Specification of the Broadcast Wave Format

A format for audio data files in broadcasting

Supplement 1 – MPEG audio

Tech. 3285–E – Supplement 1 July 1997

CONTENTS

European Broadcasting Union
Case Postale 67, CH–1218 Grand–Saconnex (Geneva) Switzerland

Specification of the Broadcast Wave Format

MPEG audio

1. Introduction

The Broadcast Wave Format is based on the Microsoft WAVE audio file format which is one type of file specified in the Microsoft "Resource Interchange File Format", RIFF [1] specifically to contain audio data. The basic building block of RIFF files is a chunk which consists of a specific collection of data with an identifying label and a size field.

For the BWF, some restrictions are applied to the original WAVE format. In addition the BWF file includes a Broadcast Audio Extension chunk. This illustrated in *Fig. 1* .

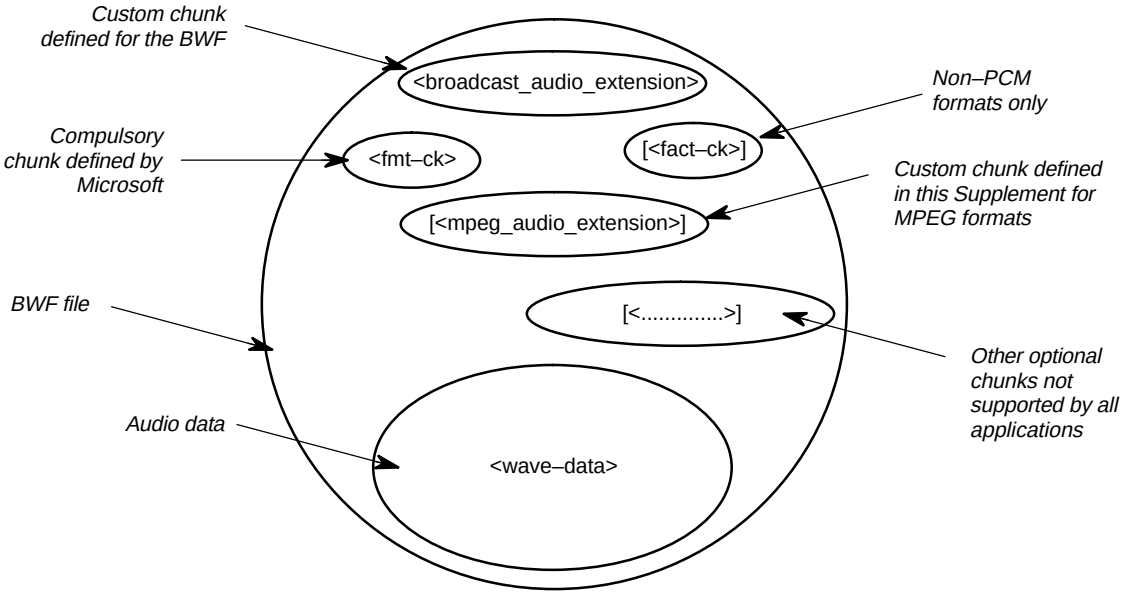

Fig. 1 Broadcast wave format file: MPEG audio only.

This Supplement contains the specification for the use of the BWF to carry MPEG–audio–only signals. For MPEG audio, it is necessary to add to the basic chunks specified in the main part of this document, the following information:

– an extension to the format chunk <fmt–ck>;

– a fact chunk <fact–ck>;

– an <mpeg_audio_extension> chunk.

The extension to the format chunk and the fact chunk are both specified as part of the WAVE format and the relevant information is given in *Appendix A* to this Supplement.

The specification of the <mpeg_audio_extension>chunk is given in *Section 2* of this Supplement.

The main part of this document contains the specification of the <broadcast_audio_extension>chunk which is used in all BWF files. Information on the basic RIFF format is given in *Appendix A* to the parent specification [2].

2. MPEG audio

The Microsoft Corporation has specified how MPEG audio data can be organised in WAVE files. An extension to the format chunk and a fact chunk carry further information needed to specify MPEG coding options. The general principles are given in *Appendix A* to the parent specification [2] and the details are given in *Appendix A* to this Supplement. For MPEG Layer 2, it has been found that extra information needs to be carried about the coding of the signal. This is carried in the <mpeg_audio_extension> chunk, developed by the MPEG Layer 2 Audio Interest group. This chunk is specified below.

2.1. <mpeg_audio_extension> chunk

The <mpeg_audio_extension>chunk is defined as follows:

```
typedef struct {
     DWORD      ckID;                    /* (mpeg_extension)ckID='mext' */
     DWORD      ckSize;                  /* size of extension chunk: cksize =000C*/
     BYTE       ckData[ckSize];          /* data of the chunk */
}
typedef struct mpeg_audio_extension {
WORD SoundInformation;                  /* more information about sound */
WORD FrameSize;                         /* nominal size of a frame */
WORD AncillaryDataLength;               /* Ancillary data length */
WORD AncillaryDataDef;                  /* Type of ancillary data */
CHAR Reserved [4];                      /* Reserved for future use; set to null */
} MPEG_EXT ;
```

Field *Description*

`SoundInformation` 16 bits giving additional information about the sound file:

For MPEG Layer 2 (or Layer 1):

Bit 0: '1' Homogeneous sound data
'0' Non homogeneous sound data

Bits 1 and 2 are used for additional information for homogeneous sound files:

Bit 1: '0' Padding bit is used in the file so may alternate between '0'or '1'
'1' Padding bit is set to '0' in the whole file

Bit 2: '1' The file contains a sequence of frames with padding bit set to '0' and sample frequency equal to 22.05 or 44.1 kHz (See *Section 2.2*).

Bit 3: '1' Free format is used
'0' No free format audio frame.

`FrameSize` 16 bit number of bytes of a nominal frame.

This field has a meaning only for homogeneous files, otherwise it is set to ’0’.

If the padding bit is not used, i.e. it remains constant in all frames of the sound file, the field `<FrameSize>` contains the same value as the field `<nBlockAlign>` in the format chunk. If padding bit is used and variable lengths occur in sound data, `<FrameSize>` contains the size of a frame with padding bit set to ’0’. The length of a frame with padding bit set to ’1’ is one byte more (four bytes for Layer I), i.e. `FrameSize+1`.

The fact that `<nBlockAlign>` is set to ’1’ means variable frame lengths (`FrameSize` or `FrameSize+1`) with variable padding bit.

`AncillaryDataLength` 16 bit number giving minimal number of known bytes for ancillary data in the full sound file. The value is relative from the end of the audio frame.

`AncillaryDataDef` This 16 bit value specifies the content of the ancillary data with:

Bit 0	set to ’1’ :	Energy of left channel present in ancillary data
Bit 1	set to ’1’ :	A private byte is free for internal use in ancillary data
Bit 2	set to ’1’ :	Energy of right channel present in ancillary data
Bit 3:	set to ’0’:	reserved for future use for ADR data
Bit 4:	set to ’0’:	reserved for future use for DAB data
Bit 5:	set to ’0’:	reserved for future use for J 52 data
Bits 6 to 15:	set to ’0’:	reserved for future use

Notes:

The items present in the ancillary data follow the same order as the bit numbers in `AncillaryDataDef`. The first item is stored at the end of the ancillary data, the second item is stored just before the first, etc., moving from back to front.

For a mono file bit 2 is always set to ’0’ and bit 0 concerns the energy of the mono frame.

For a stereo file if bit 2 equals ’0’ and bit 0 equals ’1’ the energy concerns the maximum of left and right energy.

The energy is stored in 2 bytes and corresponds to the absolute value of the maximum sample used to code the frame. This is a 15–bit value in Big Endian format.

`Reserved` 4 bytes reserved for future use. These 4 bytes must be set to NULL. In any future use, the NULL value will be used for the default value to maintain compatibility.

2.2. Homogeneous sound data

For homogeneous sound data, MPEG frames in the sound file must have the *same* frame header except for the following cases:

– The padding bit (Pd) may alternate between ’0’ and ’1’ if the sampling frequency is 44.1 kHz or 22.05 kHz or if the field bit–rate is set to free format (’0000’).

– The mode field (M) may vary between stereo (M=’00’) , joint stereo (M=’01’) and dual channel (M=’10’). If the mode field is single channel (M=’11’), it must remain the same throughout the file.

– For frames in joint stereo mode (M=’01’), the mode extension field (Mx) may vary throughout the file.

– The private bit (Pr), which is not specified by ISO, may vary throughout the file.

It is important to note that, except for 44.1 kHz or 22.05 kHz sampling frequencies or free format, every frame in the file has the same length.

Free format is allowed in a file with frame length equal to N or N+p bytes.

When the length is N, the padding bit is set to ’0’.

When the length is N+p, the padding bit is set to ’1’.

The value of ’N’ depends on the bit–rate index and sampling frequency.

The value of ’p’ is ’1’ for layer 2 compression and ’4’ for layer 1 compression.

2.3. Non homogeneous sound data

A non homogeneous file contains a sequence of MPEG frames without any restriction.

Appendix A

Extract from RIFF WAVE (.WAV) file format

The information in this Appendix is taken from the specification documents of the Microsoft RIFF file format. *It is included for information only.*

For full information, consult the latest version of the Microsoft Software Developers Kit Multimedia Standards Update, (rev 3.0, 15 April 1994 or later) [1].

[EBU Note: This section gives the specification of the extra information necessary for a WAVE file containing MPEG Audio.]

A1. MPEG–1 audio (audio–only)

A1.1. Fact chunk

The fact chunk `<fact–ck>` is required for all WAVE formats other than `WAVE_FORMAT_PCM` It stores file–dependent information about the contents of the WAVE data. It currently specifies the time length of the data in samples.

[EBU Note: See also Section A2.5 of the Appendix to the parent specification [2]]

A1.2. WAVE format header

```
#define    WAVE_FORMAT_MPEG(0x0050)


typedef struct mpeg1waveformat_tag {
     WAVEFORMATEX    wfx;
     WORD            fwHeadLayer;
     DWORD           dwHeadBitrate;
     WORD            fwHeadMode;
     WORD            fwHeadModeExt;
     WORD            wHeadEmphasis;
     WORD            fwHeadFlags;
     DWORD           dwPTSLow;
     DWORD           dwPTSHigh;
} MPEG1WAVEFORMAT;
```

`wFormatTag` This must be set to `WAVE_FORMAT_MPEG`[0x00 50].

`nChannels` Number of channels in the wave: 1 for mono, 2 for stereo.

`nSamplesPerSec` Sampling frequency (in Hz) of the wave file: 32000, 44100, or 48000 etc. Note, however, that if the sampling frequency of the data is variable, then this field should be set to zero. It is strongly recommended that a fixed sampling frequency be used for desktop applications.

[EBU Note: For broadcast use the sampling frequency is normally 48 kHz.]

`nAvgBytesPerSec` Average data–rate; this might not be a legal MPEG bit–rate if variable bit–rate coding under layer 3 is used.

`nBlockAlign` The block alignment (in bytes) of the data in `<data-ck>`. For audio streams which have a fixed audio frame length, the block alignment is equal to the length of the frame. For streams in which the frame length varies, `<nBlockAlign>` should be set to 1.

With a sampling frequency of 32 or 48 kHz, the size of an MPEG audio frame is a function of the bit–rate. If an audio stream uses a constant bit–rate, the size of the audio frames does not vary. Therefore, the following formulas apply:

Layer 1: `nBlockAlign` = 4*(int)(12*BitRate/SamplingFreq)

Layers 2 and 3: `nBlockAlign` = (int)(144*BitRate/SamplingFreq)

Example: For layer 1, with a sampling frequency of 32000 Hz and a bit rate of 256 kbits/s, `nBlockAlign` = 384 bytes.

If an audio stream contains frames with different bit rates, then the length of the frames varies within the stream. Variable frame lengths also occur when using a sampling frequency of 44.1 kHz: in order to maintain the data rate at the nominal value, the size of an MPEG audio frame is periodically increased by one "slot" (4 bytes in layer 1, 1 byte in layers 2 and 3) as compared to the formulas given above. In these two cases, the concept of block alignment is invalid. The value of `<nBlockAlign>` must therefore be set to 1, so that MPEG–aware applications can tell whether the data is block–aligned or not.

Note that it is possible to construct an audio stream which has constant–length audio frames at 44.1 kHz by setting the padding bit in each audio frame header to the same value (either 0 or 1). Note, however, that bit–rate of the resulting stream will not correspond exactly to the nominal value in the frame header, and therefore some decoders may not be capable of decoding the stream correctly. In the interests of standardization and compatibility, this approach is discouraged.

`wBitsPerSample` Not used; set to zero.

`cbSize` The size in bytes of the extended information after the `WAVEFORMATEX` structure. For the standard `WAVE_FORMAT_MPEG` format, this is 22 (0x00 16). If extra fields are added, this value will increase.

`fwHeadLayer` The MPEG audio layer, as defined by the following flags:

`ACM_MPEG_LAYER1` – layer 1
`ACM_MPEG_LAYER2` – layer 2
`ACM_MPEG_LAYER3` – layer 3

Some legal MPEG streams may contain frames of different layers. In this case, the above flags should be ORed together so that a driver may tell which layers are present in the stream.

`dwHeadBitrate` The bit–rate of the data, in bits per second. This value must be a standard bit–rate according to the MPEG specification; not all bit–rates are valid for all modes and layers. See *Tables 1* and *2*. Note that this field records the actual bit–rate, not the MPEG frame header code. If the bit rate is variable, or if it is a non–standard bit–rate, then this field should be set to zero. It is recommended that variable bit–rate coding be avoided where possible.

Table 1 – Allowable bit–rates (bit/s).

MPEG frame header code	Layer 1	Layer 2	Layer 3
’0000’	free format	free format	free format
’0001’	32000	32000	32000
’0010’	64000	48000	40000
’0011’	96000	56000	48000
’0100’	128000	64000	56000
’0101’	160000	80000	64000
’0110’	192000	96000	80000
’0111’	224000	112000	96000
’1000’	256000	128000	112000
’1001’	288000	160000	128000
’1010’	320000	192000	160000
’1011’	352000	224000	192000
’1100’	384000	256000	224000
’1101’	416000	320000	256000
’1110’	448000	384000	320000
’1111’	forbidden	forbidden	forbidden

Table 2 – Allowable mode/bit–rate combinations for Layer 2.

Bit–rate (bit/sec)	**Allowable modes**
32000	single channel
48000	single channel
56000	single channel
64000	all modes
80000	single channel
96000	all modes
112000	all modes
128000	all modes
160000	all modes
192000	all modes
224000	stereo, intensity stereo, dual channel
256000	stereo, intensity stereo, dual channel
320000	stereo, intensity stereo, dual channel
384000	stereo, intensity stereo, dual channel

`fwHeadMode` Stream mode, as defined by the following flags:

`ACM_MPEG_STEREO`	–	stereo.
`ACM_MPEG_JOINTSTEREO`	–	joint–stereo.
`ACM_MPEG_DUALCHANNEL`	–	dual–channel (for example, a bilingual stream).
`ACM_MPEG_SINGLECHANNEL`	–	single channel.

Some legal MPEG streams may contain frames of different modes. In this case, the above flags should be ORed together so that a driver may tell which modes are present in the stream. This situation is particularly likely with joint–stereo encoding, as encoders may find it useful to switch dynamically between stereo and joint–stereo according to the characteristics of the signal. In this case, both the `ACM_MPEG_STEREO` and the `ACM_MPEG_JOINTSTEREO` flags should be set.

`fwHeadModeExt` Contains extra parameters for joint–stereo coding; not used for other modes. See *Table 3*. Some legal MPEG streams may contain frames of different mode extensions. In this case, the values in *Table 3* may be ORed together. Note that `fwHeadModeExt` is only used for joint–stereo coding; for other modes (single channel, dual channel, or stereo), it should be set to zero.

In general, encoders will dynamically switch between the various possible MPEG mode_extension values according to the characteristics of the signal. Therefore, for normal joint–stereo encoding, this field should be set to 0x000f. However, if it is desirable to limit the encoder to a particular type of joint–stereo coding, this field may be used to specify the allowable types.

`wHeadEmphasis` Describes the de–emphasis required by the decoder; this implies the emphasis performed on the stream prior to encoding. See *Table 4*.

Table 3 – Mode extension.

`fwHeadModeExt`	MPEG frame header code	Layers 1 and 2	Layer 3
0x0001	'00'	sub–bands 4–31 in intensity stereo	no intensity *or* MS–stereo coding
0x0002	'01'	sub–bands 8–31 in intensity stereo	intensity stereo
0x0004	'10'	sub–bands 12–31 in intensity stereo	MS–stereo
0x0008	'11'	sub–bands 16–31 in intensity stereo	both intensity *and* MS–stereo coding

Table 4 – Emphasis field.

`wHeadEmphasis`	MPEG frame header code	De–emphasis required
1	'00'	no emphasis
2	'01'	50/15 ms emphasis
3	'10'	reserved
4	'11'	CCITT J.17

`fwHeadFlags` Sets the corresponding flags in the audio frame header:

`ACM_MPEG_PRIVATEBIT`	–	set the private bit.
`ACM_MPEG_COPYRIGHT`	–	set the copyright bit.
`ACM_MPEG_ORIGINALHOME`	–	sets the original/home bit.
`ACM_MPEG_PROTECTIONBIT`	–	sets the protection bit, and inserts a 16–bit error protection code into each frame.
`ACM_MPEG_ID_MPEG1`	–	sets the ID bit to 1, defining the stream as an MPEG–1 audio stream. *This flag must always be set explicitly to maintain compatibility with future MPEG audio extensions (i.e. MPEG–2).*

An encoder will use the value of these flags to set the corresponding bits in the header of each MPEG audio frame. When describing an encoded data stream, these flags represent a logical OR of the flags set in each frame header. That is, if the copyright bit is set in one or more frame headers in the stream, then the `ACM_MPEG_COPYRIGHT` flag will be set. Therefore, the value of these flags is not necessarily valid for every audio frame.

`dwPTSLow` This field (together with the following field) consists of the presentation time stamp (PTS) of the first frame of the audio stream, as taken from the MPEG system layer. `dwPTSLow` contains the 32 LSBs of the 33–bit PTS. The PTS may be used to aid in the re–integration of an audio stream with an associated video stream. If the audio stream is not associated with a system layer, then this field should be set to zero.

`dwPTSHigh` This field (together with the `dwPTSLow` field) consists of the presentation time stamp (PTS) of the first frame of the audio stream, as taken from the MPEG system layer. The LSB of `dwPTSHigh` contains the MSB of the 33–bit PTS. The PTS may be used to aid in the re–integration of an audio stream with an associated video stream. If the audio stream is not associated with a system layer, then this field should be set to zero.

Note: The previous two fields (`dwPTSLow` and `dwPTSHigh`) can be treated as a single 64–bit integer; optionally, the `dwPTSHigh` field can be tested as a flag to determine whether the MSB is set or cleared.

A1.3. Flags used in data fields

`fwHeadLayer`

The following flags are defined for the `<fwHeadLayer>` field. For encoding, one of these flags should be set so that the encoder knows what layer to use. For decoding, the driver can check these flags to determine whether it is capable of decoding the stream. Note that a legal MPEG stream may use different layers in different frames within a single stream. Therefore, more than one of these flags may be set.

```
#define ACM_MPEG_LAYER1                 (0x0001)
#define ACM_MPEG_LAYER2                 (0x0002)
#define ACM_MPEG_LAYER3                 (0x0004)
```

`fwHeadMode`

The following flags are defined for the `<fwHeadMode>` field. For encoding, one of these flags should be set so that the encoder knows what layer [mode ?] to use; for joint–stereo encoding, typically the `ACM_MPEG_STEREO` and `ACM_MPEG_JOINTSTEREO` flags will both be set so that the encoder can use joint–stereo coding only when it is more efficient than stereo. For decoding, the driver can check these flags to determine whether it is capable of decoding the stream. Note that a legal MPEG stream may use different layers in different frames within a single stream. Therefore, more than one of these flags may be set.

```
#define ACM_MPEG_STEREO                 (0x0001)
#define ACM_MPEG_JOINTSTEREO            (0x0002)
#define ACM_MPEG_DUALCHANNEL            (0x0004)
#define ACM_MPEG_SINGLECHANNEL          (0x0008)
```

`fwHeadModeExt`

Table 3 defines flags for the <fwHeadModeExt> field. This field is only used for joint–stereo coding; for other encoding modes, this field should be set to zero. For joint–stereo encoding, these flags indicate the types of joint–stereo encoding which an encoder is permitted to use. Normally, an encoder will dynamically select the mode extension which is most appropriate for the input signal; therefore, an application would typically set this field to 0x000f so that the encoder may select between all possibilities; however, it is possible to limit the encoder by clearing some of the flags. For an encoded stream, this field indicates the values of the MPEG mode_extension field which are present in the stream.

`fwHeadFlags`

The following flags are defined for the <fwHeadFlags> field. These flags should be set before encoding so that the appropriate bits are set in the MPEG frame header. When describing an encoded MPEG audio stream, these flags represent a logical OR of the corresponding bits in the header of each audio frame. That is, if the bit is set in any of the frames, it is set in the <fwHeadFlags> field. If an application wraps a RIFF WAVE header around a pre–encoded MPEG audio bit stream, it is responsible for parsing the bit stream and setting the flags in this field.

```
#define ACM_MPEG_PRIVATEBIT             (0x0001)
#define ACM_MPEG_COPYRIGHT              (0x0002)
#define ACM_MPEG_ORIGINALHOME           (0x0004)
#define ACM_MPEG_PROTECTIONBIT          (0x0008)
#define ACM_MPEG_ID_MPEG1               (0x0010)
```

A1.4. Audio data in MPEG files

The data chunk <data-ck> consists of an MPEG1 audio sequence as defined by the ISO 11172 Standard, Part 3 (Audio) [3]. This sequence consists of a bit stream, which is stored in the data chunk as an array of bytes. Within a byte, the MSB is the first bit of the stream, and the LSB is the last bit. The data is not byte–reversed. For example, the following data consists of the first 16 bits (from left to right) of a typical audio frame header:

Syncword	*ID*	*Layer*	*ProtectionBit*	*...*
111111111111	1	10	1	...

This data would be stored in bytes in the following order:

Byte0	*Byte1*	*...*
FF	FD	...

a) MPEG audio frames

An MPEG audio sequence consists of a series of audio frames, each of which begins with a frame header. Most of the fields within this frame header correspond to fields in the MPEG1WAVEFORMAT structure defined above. For encoding, these fields can be set in the MPEG1WAVEFORMAT structure, and the driver can use this information to set the appropriate bits in the frame header when it encodes. For decoding, a driver can check these fields to determine whether it is capable of decoding the stream.

b) Encoding

A driver which encodes an MPEG audio stream should read the header fields in the MPEG1WAVEFORMAT structure and set the corresponding bits in the MPEG frame header. If there is any other information which a driver requires, it must get this information either from a configuration dialog box, or through a driver callback function. For more information, see *Section A1.5*.

If a pre–encoded MPEG audio stream is wrapped with a RIFF header, it is the responsibility of the application to parse the bit stream and set the fields in the MPEG1WAVEFORMAT structure. If the sampling frequency or the bitrate index is not constant throughout the data stream, the driver should set the corresponding MPEG1WAVEFORMAT fields (<nSamplesPerSec> and <dwHeadBitrate>) to zero, as described above. If the

stream contains frames of more than one layer, it should set the flags in <fwHeadLayer> for all layers which are present in the stream. Since fields such as <fwHeadFlags> can vary from frame to frame, caution must be used in setting and testing these flags; in general, an application should not rely on them to be valid for every frame. When setting these flags, adhere to the following guidelines:

ACM_MPEG_COPYRIGHT	should be set if any of the frames in the stream have the copyright bit set.
ACM_MPEG_PROTECTIONBIT	should be set if any of the frames in the stream have the protection bit set.
ACM_MPEG_ORIGINALHOME	should be set if any of the frames in the stream have the original/home bit set. This bit may be cleared if a copy of the stream is made.
ACM_MPEG_PRIVATEBIT	should be set if any of the frames in the stream have the private bit set.
ACM_MPEG_ID_MPEG1	should be set if any of the frames in the stream have the ID bit set. For MPEG1 streams, the ID bit should always be set; however, future extensions of MPEG (such as the MPEG2 multi–channel format) may have the ID bit cleared.

If the MPEG audio stream was taken from a system–layer MPEG stream, or if the stream is intended to be integrated into the system layer, then the presentation time stamp (PTS) fields may be used. The PTS is a field in the MPEG system layer which is used for synchronization of the various fields. The MPEG PTS field is 33 bits, and therefore the RIFF WAVE format header stores the value in two fields: <dwPTSLow> contains the 32 LSBs of the PTS, and <dwPTSHigh> contains the MSB. These two fields may be taken together as a 64–bit integer; optionally, the <dwPTSHigh> field may be tested as a flag to determine whether the MSB is set or cleared. When extracting an audio stream from a system layer, a driver should set the PTS fields to the PTS of the first frame of the audio data. This may later be used to re–integrate the stream into the system layer. The PTS fields should not be used for any other purpose. If the audio stream is not associated with the MPEG system layer, then the PTS fields should be set to zero.

c) Decoding

A driver may test the fields in the MPEG1WAVEFORMAT structure to determine whether it is capable of decoding the stream. However, the driver must be aware that some fields, such as the <fwHeadFlags> field, may not be consistent for every frame in the bit stream. A driver should never use the fields of the MPEG1WAVEFORMAT structure to perform the actual decoding. The decoding parameters should be taken entirely from the MPEG data stream.

A driver may check the <nSamplesPerSec> field to determine whether it supports the sampling frequency specified. If the MPEG stream contains data with a variable sampling rate, then the <nSamplesPerSec> field will be set to zero. If the driver cannot handle this type of data stream, then it should not attempt to decode the data, but should fail immediately.

A1.5. Ancillary data

The audio data in an MPEG audio frame may not fill the entire frame. Any remaining data is called ancillary data. This data may have any format desired, and may be used to pass additional information of any kind. If a driver wishes to support the ancillary data, it must have a facility for passing the data to and from the calling application. The driver may use a callback function for this purpose. Basically, the driver may call a specified callback function whenever it has ancillary data to pass to the application (i.e. on decode) or whenever it requires more ancillary data (on encode).

Drivers should be aware that not all applications will want to process the ancillary data. Therefore, a driver should only provide this service when explicitly requested by the application. The driver may define a custom message which enables and disables the callback facility. Separate messages could be defined for the encoding and decoding operations for more flexibility.

[EBU Note: More information on the ancillary data is contained in the <mpeg_audio_extension chunk> which should be used for MPEG files conforming to the Broadcast Wave Format. See Section 2].

Bibliography

[1] *Microsoft Resource Interchange File Format, RIFF*
Microsoft Software Developers Kit Multimedia Standards Update, rev 3.0, 15 April 1994

[2] EBU document Tech. 3285: *Specification of the Broadcast Wave Format* – Supplement 1: *MPEG audio*

[3] Publication ISO/IEC 11172–3 (1993–08)
Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s –
Part 3: *Audio*

[4] Publication ISO/IEC 13818–3 (1995–05)
Information technology – Generic coding of moving pictures and associated audio information –
Part 3: *Audio*

See also:

EBU Technical Standard N22–1997: *The Broadcast Wave Format – A format for audio files in broadcasting*

EBU Technical Recommendation R85–1997: *Use of the Broadcast wave Format for the exchange of audio data files*

Editeur responsable : P. Laven, Ancienne Route 17A, CH–1218 Grand–Saconnex/Genève, Suisse.