# Making Waves

Until recent months almost all my work on sound files has been with CD Audio Disc files in the 'data' format which can be saved or burnt using C(DVD)Burn. However more recently I have also begun working with other types of data – e.g. with a 48 ksample/sec data rate – and with Linux systems. This meant that I needed to be able to work with files in the 'WAV' (wave) format...

In fact WAV is a sort of 'container' for data which in some ways is similar to the 'JPEG'. This is because it actually consists of a 'header' followed by a 'payload'. The header is a set of metadata that specifies the details of how the payload is organised. So to be able to read and write WAV files you first need to be able to make sense of the information in the header. This also enables you to generate headers which correctly define the sound data that follows them when creating a new WAV file. As with the JPEG, there can be a variety of types of header, and the payload can take various forms. But in practice, so far as I have been able to tell, the most common form of WAV header when working with LPCM (Linear Pulse Code Modulation) data consists of the same arrangement of 44 bytes whose values indicate the sampling rate, etc.

All being well, you should find that a number of new applications are now available from Archive to accompany this article. The first of these is called !ReadWAVHeader. This will read the header from a WAV file if you run the application and type in the WAV file's name. (Make sure first that the 'Settings' file inside !ReadWAVHeader gives the directory containing the WAV file!) With most files this will tell you details like the sampling rate, if the audio is mono or stereo, and the duration of the payload in bytes and in seconds.

There are also five other new applications. As with the ones that came with earlier articles I recommend reading the !Help file inside each of their application directories before you try using them as you may need to alter the details in a 'Settings' file, etc, to ensure they work correctly on your computer.

**!TrackVOB2WAV**. This can be used to extract the soundtrack from a VOB file when the sound has been recorded in LPCM format. This lets you rip the sound from home made DVD Videorecordings where the sound is LPCM. Note that for this application to work you will need to have installed !FFMPEG and for it to have been seen by the filer. Note also that the extraction process can be slow as it depends on how quickly FFMPEG can access your DVDROM drive.

**!WAV_Maker**. This works in the same way as !TrackMaker and allows you to snip up a WAV file into tracks by specifying a series of times in an EditFile. It should work with mono or stereo WAV files with sampling rates up to and including 48ksample/sec. So it is more flexible in terms of the formats it can deal with than !TrackMaker.

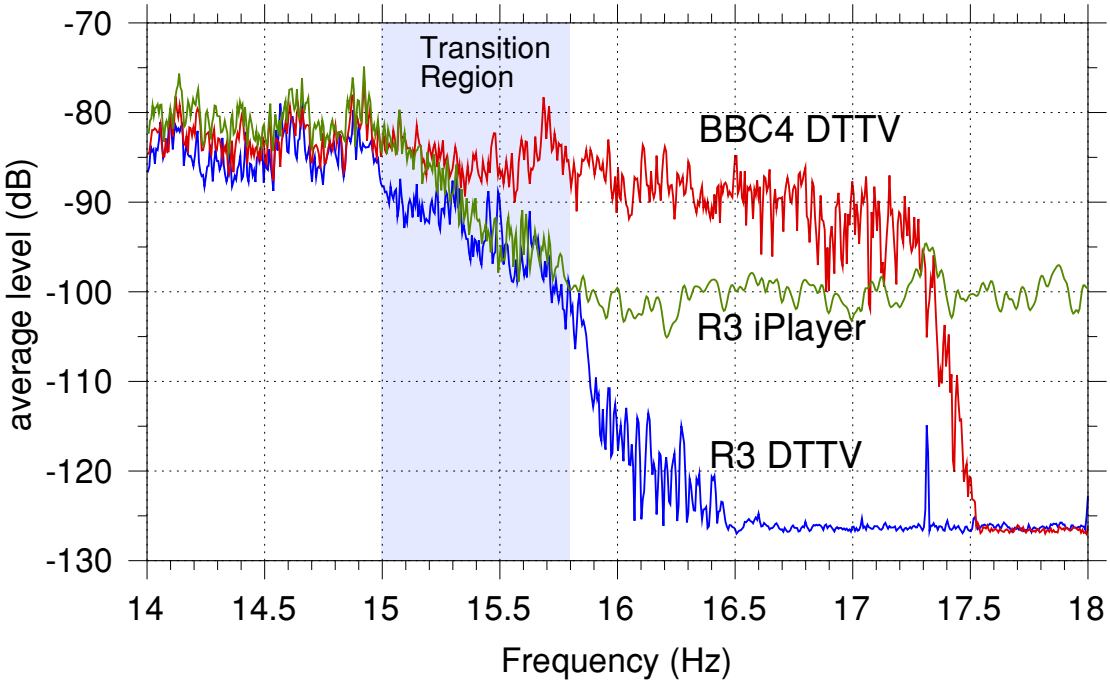
**!WAV_Glue**. This allows you to 'glue together' a series of WAV files. In effect it can reverse the trackmaking process and produce one long continuous file from a series of input files. Note that it can only work correctly if all the WAV files to be glued together share the same sampling rate, etc.

**!WAV_Stats**. This works in much the same way as !TrackStats and allows you to examine how the peak sound levels in a file vary from one 0.1 second period to the next thoughout the

duration of the recorded data payload. It also produces a histogram of how often each peak level occurs. The results are saved to files on your ramdisc.

**!WAV_FFTScan**. This provides output similar to !TrackFFTScan and allows you to obtain time-averaged spectra of the signals from a section of a WAV file. You can specify the start time and duration of the section to be analysed. The results are written to files on your ramdisc.

The output files generated by !WAV_Stats and !WAV_FFTScan are in a format that you can load into !Tau or spreadsheets if you wish to plot graphs of the results or carry out further analysis.




Expanded view of cut-off region of spectra

The above shows an example of results obtained using !WAV_FFTScan which I then plotted using !Tau. The data comes from the same 60 second section of a BBC Proms broadcast of the 'Fountains of Rome' which I had recorded from three sources. So in each case the results are the time-averaged power-frequency spectrum over hundreds of individual spectra.

One source was Radio 3 via BBC iPlayer, the next was the Freeview (DTTV) version of R3, and the third was the BBC4 TV broadcast on Freeview. This was part of a detailed examination I carried out, comparing the BBC iPlayer with DTTV. The Freeview recordings are 48ksample/sec 16 bit LPCM WAV files whereas the iPlayer recording was in the CDBurn 'data' format at 44.1ksamples/sec. (Yes, the BBC iPlayer audio streams are at the CD Audio rate of 44.1ksamples/sec, although Freeview and most internal BBC distribution uses 48k.) One of the differences I found between the iPlayer and DTTV was in the details of the spectra in the region above about 15kHz. !WAV_FFTScan was very useful for this purpose. Ideally the three spectra shown should be much the same. But in practice you can see that they are quite different. Although below about 15kHz they are very similar, so the plot I have shown just zooms in to examine their differences at high frequency clearly. More details are on the www.audiomisc.co.uk website if you are interested.

The current versions of the WAV applications do have some limitations. They can only cope with

WAV files that comply with the following requirements:

- The sampling rate is not more than 48,000 samples/sec (per channel)
- The data is LPCM
- The sound is mono or stereo (i.e. not more than 2 channels)
- Each sample only uses 1 or 2 bytes (i.e. not more than 16 bits per sample)
- The WAV header has the common 44-byte format

In recent months I've been using an assortment of Linux and RO computers to work with audio WAV files, and all the files I had to handle conform to the above limitations. So the chances are that most of the files you encounter will be OK. However some files may contain 24 bit samples (3 bytes per sample), or be at a sampling rate above 48ks/sec. Some might contain non-LPCM data payloads or use less common header formats. So if a particular problem like that crops up often, please let me know and I will see if I can alter the applications to cope. However I suspect that there will be a wide variety of rarely-used formats, so making changes probably only makes sense for relatively common problems.

One other difficulty I recently encountered was with a WAV file that did meet all the above requirements, but which I still couldn't handle using the above applications. Alas, this was because the file was 2.6GB long! So the problem was that the RO filing systems could not cope with such a large file. I therefore had to start by writing an application to use with ROX/Linux to cut up the huge WAV file into chunks RO could digest!

Jim Lesurf
1250 Words
23rd Oct 2009