INTERNATIONAL STANDARD

ISO/IEC 13818-7

First edition
1997-12-01

Information technology — Generic coding of moving pictures and associated audio information —

Part 7: Advanced Audio Coding (AAC)

Technologies de l'information — Codage générique des images animées et du son associé —

Partie 7: Codage du son avancé (AAC)

Reference number
ISO/IEC 13818-7:1997(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 13818-7 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

ISO/IEC 13818 consists of the following parts, under the general title *Information technology — Generic coding of moving pictures and associated audio information*:

— *Part 1: Systems*

— *Part 2: Video*

— *Part 3: Audio*

— *Part 4: Compliance testing*

— *Part 5: Software simulation*

— *Part 6: Extensions for DSM-CC*

— *Part 7: Advanced Audio Coding (AAC)*

— *Part 9: Extension for real time interface for systems decoders*

— *Part 10: Conformance extensions for DSM-CC*

Annex A forms an integral part of this part of ISO/IEC 13818. Annexes B to F are for information only.

Introduction

The standardization body ISO/IEC JTC 1/SC 29/WG 11, also known as the Moving Pictures Expert Group (MPEG), was established in 1988 to specify digital video and audio coding schemes at low data rates. MPEG completed its first phase of audio specifications (MPEG-1) in November 1992, ISO/IEC 11172-3 [1]. In its second phase of development, the MPEG Audio subgroup defined a multichannel extension to MPEG-1 audio that is backwards compatible with existing MPEG-1 systems (MPEG-2 BC) and defined an audio coding standard at lower sampling frequencies than MPEG-1, ISO/IEC 13818-3 [2].

Information technology — Generic coding of moving pictures and associated audio information —

Part 7:
Advanced Audio Coding (AAC)

1 Scope

This part of ISO/IEC 13818 describes the MPEG-2 audio non-backwards compatible standard called MPEG-2 Advanced Audio Coding, AAC [3], a higher quality multichannel standard than achievable while requiring MPEG-1 backwards compatibility. This MPEG-2 AAC audio standard allows for ITU-R 'indistinguishable' quality according to [4] at data rates of 320 kb/s for five full-bandwidth channel audio signals.

The AAC decoding process makes use of a number of required tools and a number of optional tools. Table 1 lists the tools and their status as required or optional. Required tools are mandatory in any possible profile. Optional tools may not be required in some profiles.

Table 1

Tool Name	Required / Optional
Bitstream Formatter	Required
Noiseless Decoding	Required
Inverse Quantizer	Required
Scalefactors	Required
M/S	Optional
Prediction	Optional
Intensity/Coupling	Optional
TNS	Optional
Filterbank	Required
Gain Control	Optional

MPEG-2 AAC Tools Overview

The basic structure of the MPEG-2 AAC system is shown in Figures 1.1 and 1.2. As is shown in Table 1, there are both required and optional tools in the decoder, Figure 1.2. The data flow in this diagram is from left to right, top to bottom. The functions of the decoder are to find the description of the quantized audio spectra in the bitstream, decode the quantized values and other reconstruction information, reconstruct the quantized spectra, process the reconstructed spectra through whatever tools are active in the bitstream in order to arrive at the actual signal spectra as described by the input bitstream, and finally convert the frequency domain spectra to the time domain, with or without an optional gain control tool. Following the initial reconstruction and scaling of the spectrum reconstruction, there are many optional tools that modify one or more of the spectra in order to provide more efficient coding. For each of the optional tools that operate in the spectral domain, the option to "pass through" is retained, and in all cases where a spectral operation is omitted, the spectra at its input are passed directly through the tool without modification.

The input to the bitstream demultiplexer tool is the MPEG-2 AAC bitstream. The demultiplexer separates the parts of the MPEG-AAC data stream into the parts for each tool, and provides each of the tools with the bitstream information related to that tool.
The outputs from the bitstream demultiplexer tool are:

- The sectioning information for the noiselessly coded spectra
- The noiselessly coded spectra
- The M/S decision information (optional)
- The predictor state information (optional)
- The intensity stereo control information and coupling channel control information (both optional)
- The temporal noise shaping (TNS) information (optional)
- The filterbank control information
- The gain control information (optional)

The noiseless decoding tool takes information from the bitstream demultiplexer, parses that information, decodes the Huffman coded data, and reconstructs the quantized spectra and the Huffman and DPCM coded scalefactors.

The inputs to the noiseless decoding tool are:

- The sectioning information for the noiselessly coded spectra
- The noiselessly coded spectra

The outputs of the Noiseless Decoding tool are:

- The decoded integer representation of the scalefactors:
- The quantized values for the spectra

The inverse quantizer tool takes the quantized values for the spectra, and converts the integer values to the non-scaled, reconstructed spectra. This quantizer is a non-uniform quantizer.
The input to the Inverse Quantizer tool is:

- The quantized values for the spectra

The output of the inverse quantizer tool is:

- The un-scaled, inversely quantized spectra

The scalefactor tool converts the integer representation of the scalefactors to the actual values, and multiplies the un-scaled inversely quantized spectra by the relevant scalefactors.
The inputs to the scalefactors tool are:

- The decoded integer representation of the scalefactors
- The un-scaled, inversely quantized spectra

The output from the scalefactors tool is:

- The scaled, inversely quantized spectra

The M/S tool converts spectra pairs from Mid/Side to Left/Right under control of the M/S decision information in order to improve coding efficiency.
The inputs to the M/S tool are:

- The M/S decision information
- The scaled, inversely quantized spectra related to pairs of channels

The output from the M/S tool is:

- The scaled, inversely quantized spectra related to pairs of channels, after M/S decoding

Note: The scaled, inversely quantized spectra of individually coded channels are not processed by the M/S block, rather they are passed directly through the block without modification. If the M/S block is not active, all spectra are passed through this block unmodified.

The prediction tool reverses the prediction process carried out at the encoder. This prediction process re-inserts the redundancy that was extracted by the prediction tool at the encoder, under the control of the predictor state information. This tool is implemented as a second order backward adaptive predictor. The inputs to the prediction tool are:

- The predictor state information
- The scaled, inversely quantized spectra

The output from the prediction tool is:

- The scaled, inversely quantized spectra, after prediction is applied.

Note: If the prediction is disabled, the scaled, inversely quantized spectra are passed directly through the block without modification.

The intensity stereo / coupling tool implements intensity stereo decoding on pairs of spectra. In addition, it adds the relevant data from a dependently switched coupling channel to the spectra at this point, as directed by the coupling control information.
The inputs to the intensity stereo / coupling tool are:

- The inversely quantized spectra
- The intensity stereo control information and coupling control information

The output from the intensity stereo / coupling tool is:

- The inversely quantized spectra after intensity and coupling channel decoding.

Note: If either part of this block is disabled, the scaled, inversely quantized spectra are passed directly through the block without modification. The intensity stereo tool and M/S tools are arranged so that the operation of M/S and Intensity stereo are mutually exclusive on any given scalefactor band and group of one pair of spectra.

The temporal noise shaping (TNS) tool implements a control of the fine time structure of the coding noise. In the encoder, the TNS process has flattened the temporal envelope of the signal to which it has been applied. In the decoder, the inverse process is used to restore the actual temporal envelope(s), under control of the TNS information. This is done by applying a filtering process to parts of the spectral data.
The inputs to the TNS tool are:

- The inversely quantized spectra
- The TNS information

The output from the TNS block is:

- The inversely quantized spectra

Note: If this block is disabled, the inversely quantized spectra are passed through without modification.

The filterbank tool applies the inverse of the frequency mapping that was carried out in the encoder, as indicated by the filterbank control information and the presence or absence of gain control information. An inverse modified discrete cosine transform (IMDCT) is used for the filterbank tool. If the gain control tool is not used, the IMDCT input consists of either 1024 or 128 spectral coefficients, depending of the value of window_sequence (see 6.3, Table 6.11). If the gain control tool is used, the filterbank tool is configured to use four sets of either 256 or 32 coefficients, depending of the value of window_sequence.
The inputs to the filterbank tool are:

- The inversely quantized spectra
- The filterbank control information

The output(s) from the filterbank tool is (are):

- The time domain reconstructed audio signal(s).

When present, the gain control tool applies a separate time domain gain control to each of 4 frequency bands that have been created by the gain control PQF filterbank in the encoder. Then, it assembles the 4 frequency bands and reconstructs the time waveform through the gain control tool's filterbank.
The inputs to the gain control tool are:

- The time domain reconstructed audio signal(s)
- The gain control information

The output(s) from the gain control tool is (are):

- The time domain reconstructed audio signal(s)

If the gain control tool is not active, the time domain reconstructed audio signal(s) are passed directly from the filterbank tool to the output of the decoder. This tool is used for the scaleable sampling rate (SSR) profile only.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 13818. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 13818 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

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

[2] ISO/IEC 13818-3:1997, *Information technology — Generic coding of moving pictures and associated audio information — Part 3: Audio.*

[3] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, Y. Oikawa, "ISO/IEC MPEG-2 Advanced Audio Coding", Journal of the Audio Engineering Society, Vol. 45, no. 10, pp. 789-814, October 1997.

[4] ITU-R Document TG10-2/3- E only, *Basic Audio Quality Requirements for Digital Audio Bit-Rate Reduction Systems for Broadcast Emission and Primary Distribution*, 28 October 1991.

[5] F. J. Harris, *On the Use of Windows For Harmonic Analysis of the Discrete Fourier Transform*, Proc. of the IEEE, Vol. 66, pp. 51- 83, January 1975.

[6] ISO/IEC 13818-1:1996, *Information technology — Generic coding of moving pictures and associated audio information: Systems.*

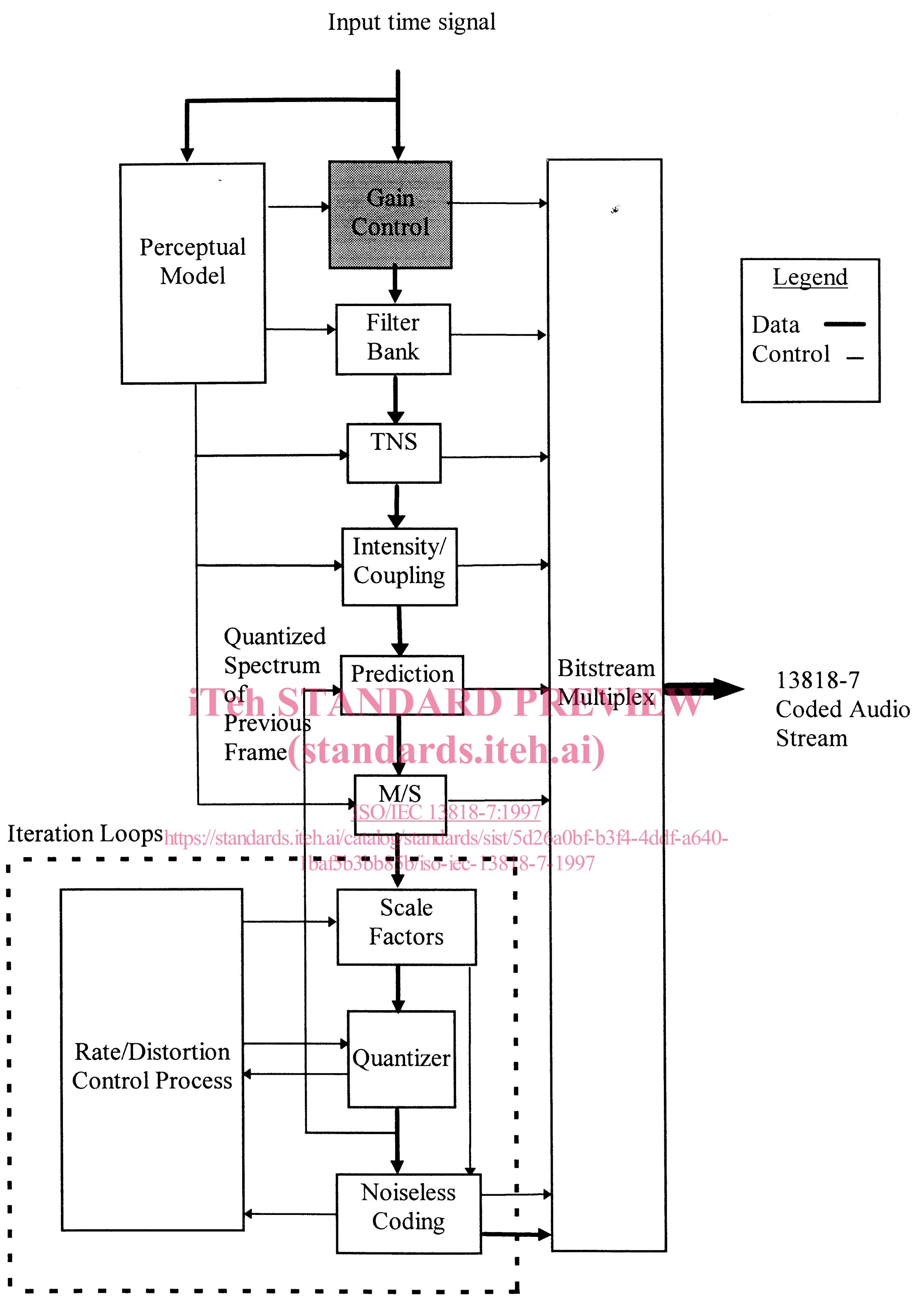

Figure – 1.1 MPEG-2 AAC Encoder Block Diagram

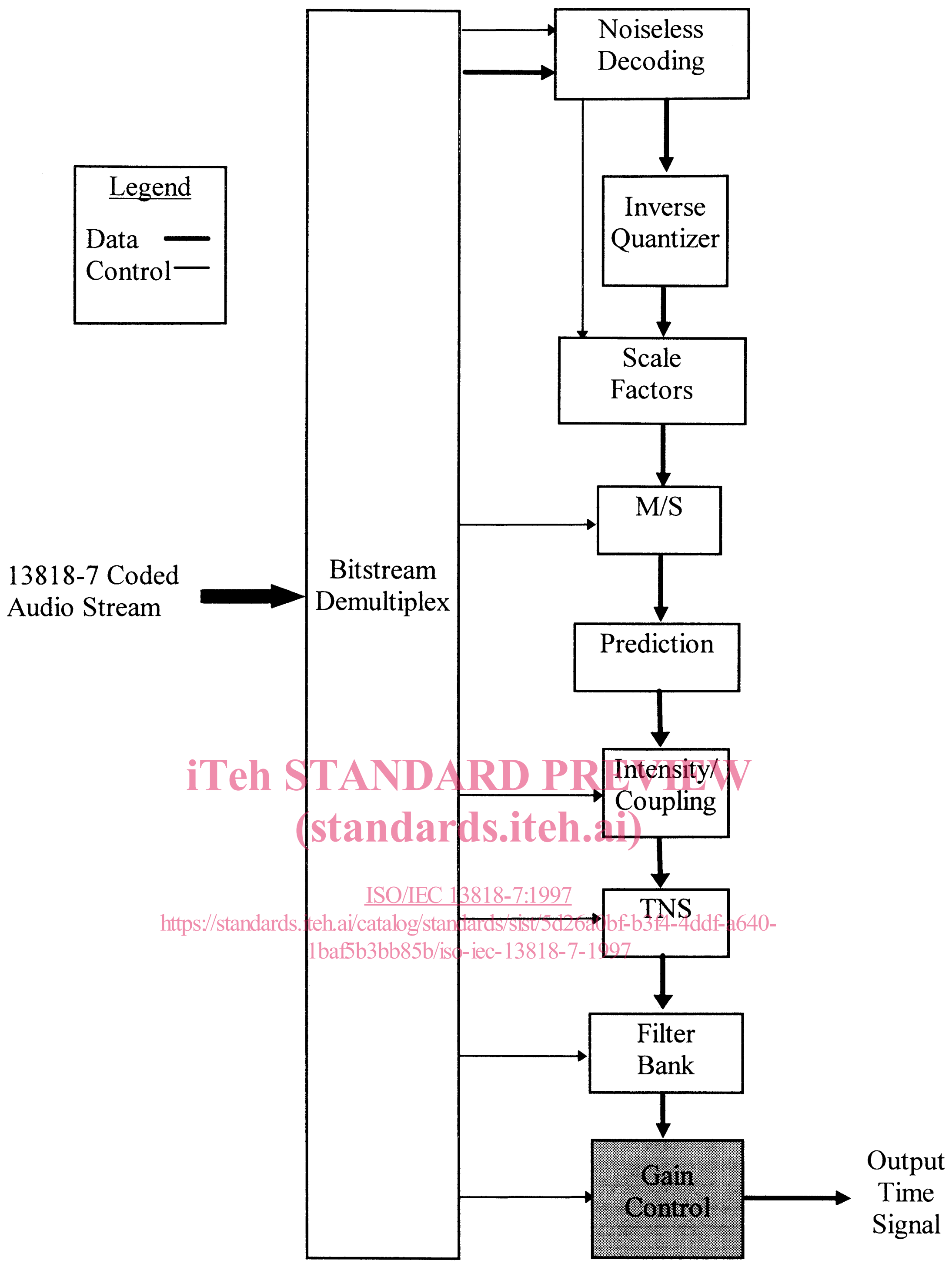

Figure 1.2 – MPEG-2 AAC Decoder Block Diagram

3 Definitions

For the purposes of this part of ISO/IEC 13818, the following definitions apply.

3.1. alias: Mirrored signal component resulting from sampling.
3.2. analysis filterbank: Filterbank in the encoder that transforms a broadband PCM audio signal into a set of spectral coefficients.
3.3. ancillary data: Part of the bitstream that might be used for transmission of ancillary data.
3.4. audio buffer: A buffer in the system target decoder (see ISO/IEC 13818-1) for storage of compressed audio data.
3.5. Bark: The Bark is the standard unit corresponding to one critical band width of human hearing.
3.6. backward compatibility: A newer coding standard is backward compatible with an older coding standard if decoders designed to operate with the older coding standard are able to continue to operate by decoding all or part of a bitstream produced according to the newer coding standard.

3.7. bitrate: The rate at which the compressed bitstream is delivered to the input of a decoder.
3.8. bitstream; stream: An ordered series of bits that forms the coded representation of the data.
3.9. bitstream verifier: A process by which it is possible to test and verify that all the requirements specified in ISO/IEC 13818-7 are met by the bitstream.
3.10. block companding: Normalising of the digital representation of an audio signal within a certain time period.
3.11. byte aligned: A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits from either the first bit in the stream for the Audio_Data_Interchange_Format (see 6.1) or the first bit in the syncword for the Audio_Data_Transport_Stream Format (see 6.2).
3.12. byte: Sequence of 8-bits.
3.13. centre channel: An audio presentation channel used to stabilise the central component of the frontal stereo image.
3.14. channel: A sequence of data representing an audio signal intended to be reproduced at one listening position.
3.15. coded audio bitstream: A coded representation of an audio signal.
3.16. coded representation: A data element as represented in its encoded form.
3.17. compression: Reduction in the number of bits used to represent an item of data.
3.18. constant bitrate: Operation where the bitrate is constant from start to finish of the coded bitstream.
3.19. CRC: The Cyclic Redundancy Check to verify the correctness of data.
3.20. critical band: This unit of bandwidth represents the standard unit of bandwidth expressed in human auditory terms, corresponding to a fixed length on the human cochlea. It is approximately equal to 100 Hz at low frequencies and 1/3 octave at higher frequencies, above approximately 700 Hz.
3.21. data element: An item of data as represented before encoding and after decoding.
3.22. de-emphasis: Filtering applied to an audio signal after storage or transmission to undo a linear distortion due to emphasis.
3.23. decoded stream: The decoded reconstruction of a compressed bitstream.
3.24. decoder: An embodiment of a decoding process.
3.25. decoding (process): The process defined in This part of ISO/IEC 13818 that reads an input coded bitstream and outputs decoded audio samples.
3.26. digital storage media; DSM: A digital storage or transmission device or system.
3.27. discrete cosine transform; DCT: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation.
3.28. downmix: A matrixing of n channels to obtain less than n channels.
3.29. editing: The process by which one or more coded bitstreams are manipulated to produce a new coded bitstream. Conforming edited bitstreams must meet the requirements defined in this part of ISO/IEC 13818.
3.30. emphasis: Filtering applied to an audio signal before storage or transmission to improve the signal-to-noise ratio at high frequencies.
3.31. encoder: An embodiment of an encoding process.
3.32. encoding (process): A process, not specified in ISO/IEC 13818, that reads a stream of input audio samples and produces a valid coded bitstream as defined in this part of ISO/IEC 13818.
3.33. entropy coding: Variable length lossless coding of the digital representation of a signal to reduce statistical redundancy.
3.34. FFT: Fast Fourier Transformation. A fast algorithm for performing a discrete Fourier transform (an orthogonal transform).
3.35. filterbank: A set of band-pass filters covering the entire audio frequency range.
3.36. flag: A variable which can take one of only the two values defined in this specification.
3.37. forward compatibility: A newer coding standard is forward compatible with an older coding standard if decoders designed to operate with the newer coding standard are able to decode bitstreams of the older coding standard.
3.38. Fs: Sampling frequency.
3.39. Hann window: A time function applied sample-by-sample to a block of audio samples before Fourier transformation.
3.40. Huffman coding: A specific method for entropy coding.
3.41. hybrid filterbank: A serial combination of subband filterbank and MDCT.
3.42. IDCT: Inverse Discrete Cosine Transform.
3.43. IMDCT: Inverse Modified Discrete Cosine Transform.
3.44. intensity stereo: A method of exploiting stereo irrelevance or redundancy in stereophonic audio programmes based on retaining at high frequencies only the energy envelope of the right and left channels.
3.45. joint stereo coding: Any method that exploits stereophonic irrelevance or stereophonic redundancy.
3.46. joint stereo mode: A mode of the audio coding algorithm using joint stereo coding.
3.47. low frequency enhancement (LFE) channel: A limited bandwidth channel for low frequency audio effects in a multichannel system.
3.48. main audio channels: All single_channel_elements (see 8.2.1) or channel_pair_elements (see 8.2.1) in one program.
3.49. mapping: Conversion of an audio signal from time to frequency domain by subband filtering and/or by MDCT.
3.50. masking: A property of the human auditory system by which an audio signal cannot be perceived in the presence of another audio signal.
3.51. masking threshold: A function in frequency and time below which an audio signal cannot be perceived by the human auditory system.
3.52. modified discrete cosine transform (MDCT): A transform which has the property of time domain aliasing cancellation. An analytical espression for the MDCT can be found in B 2.3.1.2.

3.53. M/S stereo: A method of removing imaging artefacts as well as exploiting stereo irrelevance or redundancy in stereophonic audio programmes based on coding the sum and difference signal instead of the left and right channels.
3.54. multichannel: A combination of audio channels used to create a spatial sound field.
3.55. multilingual: A presentation of dialogue in more than one language.
3.56. non-tonal component: A noise-like component of an audio signal.
3.57. Nyquist sampling: Sampling at or above twice the maximum bandwidth of a signal.
3.58. padding: A method to adjust the average length of an audio frame in time to the duration of the corresponding PCM samples, by conditionally adding a slot to the audio frame.
3.59. parameter: A variable within the syntax of this specification which may take one of a range of values. A variable which can take one of only two values is a flag or indicator and not a parameter.
3.60. parser: Functional stage of a decoder which extracts from a coded bitstream a series of bits representing coded elements.
3.61. polyphase filterbank: A set of equal bandwidth filters with special phase interrelationships, allowing for an efficient implementation of the filterbank.
3.62. prediction error: The difference between the actual value of a sample or data element and its predictor.
3.63. prediction: The use of a predictor to provide an estimate of the sample value or data element currently being decoded.
3.64. predictor: A linear combination of previously decoded sample values or data elements.
3.65. presentation channel: An audio channel at the output of the decoder.
3.66. program: A set of main audio channels, coupling_channel_elements (see 8.2.1), lfe_channel_elements (see 8.2.1), and associated data streams intended to be decoded and played back simultaneously A program may be defined by default (see 8.5.1) or specifically by a program_configuration_element (see 8.2.1). A given single_channel_element (see 8.2.1) , channel_pair_element (see 8.2.1), coupling_channel_element, lfe_channel_element or data channel may accompany one or more programs in any given bitstream.
3.67. psychoacoustic model: A mathematical model of the masking behaviour of the human auditory system.
3.68. random access: The process of beginning to read and decode the coded bitstream at an arbitrary point.
3.69. reserved: The term "reserved" when used in the clauses defining the coded bitstream indicates that the value may be used in the future for ISO/IEC defined extensions.
3.70. Sampling Frequency (Fs): Defines the rate in Hertz which is used to digitise an audio signal during the sampling process.
3.71. scalefactor: Factor by which a set of values is scaled before quantization.
3.72. scalefactor band: A set of spectral coefficients which are scaled by one scalefactor.
3.73. scalefactor index: A numerical code for a scalefactor.
3.74. side information: Information in the bitstream necessary for controlling the decoder.
3.75.spectral coefficients: Discrete frequency domain data output from the analysis filterbank.
3.76. spreading function: A function that describes the frequency spread of masking effects.
3.77. stereo-irrelevant: A portion of a stereophonic audio signal which does not contribute to spatial perception.
3.78. stuffing (bits); stuffing (bytes): Code-words that may be inserted at particular locations in the coded bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream which would otherwise be lower than the desired bitrate.
3.79. surround channel: An audio presentation channel added to the front channels (L and R or L, R, and C) to enhance the spatial perception.
3.80. syncword: A 12-bit code embedded in the audio bitstream that identifies the start of a adts_frame() (see 6.2, Table 6.4).
3.81. synthesis filterbank: Filterbank in the decoder that reconstructs a PCM audio signal from subband samples.
3.82. tonal component: A sinusoid-like component of an audio signal.
3.83. variable bitrate: Operation where the bitrate varies with time during the decoding of a coded bitstream.
3.84. variable length coding: A reversible procedure for coding that assigns shorter code-words to frequent symbols and longer code-words to less frequent symbols.
3.85. variable length code (VLC): A code word assigned by variable length encoder (See variable length coding)
3.86. variable length decoder: A procedure to obtain the symbols encoded with a variable length coding technique.
3.87. variable length encoder: A procedure to assign variable length codewords to symbols.

4 Symbols and abbreviations

The mathematical operators used to describe this Recommendation | International Standard are similar to those used in the C programming language. However, integer division with truncation and rounding are specifically defined. The bitwise operators are defined assuming twos-complement representation of integers. Numbering and counting loops generally begin from zero.

4.1 Arithmetic operators

+	Addition.
−	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment.
− −	Decrement.
*	Multiplication.
^	Power.

/	Integer division with truncation of the result toward zero. For example, 7/4 and −7/−4 are truncated to 1 and −7/4 and 7/−4 are truncated to −1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example 3//2 is rounded to 2, and −3//2 is rounded to −2.
DIV	Integer division with truncation of the result towards −∞.
| |	Absolute value. $\|x\| = x$ when $x > 0$; $\|x\| = 0$ when $x == 0$; $\|x\| = -x$ when $x < 0$
%	Modulus operator. Defined only for positive numbers.
Sign( )	Sign. $Sign(x) = 1$ when $x > 0$; $Sign(x) = 0$ when $x == 0$; $Sign(x) = -1$ when $x < 0$
INT ( )	Truncation to integer operator. Returns the integer part of the real-valued argument.
NINT ( )	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from zero.
sin	Sine.
cos	Cosine.
exp	Exponential.
√	Square root.
$\log_{10}$	Logarithm to base ten.
$\log_e$	Logarithm to base e.
$\log_2$	Logarithm to base 2.

4.2 Logical operators

||	Logical OR.
&&	Logical AND.
!	Logical NOT

4.3 Relational operators

>	Greater than.
>=	Greater than or equal to.
<	Less than.
<=	Less than or equal to.
==	Equal to.
!=	Not equal to.
max [,...,]	the maximum value in the argument list.
min [,...,]	the minimum value in the argument list.

4.4 Bitwise operators

A twos complement number representation is assumed where the bitwise operators are used.

&	AND
|	OR
>>	Shift right with sign extension.
<<	Shift left with zero fill.

4.5 Assignment

=	Assignment operator.

4.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit stream.

bslbf	Bit string, left bit first, where "left" is the order in which bit strings are written in ISO/IEC 11172. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.
L, C, R, LS, RS	Left, center, right, left surround and right surround audio signals
rpchof	Remainder polynomial coefficients, highest order first. (Audio)
uimsbf	Unsigned integer, most significant bit first.
vlclbf	Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written.
window	Number of the actual time slot in case of block_type==2, 0 <= window <= 2. (Audio)

The byte order of multi-byte words is most significant byte first.

4.7 Constants

π	3,14159265358...
e	2,71828182845...

5 Method of describing bit stream syntax

The bit stream retrieved by the decoder is described in clause 6 (Syntax). Each data item in the bit stream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.
The action caused by a decoded data element in a bit stream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and the definition of the state variables used in their decoding are described in 8.3. The following constructs are used to express the conditions when data elements are present, and are in normal type:
Note this syntax uses the 'C'-code convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true.

```
while ( condition ) {
  data_element
  . . .
}
```

If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

```
do {
  data_element
  . . .
} while ( condition )
```

The data element always occurs at least once. The data element is repeated until the condition is not true.

```
if ( condition) {
  data_element
  . . .
}
```

If the condition is true, then the first group of data elements occurs next in the data stream

```
else {
  data_element
  . . .
}
```

If the condition is not true, then the second group of data elements occurs next in the data stream.

```
for (expr1; expr2; expr3) {
  data_element
  . . .
}
```

Expr1 is an expression specifying the initialisation of the loop. Normally it specifies the initial state of the counter. Expr2 is a condition specifying a test made before each iteration of the loop. The loop terminates when the condition is not true. Expr3 is an expression that is performed at the end of each iteration of the loop, normally it increments a counter.

Note that the most common usage of this construct is as follows:

```
for ( i = 0; i < n; i++) {
  data_element
  . . .
}
```

The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to one for the second occurrence, and so forth.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the {} may be omitted when only one data element follows.

data_element [ ]	data_element [ ] is an array of data. The number of data elements is indicated by the context.
data_element [n]	data_element [n] is the n+1th element of an array of data.
data_element [m][n]	data_element [m][n] is the m+1,n+1 th element of a two-dimensional array of data.
data_element [l][m][n]	data_element [l][m][n] is the l+1,m+1,n+1 th element of a three-dimensional array of data.
data_element [m..n]	data_element [m..n]is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that clause x.x.x implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bit stream. Actual decoders must include a means to look for start codes in order to begin decoding correctly.
Definition of bytealigned function
The function bytealigned ( ) returns 1 if the current position is on a byte boundary, that is the next bit in the bit stream is the first bit in a byte. Otherwise it returns 0.
Definition of nextbits function
The function nextbits ( ) permits comparison of a bit string with the next bits to be decoded in the bit stream.

6 Syntax

6.1 Audio_Data_Interchange_Format, ADIF

Table 6.1 – Syntax of adif_sequence()

Syntax	No. of bits	Mnemonic
adif_sequence()		
{		
adif_header()		
byte_alignment()		
raw_data_stream()		
}		

Table 6.2 – Syntax of adif_header()

Syntax	No. of bits	Mnemonic
adif_header()		
{		
adif_id	**32**	**bslbf**
copyright_id_present	**1**	**bslbf**
if(copyright_id_present)		
copyright_id	**72**	**bslbf**
original_copy	**1**	**bslbf**
home	**1**	**bslbf**
bitstream_type	**1**	**bslbf**
bitrate	**23**	**uimsbf**
num_program_config_elements	**4**	**bslbf**
for (i = 0; i < num_program_config_elements + 1; i++) {		
if(bitstream_type == ‘0’)		
adif_buffer_fullness	**20**	**uimsbf**
program_config_element()		
}		
}		

6.2 Audio_Data_Transport_Stream frame, ADTS

Table 6.3 – Syntax of adts_sequence()

Syntax	No. of bits	Mnemonic
adts_sequence()		
{		
while (nextbits()==syncword) {		
adts_frame()		
}		
}		

Table 6.4 – Syntax of adts_frame()

Syntax	No. of bits	Mnemonic
adts_frame()		
{		
adts_fixed_header()		
adts_variable_header()		
adts_error_check()		
byte_alignment()		
for(i=0; i<number_of_raw_data_blocks_in_frame+1; i++) {		
raw_data_block()		
byte_alignment()		
}		
}		

6.2.1 Fixed header of ADTS

Table 6.5 – Syntax of adts_fixed_header()

Syntax	No. of bits	Mnemonic
adts_fixed_header()		
{		
syncword	12	**bslbf**
ID	1	**bslbf**
layer	2	**uimsbf**
protection_absent	1	**bslbf**
profile	2	**uimsbf**
sampling_frequency_index	4	**uimsbf**
private_bit	1	**bslbf**
channel_configuration	3	**uimsbf**
original/copy	1	**bslbf**
home	1	**bslbf**
}		

6.2.2 Variable header of ADTS

Table 6.6 – Syntax of adts_variable_header()

Syntax	No. of bits	Mnemonic
adts_variable_header()		
{		
copyright_identification_bit	1	**bslbf**
copyright_identification_start	1	**bslbf**
frame_length	13	**bslbf**
adts_buffer_fullness	11	**bslbf**
number_of_raw_data_blocks_in_frame	2	**uimsfb**
}		

6.2.3 Error detection

Table 6.7 – Syntax of adts_error_check()

Syntax	No. of bits	Mnemonic
adts_error_check()		
{		
if (protection_absent == '0')		
crc_check	16	**rpchof**
}		

6.3 Raw data

This is the raw_data_stream. It can be decoded directly or put into the Audio_Data_Transport stream using a variable rate header and a buffer fullness measure, with all the data of one frame contained between two occurences of the header.

Table 6.7a – Syntax of raw_data_stream()

Syntax	No. of bits	Mnemonic
raw_data_stream()		
{		
while (data_available()) {		
raw_data_block()		
byte_alignment()		
}		
}		

Table 6.8 – Syntax of raw_data_block()

Syntax	No. of bits	Mnemonic
raw_data_block()		
{		
while((id = **id_syn_ele**) != ID_END){	3	**uimsbf**
switch (id) {		
case ID_SCE: single_channel_element()		
break;		
case ID_CPE: channel_pair_element()		