Audio File Format Specifications

File Description: WAVE or RIFF WAVE sound file
File Extension: Commonly `.wav`, sometimes `.wave`
File Byte Order: Little-endian

Prof. Peter Kabal, MMSP Lab, ECE, McGill University: Last update: 2017-05-02

WAVE Specifications

The WAVE file specifications came from Microsoft. The WAVE file format use RIFF chunks, each chunk consisting of a chunk identifier, chunk length and chunk data.

- WAVE specifications, Version 1.0, 1991-08: riffmci.rtf
  Local copy: Multimedia Programming Interface and Data Specifications 1.0 (see pages 56-65)
- WAVE update (Revision: 3.0), 1994-04-15: Multimedia Registration Kit Revision 3.0 (Q120253))
  Local copy: New Multimedia Data Types and Data Techniques (see pages 12-22)
- Multi-channel / high bit resolution formats, 2001-12-04: Multiple Channel Audio Data and WAVE Files
  Local copy: Multiple Channel Audio Data and WAVE Files

The European Broadcast Union (EBU) has standardized on an extension to the WAVE format that they call Broadcast WAVE format (BWF). It is aimed at carrying PCM or MPEG audio data. In its simplest form, it adds a <bext> chunk with additional metadata. Full documentation is available on line from the EBU.

Data Types

The data in WAVE files can be of many different types. Data format codes are listed in the following:

- Internet RFC, Codec registrations, 1998-06: ftp://ftp.isi.edu/in-notes/rfc2361.txt
  Local copy: rfc2361.txt
- Microsoft include files (part of the MSVC compiler or the *DirectX SDK*: from Microsoft Download Center). For new installations of Visual Studio, the mmreg.h include file is installed into C:\Program Files (x86)\Windows Kits\10\Include\10.0.15063.0\shared. This document shows a huge number of (proprietary) compressed formats, most of which are now obsolete.
  Local copy: mmreg.h (extract of Version 1.58)

Wave File Format

Wave files have a master RIFF chunk which includes a WAVE identifier followed by sub-chunks. The data is stored in little-endian byte order.

Field	Length	Contents
`ckID`	4	Chunk ID: "`RIFF`"
`cksize`	4	Chunk size: `4+n`
`WAVEID`	4	WAVE ID: "`WAVE`"
WAVE chunks	*n*	Wave chunks containing format information and sampled data

fmt Chunk

The `fmt` specifies the format of the data. There are 3 variants of the Format chunk for sampled data. These differ in the extensions to the basic `fmt` chunk.

Field	Length	Contents
`ckID`	4	Chunk ID: "`fmt `"
`cksize`	4	Chunk size: 16, 18 or 40
`wFormatTag`	2	Format code
`nChannels`	2	Number of interleaved channels
`nSamplesPerSec`	4	Sampling rate (blocks per second)
`nAvgBytesPerSec`	4	Data rate
`nBlockAlign`	2	Data block size (bytes)
`wBitsPerSample`	2	Bits per sample

`cbSize`	2	Size of the extension (0 or 22)
`wValidBitsPerSample`	2	Number of valid bits
`dwChannelMask`	4	Speaker position mask
`SubFormat`	16	GUID, including the data format code

The standard format codes for waveform data are given below. The references above give more format codes for compressed data, a good fraction of which are now obsolete.

Format Code	PreProcessor Symbol	Data
`0x0001`	`WAVE_FORMAT_PCM`	PCM
`0x0003`	`WAVE_FORMAT_IEEE_FLOAT`	IEEE float
`0x0006`	`WAVE_FORMAT_ALAW`	8-bit ITU-T G.711 A-law
`0x0007`	`WAVE_FORMAT_MULAW`	8-bit ITU-T G.711 μ-law
`0xFFFE`	`WAVE_FORMAT_EXTENSIBLE`	Determined by `SubFormat`

PCM Format

The first part of the Format chunk is used to describe PCM data.

- For PCM data, the Format chunk in the header declares the number of bits/sample in each sample (`wBitsPerSample`). The original documentation (Revision 1) specified that the number of bits per sample is to be rounded up to the next multiple of 8 bits. This rounded-up value is the container size. This information is redundant in that the container size (in bytes) for each sample can also be determined from the block size divided by the number of channels (`nBlockAlign` / `nChannels`).
  - This redundancy has been appropriated to define new formats. For instance, *Cool Edit* uses a format which declares a sample size of 24 bits together with a container size of 4 bytes (32 bits) determined from the block size and number of channels. With this combination, the data is actually stored as 32-bit IEEE floats. The normalization (full scale 2^{23}) is however different from the standard float format.
- PCM data is two's-complement except for resolutions of 1-8 bits, which are represented as offset binary.

Non-PCM Formats

An extended Format chunk is used for non-PCM data. The `cbSize` field gives the size of the extension.

- For all formats other than PCM, the Format chunk *must* have an extended portion. The extension can be of zero length, but the size field (with value 0) must be present.
- For float data, full scale is 1. The bits/sample would normally be 32 or 64.
- For the log-PCM formats (μ-law and A-law), the Rev. 3 documentation indicates that the bits/sample field (`wBitsPerSample`) should be set to 8 bits.
- The non-PCM formats must have a `fact` chunk.

Extensible Format

The `WAVE_FORMAT_EXTENSIBLE` format code indicates that there is an extension to the Format chunk. The extension has one field which declares the number of "valid" bits/sample (`wValidBitsPerSample`). Another field (`dwChannelMask`) contains bits which indicate the mapping from channels to loudspeaker positions. The last field (`SubFormat`) is a 16-byte globally unique identifier (GUID).

- With the `WAVE_FORMAT_EXTENSIBLE` format, the original bits/sample field (`wBitsPerSample`) must match the container size (`8 * nBlockAlign / nChannels`). This means that `wBitsPerSample` must be a multiple of 8. Reduced precision within the container size is now specified by `wValidBitsPerSample`.
- The number of valid bits (`wValidBitsPerSample`) is informational only. The data is correctly represented in the precision of the container size. The number of valid bits can be any value from 1 to the container size in bits.
- The loudspeaker position mask uses 18 bits, each bit corresponding to a speaker position (e.g. Front Left or Top Back Right), to indicate the channel to speaker mapping. More details are in the document cited above. This field is informational. An all-zero field indicates that channels are mapped to outputs in order: first channel to first output, second channel to second output, etc.
- The first two bytes of the GUID form the sub-code specifying the data format code, e.g. `WAVE_FORMAT_PCM`. The remaining 14 bytes contain a fixed string, "`\x00\x00\x00\x00\x10\x00\x80\x00\x00\xAA\x00\x38\x9B\x71`".

The `WAVE_FORMAT_EXTENSIBLE` format should be used whenever:

- PCM data has more than 16 bits/sample.
- The number of channels is more than 2.
- The actual number of bits/sample is not equal to the container size.
- The mapping from channels to speakers needs to be specified.

fact Chunk

All (compressed) non-PCM formats *must* have a `fact` chunk (Rev. 3 documentation). The chunk contains at least one value, the number of samples in the file.

Field	Length	Contents
`ckID`	4	Chunk ID: "`fact`"
`cksize`	4	Chunk size: minimum 4
`dwSampleLength`	4	Number of samples (per channel)

- The Rev. 3 documentation states that the Fact chunk "is required for all new new WAVE formats", but "is not required" for the standard `WAVE_FORMAT_PCM` file. One presumes that files with IEEE float data (introduced after the Rev. 3 documention) need a `fact` chunk.
- The number of samples field is redundant for sampled data, since the Data chunk indicates the length of the data. The number of samples can be determined from the length of the data and the container size as determined from the Format chunk.
- There is an ambiguity as to the meaning of "number of samples" for multichannel data. The implication in the Rev. 3 documentation is that it should be interpreted to be "number of samples per channel". The statement in the Rev. 3 documentation is:

  > The `nSamplesPerSec` field from the wave format header is used in conjunction with the `dwSampleLength` field to determine the length of the data in seconds.

  With no mention of the number of channels in this computation, this implies that `dwSampleLength` is the number of samples per channel.
- There is a question as to whether the `fact` chunk should be used for (including those with PCM) `WAVE_FORMAT_EXTENSIBLE` files. One example of a `WAVE_FORMAT_EXTENSIBLE` with PCM data from Microsoft, does not have a `fact` chunk.

data Chunk

The `data` chunk contains the sampled data.

Field	Length	Contents
`ckID`	4	Chunk ID: "`data`"
`cksize`	4	Chunk size: n
sampled data	n	Samples
pad byte	0 or 1	Padding byte if n is odd

Examples

Consider sampled data with the following parameters,

- N_c channels
- The total number of blocks is N_s. Each block consists of N_c samples.
- Sampling rate F (blocks per second)
- Each sample is M bytes long

PCM Data

Field	Length	Contents
`ckID`	4	Chunk ID: "`RIFF`"
`cksize`	4	Chunk size: `4 + 24 + (8 +` $M*N_c*N_s$ `+ (0 or 1)`
`WAVEID`	4	WAVE ID: "`WAVE`"
`ckID`	4	Chunk ID: "`fmt `"
`cksize`	4	Chunk size: 16
`wFormatTag`	2	`WAVE_FORMAT_PCM`
`nChannels`	2	N_c
`nSamplesPerSec`	4	F
`nAvgBytesPerSec`	4	$F*M*N_c$
`nBlockAlign`	2	$M*N_c$
`wBitsPerSample`	2	rounds up to `8*`M

ckID	4	Chunk ID: "data"
cksize	4	Chunk size: $M*N_C*N_S$
sampled data	$M*N_C*N_S$	N_C*N_S channel-interleaved M-byte samples
pad byte	0 or 1	Padding byte if $M*N_C*N_S$ is odd

Notes

- WAVE files often have information chunks that precede or follow the sound data (`data` chunk). Some programs (naively) assume that for PCM data, the preamble in the file header is exactly 44 bytes long (as in the table above) and that the rest of the file contains sound data. This is not a safe assumption.

Non-PCM Data

Field	Length	Contents
ckID	4	Chunk ID: "RIFF"
cksize	4	Chunk size: 4 + 26 + 12 + (8 + $M*N_C*N_S$ + (0 or 1))
WAVEID	4	WAVE ID: "WAVE"
ckID	4	Chunk ID: "fmt ";
cksize	4	Chunk size: 18
wFormatTag	2	Format code
nChannels	2	N_C
nSamplesPerSec	4	F
nAvgBytesPerSec	4	$F*M*N_C$
nBlockAlign	2	$M*N_C$
wBitsPerSample	2	$8*M$ (float data) or 16 (log-PCM data)
cbSize	2	Size of the extension: 0
ckID	4	Chunk ID: "fact"
cksize	4	Chunk size: 4
dwSampleLength	4	N_C*N_S
ckID	4	Chunk ID: "data"
cksize	4	Chunk size: $M*N_C*N_S$
sampled data	$M*N_C*N_S$	N_C*N_S channel-interleaved M-byte samples
pad byte	0 or 1	Padding byte if $M*N_C*N_S$ is odd

- Microsoft *Windows Media Player* will not play non-PCM data (e.g. μ-law data) if the `fmt` chunk does not have the extension size field (`cbSize`) or a `fact` chunk is not present.

Extensible Format

Field	Length	Contents
ckID	4	Chunk ID: "RIFF"
cksize	4	Chunk size: 4 + 48 + 12 + (8 + $M*N_C*N_S$ + (0 or 1)
WAVEID	4	WAVE ID: "WAVE"
ckID	4	Chunk ID: "fmt "
cksize	4	Chunk size: 40
wFormatTag	2	WAVE_FORMAT_EXTENSIBLE
nChannels	2	N_C

nSamplesPerSec	4	F
nAvgBytesPerSec	4	$F * M*N_C$
nBlockAlign	2	$M*N_C$
wBitsPerSample	2	$8 * M$
cbSize	2	Size of the extension: 22
wValidBitsPerSample	2	at most $8*M$
dwChannelMask	4	Speaker position mask
SubFormat	16	GUID (first two bytes are the data format code)
ckID	4	Chunk ID: "fact"
cksize	4	Chunk size: 4
dwSampleLength	4	N_C*N_S
ckID	4	Chunk ID: "data"
cksize	4	Chunk size: $M*N_C*N_S$
sampled data	$M*N_C*N_S$	N_C*N_S channel-interleaved M-byte samples
pad byte	0 or 1	Padding byte if $M*N_C*N_S$ is odd

- The fact chunk can normally be omitted if the sampled data is in PCM format.
- In some cases, Microsoft *Windows Media Player* enforces the use of the WAVE_FORMAT_EXTENSIBLE format code. For instance a file with 24-bit data declared as a standard FORMAT_PCM format code will not play, but a file with 24-bit data declared as a WAVE_FORMAT_EXTENSIBLE file with a WAVE_FORMAT_PCM subcode can be played.

Sample Wave Files