
>> b = [0.3333 0.3333 0.3333];  
>> a = [1];  
>> freqz_fwd(b, a, 512, 16000);
```

In this example, the sample rate  $S=16000\text{Hz}$  and 512 points are used for plotting the frequency response.

### Implement the Digital Filter by Editing and Compiling Program

- *In the Dev-C++ tool*, Edit the main\_wav.cpp file. Add variables and make any modifications needed to implement your filter or processing algorithm.
- *In the Dev-C++ tool*, Use the 'Execute' menu to 'Compile' your program (or use the toolbar). This creates your 'WAV Project.exe' executable. Run the executable from the desktop (see below).

### Run the Digital Filter to Process WAV File

- *On the Desktop*, Drag & drop the 'noise.wav' file onto your new executable. The output file 'digital\_filter\_output.wav' should be created.

You may also process other WAV files (mono), and are encouraged to listen to the results of your digital filters.

### Test Digital Filter by Verifying Frequency Response

- *On the Desktop*, Drag & drop the 'digital\_filter\_output.wav' output of your digital filter, onto the shortcut for the 'spectrum analyzer.exe' program. This will result in a .bmp image file being created: magnitude\_freq\_response.bmp.
- *On the Desktop*, Double-click on the .bmp file to examine the frequency response.
- Note the spectrum plotted for the 'digital\_filter\_output.wav' is not normalized with respect to the amplitude of the input 'noise.wav'. Hence a filter with unity gain in the pass band will not appear as such in the .bmp image. However, then overall shape should match reasonably well (spectral spacing limitations...).