Jared Nixon

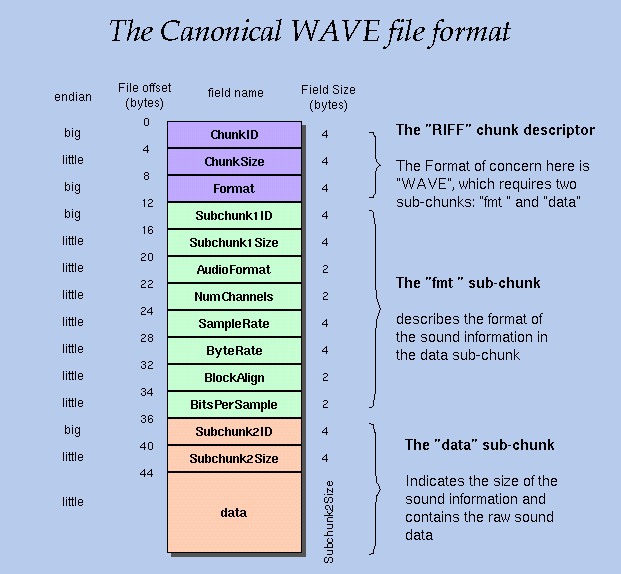
Project Phase One Report

3/13/2016

1. **Problem Description:** Phase one of the project consisted of reading the data from a WAV file and adding a sine wave at 2500 Hz at one quarter of the amplitude to the file. The sample data values must be written sample by sample. This is done to conserve memory, just in case the program needed to be put to a microcontroller that has very limited memory. The output file of phase one is produced to be used as the input file of phase two, where a low pass filter will be implemented to take the sine wave back out fo the file. This data processing must be done in C/C++.

**Proposed Solution:** In order to achieve the task given, the header of the fie must be written correctly before moving on. This header will give viable pieces of information essential to the processing of the file (see section below for format). Once this hs been read and written correctly, the data samples can be read in one by one. Each time a sample is read in, the sine wave must be added to each channel (because the WAV file used is stereo), and written to the output file. Overflow must be taken into account so that no nasty noisees will pop up due to the sample value being higher than the maximum possible data value. Once the value has been written, the next sample will be real in and the loop will be repeated until all data samples have been processed (aka the end of the file). This will properly achieve the sample by sample processing that was required in the project.

1. **Format of the wave file:**



This format can give us all the important information needed to complete the project. The main points of emphasis are ChunkSize (size of entire file – 8), SubChunkOneSize, SampleRate (Frequency in Hz), NumChannels (Tells us stereo or mono), BitsPerSample (Divide by 8 for BytesPerSample), and SubChunkTwoSize (Tells us size of the sampled data to be read). If the header is written and read correctly, you can output any of these variables and be sure they are correct before moving on to the reading and processing of the data samples. You can also use an offset to get to the data samples without worrying about the header (Because we know that the header is 44 bytes), or for my case, I added a counter so each time the data sample loop is ran through, the offset would adjust by four bytes just to be sure the program is always getting the next sample (more of a measure to ensure nothing gets skipped for some reason).

1. **Summary of the text file:**

CPE381 project - Processing WAV File - Jared Nixon

The sampling frequency of this WAV file is 44100 Hz.

The length of this file in seconds (calculated from the number of samples) is 34 seconds.

This program completed the processing of the WAV file in 1.631 seconds. (i7 – 4790K, 8 GB RAM)

I believe this program can run in real time, because it recieves data, processes it, and outputs the processed data very quickly. This could be thrown back in as input in order to create a real time system that responds according to that recieved data.