Wav file format

Original article from: http://www.sonicspot.com/guide/wavefiles.html
Reference: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/

Table of Contents

Overview

The Wave file format is Windows' native file format for storing digital audio data. It has become one of the most widely supported digital audio file formats on the PC due to the popularity of Windows and the huge number of programs written for the platform. Almost every modern program that can open and/or save digital audio supports this file format, making it both extremely useful and a virtual requirement for software developers to understand. The following specification gives a detailed description of the structure and inner workings of this format.

Data Formats

Since the Wave file format is native to Windows and therefor Intel processors, all data values are stored in Little-Endian (least significant byte first) order.

Strings
Wave files may contain strings of text for specifying cue point labels, notes, etc. Strings are stored in a format where the first byte specifies the number of following ASCII text bytes in the string. The following bytes are of course the ASCII character bytes that

make up the text string. Pascal programmers will notice that this is the same format used for Pascal strings.

7	'e'	'x'	'a'	'm'	'p'	'l'	'e'

Wave String Format Example

File Structure

Wave files use the standard RIFF structure which groups the files contents (sample format, digital audio samples, etc.) into separate chunks, each containing it's own header and data bytes. The chunk header specifies the type and size of the chunk data bytes. This organization method allows programs that do not use or recognize particular types of chunks to easily skip over them and continue processing following known chunks. Certain types of chunks may contain sub-chunks. For example, in the diagram to the right, you can see that the "fmt " and "data" chunks are actually sub-chunks of the "RIFF" chunk.

Basic Wave File Layout

Chunk ID "RIFF"
Chunk Data Size

RIFF Type ID "WAVE"

Chunk ID "fmt "
Chunk Data Size

Sample Format Info

Chunk ID "data"
Chunk Data Size

Digital Audio Samples

Chunk Header

Chunk Data Bytes

One tricky thing about RIFF file chunks is that they must be word aligned. This means that their total size must be a multiple of 2 bytes (ie. 2, 4, 6, 8, and so on). If a chunk contains an odd number of data bytes, causing it not to be word aligned, an extra padding byte with a value of zero must follow the last data byte. This extra padding byte is not counted in the chunk size, therefor a program must always word align a chunk headers size value in order to calculate the offset of the following chunk.

Wave File Header - RIFF Type Chunk

Wave file headers follow the standard RIFF file format structure. The first 8 bytes in the file is a standard RIFF chunk header which has a chunk ID of "RIFF" and a chunk size equal to the file size minus the 8 bytes used by the header. The first 4 data bytes in the "RIFF" chunk determines the type of resource found in the RIFF chunk. Wave files always use "WAVE". After the RIFF type comes all of the Wave file chunks that define the audio waveform.

Offset	Size	Description	Value
0x00	4	Chunk ID	"RIFF" (0x52494646)
0x04	4	Chunk Data Size	(file size) - 8
0x08	4	RIFF Type	"WAVE" (0x57415645)
0x10		Wave chunks	

RIFF Type Chunk Values

Wave File Chunks

There are quite a few types of chunks defined for Wave files. Many Wave files contain only two of them, specifically the Format Chunk and the Data Chunk. These are the two chunks needed to describe the format of the digital audio samples and the samples themselves. Although it is not required by the official Wave file specification, it is good practice to place the Format Chunk before the Data Chunk. Many programs expect the chunks to be stored in this order and it is more sensible when streaming digital audio from a slow, linear source such as the Internet. If the format were to come after the data, all of the data and then the format would have to be streamed before playback could start correctly.

All RIFF Chunks and therefore Wave Chunks are stored in the following format. Notice that even the above mentioned RIFF Type Chunk conforms to this format.

Offset	Size	Description
0x00	4	Chunk ID
0x04	4	Chunk Data Size
0x08	Chunk Data Bytes	

RIFF Chunk Format

The rest of this document goes through the different types of Wave chunks, describing the format of their data bytes and what they mean. You can use the table of contents at the beginning of this document to help find the chunk type you are interested in.

Format Chunk - "fmt "

The format chunk contains information about how the waveform data is stored and should be played back including the type of compression used, number of channels, sample rate, bits per sample and other attributes.

Offset	Size	Description	Value
0x00	4	Chunk ID	"fmt " (0x666D7420)
0x04	4	Chunk Data Size	16 + extra format bytes
0x08	2	Compression code	1 - 65,535
0x0a	2	Number of channels	1 - 65,535
0x0c	4	Sample rate	1 - 0xFFFFFFFF
0x10	4	Average bytes per second	1 - 0xFFFFFFFF
0x14	2	Block align	1 - 65,535
0x16	2	Significant bits per sample	2 - 65,535
0x18	2	Extra format bytes	0 - 65,535
0x1a	Extra format bytes *		

Wave Format Chunk Values, * read following text for details

Chunk ID and Data Size
The chunk ID is always "fmt " (0x666D7420) and the size is the size of the standard wave format data (16 bytes) plus the size of any extra format bytes needed for the

specific Wave format, if it does not contain uncompressed PCM data. Note the chunk ID string ends with the space character (0x20).

Compression Code
The first word of format data specifies the type of compression used on the Wave data included in the Wave chunk found in this "RIFF" chunk. The following is a list of the common compression codes used today.

Code	Description
0 (0x0000)	Unknown
1 (0x0001)	PCM/uncompressed
2 (0x0002)	Microsoft ADPCM
6 (0x0006)	ITU G.711 a-law
7 (0x0007)	ITU G.711 Âµ-law
17 (0x0011)	IMA ADPCM
20 (0x0016)	ITU G.723 ADPCM (Yamaha)
49 (0x0031)	GSM 6.10
64 (0x0040)	ITU G.721 ADPCM
80 (0x0050)	MPEG
65,536 (0xFFFF)	Experimental

Common Wave Compression Codes

Number of Channels
The number of channels specifies how many separate audio signals that are encoded in the wave data chunk. A value of 1 means a mono signal, a value of 2 means a stereo signal, etc.

Sample Rate
The number of sample slices per second. This value is unaffected by the number of channels.

Average Bytes Per Second
This value indicates how many bytes of wave data must be streamed to a D/A converter per second in order to play the wave file. This information is useful when determining if data can be streamed from the source fast enough to keep up with playback. This value can be easily calculated with the formula:

```
AvgBytesPerSec = SampleRate * BlockAlign
```

Block Align
The number of bytes per sample slice. This value is not affected by the number of channels and can be calculated with the formula:

```
BlockAlign = SignificantBitsPerSample / 8 * NumChannels
```

Significant Bits Per Sample
This value specifies the number of bits used to define each sample. This value is usually 8, 16, 24 or 32. If the number of bits is not byte aligned (a multiple of 8) then the number of bytes used per sample is rounded up to the nearest byte size and the unused bytes are set to 0 and ignored.

Extra Format Bytes
This value specifies how many additional format bytes follow. It does not exist if the compression code is 0 (uncompressed PCM file) but may exist and have any value for other compression types depending on what compression information is need to decode the wave data. If this value is not word aligned (a multiple of 2), padding should be added to the end of this data to word align it, but the value should remain non-aligned.

Data Chunk - "data"

The Wave Data Chunk contains the digital audio sample data which can be decoded using the format and compression method specified in the Wave Format Chunk. If the Compression Code is 1 (uncompressed PCM), then the Wave Data contains raw sample values. This document explains how an uncompressed PCM data is stored, but will not get into the many supported compression formats.

Wave files usually contain only one data chunk, but they may contain more than one if they are contained within a Wave List Chunk ("wavl").

Offset	Length	Type	Description	Value
0x00	4	char[4]	chunk ID	"data" (0x64617461)
0x04	4	dword	chunk size	depends on sample length and compression
0x08				sample data

Data Chunk Format

Multi-channel digital audio samples are stored as interlaced wave data which simply means that the audio samples of a multi-channel (such as stereo and surround) wave file are stored by cycling through the audio samples for each channel before advancing to the next sample time. This is done so that the audio files can be played or streamed before the entire file can be read. This is handy when playing a large file from disk (that may not completely fit into memory) or streaming a file over the Internet. The values in the diagram below would be stored in a Wave file in the order they are listed in the Value column (top to bottom).

Time	Channel	Value
0	1 (left)	0x0053
	2 (right)	0x0024
1	1 (left)	0x0057
	2 (right)	0x0029
2	1 (left)	0x0063
	2 (right)	0x003C

Interlaced Stereo Wave Samples

One point about sample data that may cause some confusion is that when samples are represented with 8-bits, they are specified as unsigned values. All other sample bit-sizes are specified as signed values. For example a 16-bit sample can range from -32,768 to +32,767 with a mid-point (silence) at 0.

As mentioned earlier, all RIFF chunks (including WAVE "data" chunks) must be word aligned. If the sample data uses an odd number of bytes, a padding byte with a value of zero must be placed at the end of the sample data. The "data" chunk header's size should not include this byte.

Fact Chunk - "fact"

A fact chunk stores compression code dependant information about the contents of the Wave file. It is required by all compressed WAVE formats and if the waveform data is contained inside a "wavl" LIST chunk, but is not required for the uncompressed PCM WAVE format files (compression code 1) that contain the waveform data inside a "data" chunk.

Offset	Size	Description	Value
0x00	4	Chunk ID	"fact" (0x66616374)
0x04	4	Chunk Data Size	depends on format
0x08		Format Dependant Data	

Fact Chunk Format

Format Dependant Data
There is currently only one field defined for the format dependant data. It is a single 4-byte value that specifies the number of samples in the waveform data chunk. This value can be used with the Samples Per Second value specified in the format chunk to calculate the waveforms length in seconds.

As new WAVE formats are introduced, the fact chunk will be expanded, appending fields after the defined number of samples field. Applications can use the fact chunk size to determine which fields are present in the chunk.

Wave List Chunk - "wavl"

A wave list chunk is used to specify several alternating "slnt" and "data" chunks. These chunks can help reduce a Wave file's size while specifying audible segments of samples when a stream of digital audio contains several periods of silence.

This type of chunk is considered to be an abuse of the Wave file format by many programmers (including myself) and it's support is not recommended. Also many programs will not recognize this type of chunk, simply ignoring it. This form of compression unnecessarily complicates the structure of a Wave file and could be better implemented in other ways, including the use of several existing compression formats.

Offset	Size	Description	Value
0x00	4	Chunk ID	"slnt" (0x736C6E74)
0x04	4	Chunk Data Size	depends on size of data and slnt chunks
0x08	List of Alternating "slnt" and "data" Chunks		

Wave List Chunk Format

Silent Chunk - "slnt"

A silent chunk is used to specify a segment of silence that will last some duration of samples. It is always contained within a wave list chunk. While this chunk represents silence, it does not necessarily define a zero volume or baseline sample. It actually holds the last sample value present in the preceding data chunk in the wave list chunk. If there is no preceding data chunk in the wave list chunk, a baseline value should be used (127 for 8-bit data, 0 for 16-bit or higher data). This may seem trivial, but if not followed, may cause undesired clicks and pops in the audio signal.

Offset	Size	Description	Value
0x00	4	Chunk ID	"slnt" (0x736C6E74)
0x04	4	Chunk Data Size	4
0x08	4	Number of Silent Samples	0 - 0xFFFFFFFF

Silent Chunk Format

Number of Silent Samples
This value specifies the number of silent samples that appear in the waveform at this point in the wave list chunk.

Cue Chunk - "cue "

A cue chunk specifies one or more sample offsets which are often used to mark noteworthy sections of audio. For example, the beginning and end of a verse in a song may have cue points to make them easier to find. The cue chunk is optional and if included, a single cue chunk should specify all cue points for the "WAVE" chunk. No more than one cue chunk is allowed in a "WAVE" chunk.

Offset	Size	Description	Value
0x00	4	Chunk ID	"cue " (0x63756520)
0x04	4	Chunk Data Size	depends on Num Cue Points
0x08	4	Num Cue Points	number of cue points in list
0x0c		List of Cue Points	

Cue Chunk Format

Chunk ID and Data Size
The chunk ID for a cue chunk is always "cue " (0x666D7420). Note that the chunk ID string ends with the space character (0x20). The chunk data size is equal to the size of the Num Cue Points value (4) plus the number of following cue points multiplied by the size of each cue point's data (24). The following formula can be used to calculate a Cue Chunk's data size:

```
ChunkDataSize = 4 + (NumCuePoints * 24)
```

Num Cue Points
This value specifies the number of following cue points in this chunk.

List of Cue Points
A list of cue points is simply a set of consecutive cue point descriptions that follow the

format described below.

Offset	Size	Description	Value
0x00	4	ID	unique identification value
0x04	4	Position	play order position
0x08	4	Data Chunk ID	RIFF ID of corresponding data chunk
0x0c	4	Chunk Start	Byte Offset of Data Chunk *
0x10	4	Block Start	Byte Offset to sample of First Channel
0x14	4	Sample Offset	Byte Offset to sample byte of First Channel

Cue Point Format

ID
Each cue point has a unique identification value used to associate cue points with information in other chunks. For example, a Label chunk contains text that describes a point in the wave file by referencing the associated cue point.

Position
The position specifies the sample offset associated with the cue point in terms of the sample's position in the final stream of samples generated by the play list. Said in another way, if a play list chunk is specified, the position value is equal to the sample number at which this cue point will occur during playback of the entire play list as defined by the play list's order. If no play list chunk is specified this value should be 0.

Data Chunk ID
This value specifies the four byte ID used by the chunk containing the sample that corresponds to this cue point. A Wave file with no play list is always "data". A Wave file with a play list containing both sample data and silence may be either "data" or "slnt".

Chunk Start
The Chunk Start value specifies the byte offset into the Wave List Chunk of the chunk containing the sample that corresponds to this cue point. This is the same chunk described by the Data Chunk ID value. If no Wave List Chunk exists in the Wave file, this value is 0. If a Wave List Chunk exists, this is the offset into the "wavl" chunk. The first chunk in the Wave List Chunk would be specified with a value of 0.

Block Start
The Block Start value specifies the byte offset into the "data" or "slnt" Chunk to the start of the block containing the sample. The start of a block is defined as the first byte in uncompressed PCM wave data or the last byte in compressed wave data where decompression can begin to find the value of the corresponding sample value.

Sample Offset
The Sample Offset specifies an offset into the block (specified by Block Start) for the sample that corresponds to the cue point. In uncompressed PCM waveform data, this is simply the byte offset into the "data" chunk. In compressed waveform data, this value is equal to the number of samples (may or may not be bytes) from the Block Start to the sample that corresponds to the cue point.

Playlist Chunk - "plst"

The playlist chunk specifies the play order of a series of cue points. The cue points are defined in the cue chunk, somewhere else in the file. A playlist consists of an array of segments, each containing information about what sample the segment should start playing from, how long the segment is (in samples) and how many times to repeat the segment before moving on to the next segment in the play order.

Offset	Size	Description	Value
0x00	4	Chunk ID	"plst" (0x736C6E74)
0x04	4	Chunk Data Size	num segments * 12
0x08	4	Number of Segments	1 - 0xFFFFFFFF
0x0a		List of Segments	

Playlist Chunk Format

Number of Segments
This value specifies the number of following segments in the playlist chunk.

List of Segments
A list of segments is simply a set of consecutive segment descriptions that follow the format described below. The segments do not have to be in any particular order because each segments associated cue point position is used to determine the play order.

Offset	Size	Description	Value
0x00	4	Cue Point ID	0 - 0xFFFFFFFF
0x04	4	Length (in samples)	1 - 0xFFFFFFFF
0x08	4	Number of Repeats	1 - 0xFFFFFFFF

Segment Format

Cue Point ID
The Cue Point ID specifies the starting sample for this segment by providing the value of a Cue Point defined in the Cue Point List. The ID that associates this segment with a Cue Point must be unique to all other segment Cue Point IDs.

Length
The Segment Length specifies the number of samples to play/loop from the starting sample defined by the associated Cue Point.

Number of Repeats
The number of repeats determines how many times this segment should be looped before playback should continue onto the next segment.

Associated Data List Chunk - "list"

An associated data list chunk is used to define text labels and names which are associated with the cue points to provide each text label or name a position.

Offset	Size	Description	Value
0x00	4	Chunk ID	"list" (0x6C696E74)
0x04	4	Chunk Data Size	depends on contained text

0x08	4	Type ID	"adtl" (0x6164746C)
0x0c		List of Text Labels and Names	

Associated Data List Chunk Format

Type ID
The type ID is used to identify the type of associated data list and is always "adtl".

List of Text Labels and Names
The list of text labels and names is a list of assorted chunks that define text in different ways. The three main chunk types that are used in WAVE files are the Label Chunk, Note Chunk and Labeled Text Chunk.

Label Chunk - "labl"

The label chunk is always contained inside of an associated data list chunk. It is used to associate a text label with a Cue Point. This information is often displayed next to markers or flags in digital audio editors.

Offset	Size	Description	Value
0x00	4	Chunk ID	"labl" (0x6C61626C)
0x04	4	Chunk Data Size	depends on contained text
0x08	4	Cue Point ID	0 - 0xFFFFFFFF
0x0c		Text	

Label Chunk Format

Cue Point ID
The Cue Point ID specifies the sample point that corresponds to this text label by providing the ID of a Cue Point defined in the Cue Point List. The ID that associates this label with a Cue Point must be unique to all other label Cue Point IDs.

Text
The text is a null terminated string of characters. If the number of characters in the string is not even, padding must be appended to the string. The appended padding is not considered in the label chunk's chunk size field.

Note Chunk - "note"

The label chunk is always contained inside of an associated data list chunk. It is used to associate a text comment with a Cue Point. This information is stored in an identical fashion to the labels in the label chunk.

Offset	Size	Description	Value
0x00	4	Chunk ID	"note" (0x6E6F7465)
0x04	4	Chunk Data Size	depends on contained text
0x08	4	Cue Point ID	0 - 0xFFFFFFFF
0x0C		Text	

Label Chunk Format

Cue Point ID
The Cue Point ID specifies the sample point that corresponds to this text comment by providing the ID of a Cue Point defined in the Cue Point List. The ID that associates this label with a Cue Point must be unique to all other note chunk Cue Point IDs.

Text
The text is a null terminated string of characters. If the number of characters in the string is not even, padding must be appended to the string. The appended padding is not considered in the note chunk's chunk size field.

Labeled Text Chunk - "ltxt"

The labeled text chunk is always contained inside of an associated data list chunk. It is used to associate a text label with a region or section of waveform data. This information is often displayed in marked regions of a waveform in digital audio editors.

Offset	Size	Description	Value
0x00	4	Chunk ID	"ltxt" (0x6C747874)
0x04	4	Chunk Data Size	depends on contained text
0x08	4	Cue Point ID	0 - 0xFFFFFFFF
0x0c	4	Sample Length	0 - 0xFFFFFFFF
0x10	4	Purpose ID	0 - 0xFFFFFFFF
0x12	2	Country	0 - 0xFFFF
0x14	2	Language	0 - 0xFFFF
0x16	2	Dialect	0 - 0xFFFF
0x18	2	Code Page	0 - 0xFFFF
0x1A			Text

Label Chunk Format

Cue Point ID
The Cue Point ID specifies the starting sample point that corresponds to this text label by providing the ID of a Cue Point defined in the Cue Point List. The ID that associates this label with a Cue Point must be unique to all other note chunk Cue Point IDs.

Sample Length
The sample length defines how many samples from the cue point the region or section spans.

Purpose ID
The purpose field specifies what the text is used for. For example a value of "scrp" means script text, and "capt" means close-caption. There are several more purpose IDs, but they are meant to be used with other types of RIFF files (not usually found in WAVE files).

Country, Language, Dialect and Code Page
These fields are used to specify information about the location and language used by the text and are typically used for queries to obtain information from the operating system.

Text
The text is a null terminated string of characters. If the number of characters in the string is not even, padding must be appended to the string. The appended padding is not considered in the note chunk's chunk size field.

Sampler Chunk - "smpl"

Offset	Size	Description	Value
0x00	4	Chunk ID	"smpl" (0x736D706C)
0x04	4	Chunk Data Size	36 + (Num Sample Loops * 24) + Sampler Data
0x08	4	Manufacturer	0 - 0xFFFFFFFF
0x0C	4	Product	0 - 0xFFFFFFFF
0x10	4	Sample Period	0 - 0xFFFFFFFF
0x14	4	MIDI Unity Note	0 - 127
0x18	4	MIDI Pitch Fraction	0 - 0xFFFFFFFF
0x1C	4	SMPTE Format	0, 24, 25, 29, 30
0x20	4	SMPTE Offset	0 - 0xFFFFFFFF
0x24	4	Num Sample Loops	0 - 0xFFFFFFFF
0x28	4	Sampler Data	0 - 0xFFFFFFFF
0x2C			List of Sample Loops

Sampler Chunk Format

Manufacturer
The manufacturer field specifies the MIDI Manufacturer's Association (MMA) Manufacturer code for the sampler intended to receive this file's waveform. Each manufacturer of a MIDI product is assigned a unique ID which identifies the company. If no particular manufacturer is to be specified, a value of 0 should be used.

The value is stored with some extra information to enable translation to the value used in a MIDI System Exclusive transmission to the sampler. The high byte indicates the number of low order bytes (1 or 3) that are valid for the manufacturer code. For example, the value for Digidesign will be 0x01000013 (0x13) and the value for Microsoft will be 0x30000041 (0x00, 0x00, 0x41). See the MIDI Manufacturers List for a list.

Product
The product field specifies the MIDI model ID defined by the manufacturer corresponding to the Manufacturer field. Contact the manufacturer of the sampler to get the model ID. If no particular manufacturer's product is to be specified, a value of 0 should be used.

Sample Period
The sample period specifies the duration of time that passes during the playback of one sample in nanoseconds (normally equal to 1 / Samplers Per Second, where Samples Per Second is the value found in the format chunk).

MIDI Unity Note
The MIDI unity note value has the same meaning as the instrument chunk's MIDI

Unshifted Note field which specifies the musical note at which the sample will be played at it's original sample rate (the sample rate specified in the format chunk).

MIDI Pitch Fraction
The MIDI pitch fraction specifies the fraction of a semitone up from the specified MIDI unity note field. A value of 0x80000000 means 1/2 semitone (50 cents) and a value of 0x00000000 means no fine tuning between semitones.

SMPTE Format
The SMPTE format specifies the Society of Motion Pictures and Television E time format used in the following SMPTE Offset field. If a value of 0 is set, SMPTE Offset should also be set to 0.

Value	SMPTE Format
0	no SMPTE offset
24	24 frames per second
25	25 frames per second
29	30 frames per second with frame dropping (30 drop)
30	30 frames per second

SMPTE Format Values

SMPTE Offset
The SMPTE Offset value specifies the time offset to be used for the synchronization / calibration to the first sample in the waveform. This value uses a format of 0xhhmmssff where hh is a signed value that specifies the number of hours (-23 to 23), mm is an unsigned value that specifies the number of minutes (0 to 59), ss is an unsigned value that specifies the number of seconds (0 to 59) and ff is an unsigned value that specifies the number of frames (0 to -1).

Sample Loops
The sample loops field specifies the number Sample Loop definitions in the following list. This value may be set to 0 meaning that no sample loops follow.

Sampler Data
The sampler data value specifies the number of bytes that will follow this chunk (including the entire sample loop list). This value is greater than 0 when an application needs to save additional information. This value is reflected in this chunks data size value.

List of Sample Loops
A list of sample loops is simply a set of consecutive loop descriptions that follow the format described below. The sample loops do not have to be in any particular order because each sample loop associated cue point position is used to determine the play order. The sampler chunk is optional.

Offset	Size	Description	Value
0x00	4	Cue Point ID	0 - 0xFFFFFFFF
0x04	4	Type	0 - 0xFFFFFFFF
0x08	4	Start	0 - 0xFFFFFFFF
0x0C	4	End	0 - 0xFFFFFFFF

0x10	4	Fraction	0 - 0xFFFFFFFF
0x14	4	Play Count	0 - 0xFFFFFFFF

Sample Loop Format

Cue Point ID
The Cue Point ID specifies the unique ID that corresponds to one of the defined cue points in the cue point list. Furthermore, this ID corresponds to any labels defined in the associated data list chunk which allows text labels to be assigned to the various sample loops.

Type
The type field defines how the waveform samples will be looped.

Value	Loop Type
0	Loop forward (normal)
1	Alternating loop (forward/backward, also known as Ping Pong)
2	Loop backward (reverse)
3 - 31	Reserved for future standard types
32 - 0xFFFFFFFF	Sampler specific types (defined by manufacturer)

Loop Type Values

Start
The start value specifies the byte offset into the waveform data of the first sample to be played in the loop.

End
The end value specifies the byte offset into the waveform data of the last sample to be played in the loop.

Fraction
The fractional value specifies a fraction of a sample at which to loop. This allows a loop to be fine tuned at a resolution greater than one sample. The value can range from 0x00000000 to 0xFFFFFFFF. A value of 0 means no fraction, a value of 0x80000000 means 1/2 of a sample length. 0xFFFFFFFF is the smallest fraction of a sample that can be represented.

Play Count
The play count value determines the number of times to play the loop. A value of 0 specifies an infinite sustain loop. An infinite sustain loop will continue looping until some external force interrupts playback, such as the musician releasing the key that triggered the wave's playback. All other values specify an absolute number of times to loop.

Instrument Chunk - "inst"

The instrument chunk is used to describe how the waveform should be played as an instrument sound. This information is useful for communicating musical information between sample-based music programs, such as trackers or software wavetables. This chunk is optional and no more than 1 may appear in a WAVE file.

Offset	Size	Description	Value

0x00	4	Chunk ID	"ltxt" (0x6C747874)
0x04	4	Chunk Data Size	7
0x08	1	Unshifted Note	0 - 127
0x09	1	Fine Tune (dB)	-50 - +50
0x0A	1	Gain	-64 - +64
0x0B	1	Low Note	0 - 127
0x0C	1	High Note	0 - 127
0x0D	1	Low Velocity	1 - 127
0x0E	1	High Velocity	1 - 127

Instrument Chunk Format

Unshifted Note
The unshifted note field has the same meaning as the sampler chunk's MIDI Unity Note which specifies the musical note at which the sample will be played at it's original sample rate (the sample rate specified in the format chunk).

Fine Tune
The fine tune value specifies how much the sample's pitch should be altered when the sound is played back in cents (1/100 of a semitone). A negative value means that the pitch should be played lower and a positive value means that it should be played at a higher pitch.

Gain
The gain value specifies the number of decibels to adjust the output when it is played. A value of 0dB means no change, 6dB means double the amplitude of each sample and -6dB means to halve the amplitude of each sample. Every additional +/-6dB will double or halve the amplitude again.

Low Note and High Note
The note fields specify the MIDI note range for which the waveform should be played when receiving MIDI note events (from software or triggered by a MIDI controller). This range does not need to include the Unshifted Note value.

Low Velocity and High Velocity
The velocity fields specify the range of MIDI velocities that should cause the waveform to be played. 1 being the lightest amount and 127 being the hardest.

Format Variations

The down side to the Wave file format's popularity is that out of the hundreds of programs that support it, many abuse or misuse it due to bad programming and/or poor documentation. Once some of these "naughty" programs get fairly popular and churn out millions of incorrect Wave files, the rest of the software industry is forced to deal with it and write code that can read the incorrect files. New code should never write these errors, but possibly read them. Below are a few exceptions that have been made to the strict/original Wave file format.

- Incorrect Block Alignment value - this can be dealt with by calculating the Block Alignment with the formula mentioned above.

- Incorrect Average Samples Per Second value - this can be dealt with by calculating the Average Samples Per Second with the formula mentioned above.
- Missing word alignment padding - this can be difficult to deal with, but can be done by giving the user a warning when unrecognized chunk ID's are encountered where a one byte read offset produces a recognized chunk ID. This is not a concrete solution, but will usually work even if the program doesn't have a comprehensive list of legal IDs.

Reacties

Je hebt geen rechten om reacties toe te voegen.