Konrad Windszus
Web Developer
Germany
14,574 points

Image Unavailable

Articles udio

MP e Header

By **Konrad Windszus**, 12 Apr 2007

4.88 (82 votes)

Download demo project (binary and source, V2.2) - 225.4 KB

Download source (V2.2) - 21.9 KB

Contents

1. Introduction

This article is about the structure of the MPEG audio frame header including the XING and VBRI headers. The aim is to estimate the duration of the MPEG audio file as exact and fast as possible. The article does not include any hints on how to decode/encode the actual audio data. MPEG audio files exist in different layers. The most common is the MPEG 1 Layer III (also known as MP3), as it has the most sophisticated compression technology.

I know that there are other articles about the MPEG audio frame header (even on CodeProject), but I will go into it a little bit deeper.

2. MPEG Audio Frame

An MPEG audio file consists out of frames. Each frame contains a header at its beginning followed by the audio data. This audio data always contains a **fixed number of samples**. There currently exists three layers of MPEG audio, which differ in how the audio data is encoded in the frame, although they all have the same header format. The frame itself consists of slots. In Layer I, a slot is always 4 byte long, in all other the layers a slot is 1 byte long.

Additional to the layers there are also three versions of MPEG audio, which differ in the sampling rate they can handle (see table 2.1.2). MPEG 1 (ISO/IEC 13818-3) and MPEG 2 (ISO/IEC 11172-3) are ISO standards. MPEG 2.5 is an unofficial extension of MPEG 2 to support even lower sampling rates. MPEG 2/2.5 is also known under the abbreviation **LSF**, which stands for Lower Sampling Frequencies. Each version can handle the three layers. If you want to know more about the technical details of an MPEG audio file please have a look at the specifications. You can find them and many other useful information about MPEG at www.MP3-Tech.org.

A file can be encoded either with a constant bitrate (**CBR**) or with a variable bitrate (**VBR**), which means that each frame can have a different bitrate. Therefore, the quality of those files is often higher than files encoded in constant a bitrate mode, because they can use higher bitrates where the music needs it.

2.1. MPEG Audio Frame Header

The header at the beginning of each frame is 32 bits long and has the following format. The bit 0 in the header is the most significant bit (**MSB**) of the complete header. Note that the position is zero-based; position, length and example are all in bit-format.

Position	Length	Meaning	Example
0	11	Frame sync to find the header (all bits are always set)	1111 1111 111
11	2	Audio version ID (see table 3.2 also) 00 - MPEG Version 2.5 (unofficial extension of MPEG 2) 01 - reserved 10 - MPEG Version 2 (ISO/IEC 13818-3) 11 - MPEG Version 1 (ISO/IEC 11172-3)	11
13	2	Layer index 00 - reserved 01 - Layer III 10 - Layer II	01

		11 - Layer I	
15	1	Protection bit 0 - protected by 16 bit CRC following header 1 - no CRC	1
16	4	Bitrate index (see table 2.1.3)	1001
20	2	Sampling rate index (see table 2.1.2)	11
22	1	Padding bit If it is set, data is padded with with one slot (important for frame size calculation)	0
23	1	Private bit (only informative)	1
24	2	Channel mode 00 - Stereo 01 - Joint Stereo (Stereo) 10 - Dual channel (Two mono channels) 11 - Single channel (Mono) Note: Dual channel files are made of two independent mono channels. Each one uses exactly half the bitrate of the file. Most decoders output them as stereo, but it might not always be the case.	01
26	2	Mode extension (Only used in Joint Stereo) (see table 2.1.6)	00
28	1	Copyright bit (only informative)	1
29	1	Original bit (only informative)	1
30	2	Emphasis 00 - none 01 - 50/15 ms 10 - reserved 11 - CCIT J.17 The emphasis indication is here to tell the decoder that the file must be de-emphasized, that means the decoder must 're-equalize' the sound after a Dolby-like noise suppression. It is rarely used.	00

2.1.1 MPEG Audio Frame Header

The sampling rate specifies how many samples per second are recorded. Each MPEG version can handle different sampling rates.

Sampling Rate Index	MPEG 1	MPEG 2 (LSF)	MPEG 2.5 (LSF)
00	44100 Hz	22050 Hz	11025 Hz
01	48000 Hz	24000 Hz	12000 Hz
10	32000 Hz	16000 Hz	8000 Hz

11	reserved

2.1.2 MPEG Versions and Sampling Rates

The bitrates are always displayed in kilobits per second. Note that the prefix kilo (abbreviated with the small 'k') doesn't mean 1024 but 1000 bits per second! The bitrate index 1111 is reserved and should never be used. In the MPEG audio standard there is a free format described. This free format means that the file is encoded with a constant bitrate, which is not one of the predefined bitrates. Only very few decoders can handle those files.

Bitrate Index	MPEG 1			MPEG 2, 2.5 (LSF)	
	Layer I	Layer II	Layer III	Layer I	Layer II & III
0000	free				
0001	32	32	32	32	8
0010	64	48	40	48	16
0011	96	56	48	56	24
0100	128	64	56	64	32
0101	160	80	64	80	40
0110	192	96	80	96	48
0111	224	112	96	112	56
1000	256	128	112	128	64
1001	288	160	128	144	80
1010	320	192	160	160	96
1011	352	224	192	176	112
1100	384	256	224	192	128
1101	416	320	256	224	144
1110	448	384	320	256	160
1111	reserved				

2.1.3 Bitrates (in kilobits per second)

In MPEG 1 Layer II, there are only some combinations of bitrates and modes allowed. In MPEG 2/2.5, there is no such restriction.

Bitrate	Allowed modes
free	all
32	single channel
48	single channel
56	single channel
64	all
80	single channel
96	all
112	all
128	all

160	all
192	all
224	stereo, intensity stereo, dual channel
256	stereo, intensity stereo, dual channel
320	stereo, intensity stereo, dual channel
384	stereo, intensity stereo, dual channel

2.1.4 Allowed bitrate and mode combinations

For the calculation of the frame size, you need the number of samples per MPEG audio frame. Therefore, you can use the following table:

	MPEG 1	MPEG 2 (LSF)	MPEG 2.5 (LSF)
Layer I	384	384	384
Layer II	1152	1152	1152
Layer III	1152	576	576

2.1.5 Samples Per Frame

Then you can calculate the frame size like this:

```
Frame Size = ( (Samples Per Frame / 8 * Bitrate) / Sampling Rate) + Padding Size
```

Because of rounding errors, the official formula to calculate the frame size is a little bit different. According to the ISO standards, you have to calculate the frame size in slots (see 2. MPEG Audio Format), then truncate this number to an integer, and after that multiply it with the slot size. You can find the correct way of calculating the frame size in the class `CMPAHeader` in my code.

You get the duration of the file in seconds by applying the following formula:

```
Duration = File Size / Bitrate * 8
```

The method of getting the first frame header in the file and then calculating the duration by the above formula works correctly only for CBR files.

The mode extension is used to join information that is of no use for the stereo effect, thus reducing the needed bits. These bits are dynamically determined by an encoder in the Joint Stereo mode, and Joint Stereo can be changed from one frame to another, or even switched on or off. For all other channel modes, the mode extension field is invalid.

The complete frequency range of MPEG audio files is divided into subbands. There are 32 subbands. For Layers I & II, the two bits in the header determine the frequency range (bands) where the intensity stereo is applied. Within this frequency range, only one channel is stored. All other bands contain information in two separate channels. For Layer III, these two bits determine which type of joint stereo is used (intensity stereo and/or M/S stereo).

Value	Layer I & II	Layer III	
		M/S stereo	Intensity stereo
00	bands 4 to 31	off	off
01	bands 8 to 31	off	on
10	bands 12 to 31	on	off
11	bands 16 to 31	on	on

2.1.6 Mode Extension

2.2. Verifying CRC

If the protection bit in the header is not set, the frame contains a 16 bit CRC (Cyclic Redundancy Checksum). This checksum directly follows the frame header and is a big-endian `WORD`. To verify this checksum you have to calculate it for the frame and compare the calculated CRC with the stored CRC. If they aren't equal probably a transfer error has appeared. It is also helpful to check the CRC to verify that you really found the beginning of a frame, because the sync bits do in same cases also occur within the data section of a frame.

The CRC is calculated by applying the CRC-16 algorithm (with the generator polynom 0x8005) to a part of the frame. The following data is considered for the CRC: the last two bytes of the header and a number of bits from the audio data which follows the checksum after the header. The checksum itself must be skipped for CRC calculation. Unfortunately there is no easy way to compute the number of frames which are necessary for the checksum calculation in Layer II. Therefore I left it out in the code. You would need other information apart from the header to calculate the necessary bits. However it is possible to compute the number of protected bits in Layer I and Layer III only with the information from the header.

For Layer III, you consider the complete side information for the CRC calculation. The side information follows the header or the CRC in Layer III files. It contains information about the general decoding of the frame, but doesn't contain the actual encoded audio samples. The following table shows the size of the side information for all Layer III files.

	MPEG 1	MPEG 2/2.5 (LSF)
Stereo, Joint Stereo, Dual Channel	32	17
Mono	17	9

2.2.1 Layer III side information size (in bytes)

For Layer I files, you must consider the mode extension (see table 2.1.6) from the header. Then you can calculate the **number of bits** which are necessary for CRC calculation by applying the following formula:

```
4 * (number of channels * bound of intensity stereo +
                   (32 - bound of intensity stereo));
```

This can be read as two times the number of stereo subbands plus the number of mono subbands and the result multiplied with 4. For simple mono frames, this equals 128, because the number of channels is one and the bound of intensity stereo is 32, meaning that there is no intensity stereo. For stereo frames this is 256. For more information have a look at the CRC code in the class `CMPAFrame`.

2.3. VBR Headers

Some files are encoded with variable bitrate mode (VBR). To estimate the duration of those files, you have to know the **average bitrate** of the whole file. It often differs a lot from the bitrate of the first frame, because the lowest bitrate available is used for silence in music titles (especially at the beginning). To get this average bitrate, you must go through all the frames in the file and calculate it, by summarizing the bitrates of each frame and dividing it through the number of frames. Because this isn't a good practice (very slow), there exists additional VBR headers within the data section of the first frame (after the frame header). They contain the total number of frames in the file from which you can calculate the duration in seconds with the following formula:

```
Duration = Number of Frames * Samples Per Frame / Sampling Rate
```

Additional to that, the VBR header often contains a table which is necessary to seek positions within the file.

2.3.1 XING Header

This header is often (but unfortunately not always) added to files which are encoded with variable bitrate mode. This header stands after the first MPEG audio header at a specific position. The whole first frame which contains the XING header is a valid but empty audio frame, so even decoders which don't consider this header can decode the file. The XING header stands after the side information in Layer III files. So you can calculate the beginning of a XING header relative to the beginning of the frame by adding 4 bytes (for the MPEG audio header) to the values from the table 2.2.1. The offset calculation doesn't consider the 16 bit CRC following the header and is equal for all Layers, although only Layer III has a side information.

For reading out this header, you have to find the first MPEG audio header and then go to this specific position within the frame. The XING header itself has the following format. (Note that the position is zero-based; position, length and example are each in byte-format.)

Position	Length	Meaning	Example
0	4	VBR header ID in 4 ASCII chars, either 'Xing' or 'Info', not NULL-terminated	'Xing'
4	4	Flags which indicate what fields are present, flags are combined with a logical OR. Field is mandatory. 0x0001 - Frames field is present 0x0002 - Bytes field is present 0x0004 - TOC field is present 0x0008 - Quality indicator field is present	0x0007 (means Frames, Bytes & TOC valid)
8	4	Number of **Frames** as Big-Endian `DWORD` (optional)	7344
8 or 12	4	Number of **Bytes** in file as Big-Endian `DWORD` (optional)	45000
8, 12 or 16	100	100 **TOC** entries for seeking as integral `BYTE` (optional)	
8, 12, 16, 108, 112 or 116	4	**Quality indicator** as Big-Endian `DWORD` from 0 - best quality to 100 - worst quality (optional)	0

2.3.1.1 XING Header

According to this format, a XING header must only contain the ID and the flags. All other fields are optional and depend on the flags which are set. Sometimes this header is also added to CBR files. It then often has the ID 'Info' instead of 'Xing'.

There exists the **LAME extension** to this header, which is used by the common LAME Encoder, but I didn't take it into account because it isn't necessary for duration estimation. Nonetheless, here is the link for the documentation of the MP3 Info Tag.

2.3.2 VBRI Header

This header is only used by MPEG audio files encoded with the Fraunhofer Encoder as far as I know. It is different from the XING header. You find it exactly **32 bytes** after the end of the first MPEG audio header in the file. (Note that the position is zero-based; position, length and example are each in byte-format.)

Position	Length	Meaning	Example
0	4	VBR header ID in 4 ASCII chars, always 'VBRI', not NULL-terminated	'VBRI'
4	2	Version ID as Big-Endian `WORD`	1
6	2	Delay as Big-Endian `float`	7344
8	2	Quality indicator	75
10	4	Number of **Bytes** as Big-Endian `DWORD`	45000
14	4	Number of **Frames** as Big-Endian `DWORD`	7344
18	2	Number of entries within TOC table as Big-Endian `WORD`	100
20	2	Scale factor of TOC table entries as Big-Endian `DWORD`	1
22	2	Size per table entry in bytes (max 4) as Big-Endian `WORD`	2
24	2	Frames per table entry as Big-Endian `WORD`	845
26		**TOC** entries for seeking as Big-Endian integral. From size per table entry and number of entries, you can calculate the length of this field.	

2.3.2.1 VBRI Header

3. Additional Tags

Please consider that at the end or at the beginning of the file, there might be additional data which is not part of the MPEG audio frames. This data is called a tag, because it contains metadata about the file, like title, artist, track, years etc. You must consider these tags, because only the MPEG audio data count for the duration estimation.. At the end of a file there might be an ID3V1 tag, a Lyrics3 tag or a Musicmatch tag. At the beginning or at the end of the file there might be an ID3V2 tag and/or an APE tag. You find information about ID3 and Lyrics3 tags at www.id3.org. APE was originally developed for lossless compressed audio files in APE (Monkey's Audio) format. But now this tag is also used for MPEG audio files. The Musicmatch tag was used by older version of the Musicmatch Encoder. Unfortunately there is very little information on the web about this tag, since it is a proprietary format of Musicmatch.

4. Using The Code

I wrote some C++ classes to handle the MPEG audio frame header and the VBR headers. The class `CMPAFile` represents the whole file, and provides methods for accessing a specific frame (`CMPAFrame` class). This class instantiates a `CMPAHeader` class which represents the MPEG audio frame header of the frame itself. All fields from the header can be accessed via the class variables. Additionally, there is the `CVBRHeader` class which is the generalization of `CXINGHeader` and of `CVBRIHeader`. All tags are derived from the class `CTag`. These classes are completely independent of any libraries like MFC or ATL. They only use the Win32 API. All classes will throw exceptions in case of errors. All exceptions are of the type `CMPAException`. You can display information about the exception by using the `ShowError()` method. All classes use `CMPAStream` for file-access. This class includes a simple buffer.

Here is a code snippet which demonstrates how you could use these classes. Note that you must at least include the header *mpafile.h*:

```
try {
    // open file and look for first header
    CMPAFile MPAFile(_T("C:\\test.mp3")) ;
    cout << MPAFile.GetLengthSec() << _T(" seconds");
}
catch(CMPAException& Exc)
{
    // show error message
    Exc.ShowError();
}
```

For more information, have a look at the source of the demo project **MPEG Audio Info**, which is a simple MFC Dialog Application, which uses the class `CMPAFile` for getting information about a MPEG audio file. It also can perform a check of a whole file for errors within the frame structure.

5. Links And Miscellaneous

Here, you can find the sources which I used for this article:

- MPEG 1 Specification (ISO/IEC 13818-3).
- MPEG 2 Specification (ISO/IEC 11172-3).
- MPEG Audio Header specification from MP3-Tech.org.
- XING Header SDK from Real Networks.
- MP3 Info Tag.
- VBRI Header SDK from Fraunhofer Institute.
- MP3 sample files with different headers and tags.
- ID3 & Lyrics tag.
- APE tag.
- Musimatch tag.

If you find any mistakes in the code or in the article, or have suggestions for improvements, just post them to the forum below or write me an e-mail to webmaster@wincd.de.

6. History

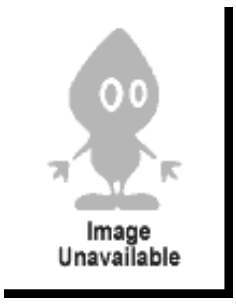
- 2004-09-15
  - **Version 1.0 of MPEG Audio Info** released.
  - **First version of the article** published.
- 2004-11-01
  - **Version 2.0 of MPEG Audio Info** released.
    - added possibility to get information about arbitrary frames (not just the first one).
    - added checking of MPEG audio files.
    - better handling of faulty MPEG files (tolerance range +/-3 bytes to look for the next frame).
    - padding is detected correctly.
    - improved buffer management.
    - fixed some other small issues.
    - added a readme file.
  - **Article updated**:
    - added information about mode extension field.
    - XING header offsets fixed (Mono and Stereo inverted).
    - frames and bytes in VBRI fixed (they were inverted).
    - position of quality indicator in VBRI header fixed.
    - corrected some typos.

- 2005-11-17
  - **Version 2.1 of MPEG Audio Info** released.
    - cleaned up class architecture.
    - better navigation through frames.
    - improved checking of files.
    - fixed original bit checking.
    - fixed frame size calculation for all layers (correct truncation).
    - fixed skipping of small frames in VBR files.
    - new cache management.
    - added CRC check.
    - added detection of ID3V1/V2, APE and Lyrics3 tags.
  - **Article updated:**
    - added information about CRC.
    - fixed sampling rate table.
    - some minor enhancements.
- 2007-04-09
  - **Version 2.2 of MPEG Audio Info** released.
    - added Musicmatch-Tag detection
    - better exception handling
    - improved design of CMPAStream
    - added Drag&Drop functionality to dialog
    - fixed memory leaks
    - updated solution to Visual Studio 2005
    - better performance for finding frames
  - **Article updated:**
    - fixed broken links
    - added link with MPEG Audio Specifications
    - added link with informations about Musicmatch

License

This article, along with any associated source code and files, is licensed under The GNU Lesser General Public License (LGPLv3)

About the Author



Konrad Windszus

Web Developer
Germany

Author of the shareware WinCD.

Comments and Discussions

127 messages have been posted for this article Visit **http://www.codeproject.com/Articles/8295/MPEG-Audio-Frame-Header** to post and view comments on this article, or click **here** to get a print view with messages.

Permalink | Advertise | Privacy | Mobile
Web03 | 2.7.131219.1 | Last Updated 12 Apr 2007

Article Copyright 2004 by Konrad Windszus
Everything else Copyright © CodeProject, 1999-2013
Terms of Use