ISO/JTC 1/SC 29 N2203

Date: 1998-05-15

ISO/IEC FCD 14496-3 Subpart 1

ISO/JTC 1/SC 29/WG11

Secretariat:

Information Technology - Very Low Bitrate Audio-Visual Coding

Part 3: Audio

Subpart 1: Main Document

Authors: **B. Grill, B. Edler, R. Funken, M. Hahn, K. Iijima, N. Iwakami, Y. Lee, T. Moryia, J. Ostermann, S. Nakajima, M. Nishiguchi, T. Nomura, W. Oomen, H. Purnhagen, E. Scheirer, N. Tanaka, A.P. Tan, R. Taori, MPEG-2 AAC Authors (IS -13818-7)**

Document type: International standard
Document:sub-type if applicable
Document:stage (20) Préparation
Document:language E

C:\ISOST18\BASICEN.DOT ISOSTD Basic Version 1.8 1996-10-30

Organisation of the document

MPEG-4 Audio comprises 6 subparts:

Subpart 1: Main

Subpart 2: Parametric Coding

Subpart 3: CELP Coding

Subpart 4: Time/Frequency Coding

Subpart 5: Structured Audio

Subpart 6: Text to Speech

For reasons of managability of large documents, the CD is divided in several files:

1. w2203: The master file, containing general introduction and description of tools for other functionalities (this file)
2. w2203par: Contains the description of Subpart 2.
3. w2203cel: Contains the description of Subpart 3.
4. w2203tft: Contains the tool descriptions of Subpart 4.
5. w2203tfs: Contains the syntax part of Subpart 4.
6. w2203tfa: Contains the informative part of Subpart 4.
7. w2203sa: Contains the description of Subpart 5.
8. w2203tts: Contains the description of Subpart 6.

For each of the five coding schemes as well as for the tools for other functionalities, there is a section containing a normative part (description of the syntax and the decoding process) and an informative annex with encoder and interface description. In addition there are two files containing very large tables. These are:

w2203tvq Twin-VQ vector quantizer tables (Subpart 4).

w2203pvq Parametric coder vector quantizer tables (Subpart 2).

1 Introduction

MPEG-4 audio coding integrates the worlds of speech and high quality audio coding as well as the worlds of sound synthesis and the representation of natural audio. The sound synthetis part is comprised of tools for the realisation of symbolically defined music and speech. This includes MIDI and Text-to-Speech systems. Furthermore, tools for effects processing and 3-D localisation of sound are included, allowing the creation of artificial sound environments using artificial and natural sources.

Synthetic audio is described by first defining a set of 'instrument' modules that can create and process audio signals under the control of a script or score file. An instrument is a small network of signal processing primitives that can emulate the effects of a natural acoustic instrument. A script or score is a time-sequenced set of commands that invokes various instruments at specific times to contribute their output to an overall music performance. Other instruments, serving the function of effects processors (reverberators, spatialisers, mixers), can be similarly invoked to receive and process the outputs of the performing instruments. These actions can not only realise a music composition but can also organise any other kind of audio, such as speech, sound effects and general ambience. Likewise, the audio sources can themselves be natural sounds, perhaps emanating from an audio channel decoder, thus enabling synthetic and natural sources to be merged with complete timing accuracy.

TTS is becoming a rather common interface and plays an important role in various multi-media application areas. For instance, by using TTS functionality, multi-media contents with narration can be easily composed without recording natural speech sound. Moreover, TTS with FA/AP/MP functionality would possibly make the contents much richer. In MPEG-4 activity, common interfaces for TTS and TTS for FA/AP/MP are proposed. The proposed MPEG-4 TTS functionality; Hybrid/Multi-Level Scalable TTS, can be considered as a superset of the conventional TTS framework. This extended TTS can utilize prosodic information of natural speech in addition to input texts and can generate much higher quality of synthetic speech. The interface and its bitstream format is strongly scalable; for example, if some parameters of prosodic information are not available, it then generates the missing parameters by rule. Still the basic idea for the scalable MPEG-4 TTS interface is it can fully utilize all the provided information according to the level of user's requirements. The functionality of this extended TTS thus ranges from conventional TTS to natural speech coding and its application areas, from simple TTS to AP with TTS and MP dubbing with TTS.

MPEG-4 standardises natural audio coding at bitrates ranging from 2 kbit/s up to 64 kbit/s. The presence of the MPEG-2 AAC standard within the MPEG-4 tool set will provide for compression of general audio. For the bitrates from 2 kbit/s up to 64 kbit/s, the MPEG-4 standard normalises the bitstream syntax and decoding processes in terms of a set of tools. In order to achieve the highest audio quality within the full range of bitrates and at the same time provide the extra functionalities, three types of coder have been defined. The lowest bitrate range between about 2 and 6 kbit/s, mostly used for speech coding at 8 kHz sampling frequency, is covered by parametric coding techniques. Coding at the medium bitrates between about 6 and 24 kbit/s uses Code Excited Linear Predictive (CELP) coding techniques. In this region, two sampling rates, 8 and 16 kHz, are used to support a broader range of audio signals (other than speech). For the bitrates typically starting at about 16 kbit/s, time to frequency coding techniques are applied. The audio signals in this region typically have bandwidths starting at 8 kHz.

A number of functionalities are provided to facilitate a wide variety of applications which could range from intelligible speech to high quality multichannel audio. Examples of the functionalities are speed control, pitch change, error resilience and scalability in terms of bitrate, bandwidth, error robustness, complexity, etc. as defined below. These functionalities are applicable to the individual coding schemes (parametric, CELP and t/f) as well as across the coding schemes.

- The speed change functionality allows the change of the time scale without altering the pitch during the decoding process. This can, for example, be used to implement a "fast forward" function (data base search) or to adapt the length of an audio sequence to a given video sequence.
- The pitch change functionality allows the change of the pitch without altering the time scale during the encoding or decoding process.This can be used for example for voice alteration or Karaoke type applications.

Bitrate scalability allows a bitstream to be parsed into a bitstream of lower bitrate such that the combination can still be decoded into a meaningful signal. The bit stream parsing can occur either during transmission or in the decoder.

- Bandwidth scalability is a particular case of bitrate scalability, whereby part of a bitstream representing a part of the frequency spectrum can be discarded during transmission or decoding.
- Encoder complexity scalability allows encoders of different complexity to generate valid and meaningful bitstreams.
- Decoder complexity scalability allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used.
- Error robustness provides the ability for a decoder to avoid or conceal audible distortion caused by transmission errors.

To allow for smooth transitions between the bitrates and to allow for bitrate and bandwidth scalability, a general framework has been defined. This is illustrated in figure 1.

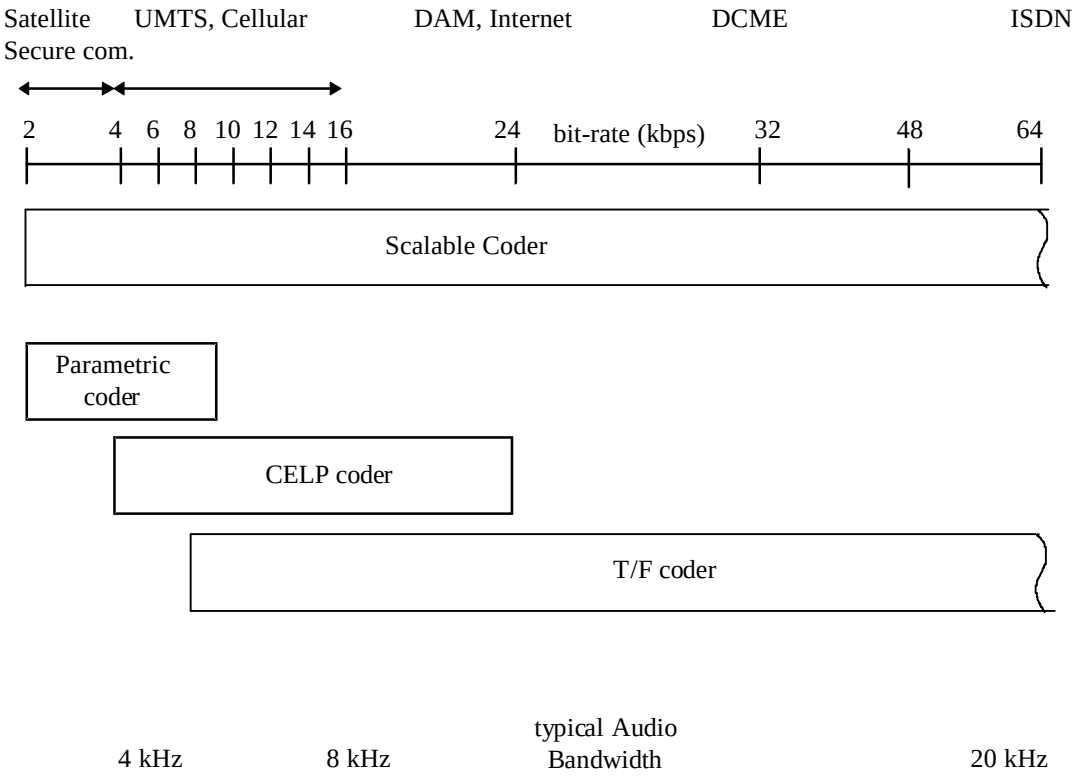

Figure 1

Starting with a coder operating at a low bitrate, by adding enhancements both the coding quality as well as the audio bandwidth can be improved. These enhancements are realised within a single coder or alternatively by combining different techniques.

Additional functionalities are realised both within individual coders, and by means of additional tools around the coders. An example of a functionality within an individual coder is pitch change within the parametric coder.

1.1 Glossary

For the purposes of this International Draft Standard, the following definitions apply. If specific to a part, this is noted in square brackets.

1. alias: Mirrored signal component resulting from sampling.

2. analysis filterbank: Filterbank in the encoder that transforms a broadband PCM audio signal into a set of spectral coefficients.

3. ancillary data: Part of the bitstream that might be used for transmission of ancillary data.

4. audio buffer: A buffer in the system target decoder (see ISO/IEC 13818-1) for storage of compressed audio data.

5. Bark: The Bark is the standard unit corresponding to one critical band width of human hearing.

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.

7. bitrate: The rate at which the compressed bitstream is delivered to the input of a decoder.

8. bitstream; stream: An ordered series of bits that forms the coded representation of the data.

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.

10. block companding: Normalising of the digital representation of an audio signal within a certain time period.

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 1.1) or the first bit in the syncword for the Audio_Data_Transport_Stream Format (see 1.2).

12. byte: Sequence of 8-bits.

13. centre channel: An audio presentation channel used to stabilise the central component of the frontal stereo image.

14. channel: A sequence of data representing an audio signal intended to be reproduced at one listening position.

15. coded audio bitstream: A coded representation of an audio signal.

16. coded representation: A data element as represented in its encoded form.

17. compression: Reduction in the number of bits used to represent an item of data.

18. constant bitrate: Operation where the bitrate is constant from start to finish of the coded bitstream.

19. CRC: The Cyclic Redundancy Check to verify the correctness of data.

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.

21. data element: An item of data as represented before encoding and after decoding.

22. de-emphasis: Filtering applied to an audio signal after storage or transmission to undo a linear distortion due to emphasis.

23. decoded stream: The decoded reconstruction of a compressed bitstream.

24. decoder: An embodiment of a decoding process.

25. decoding (process): The process defined in ISO/IEC 13818 part 7 that reads an input coded bitstream and outputs decoded audio samples.

26. digital storage media; DSM: A digital storage or transmission device or system.

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.

28. downmix: A matrixing of n channels to obtain less than n channels.

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 part 7 of ISO/IEC 13818.

30. emphasis: Filtering applied to an audio signal before storage or transmission to improve the signal-to-noise ratio at high frequencies.

31. encoder: An embodiment of an encoding process.

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 part 7 of ISO/IEC 13818.

33. entropy coding: Variable length lossless coding of the digital representation of a signal to reduce statistical redundancy.

34. FFT: Fast Fourier Transformation. A fast algorithm for performing a discrete Fourier transform (an orthogonal transform).

35. filterbank: A set of band-pass filters covering the entire audio frequency range.

36. flag: A variable which can take one of only the two values defined in this specification.

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.

38. Fs: Sampling frequency.

39. Hann window: A time function applied sample-by-sample to a block of audio samples before Fourier transformation.

40. Huffman coding: A specific method for entropy coding.

41. hybrid filterbank: A serial combination of subband filterbank and MDCT.

42. IDCT: Inverse Discrete Cosine Transform.

43. IMDCT: Inverse Modified Discrete Cosine Transform.

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.

45. joint stereo coding: Any method that exploits stereophonic irrelevance or stereophonic redundancy.

46. joint stereo mode: A mode of the audio coding algorithm using joint stereo coding.

47. low frequency enhancement (LFE) channel: A limited bandwidth channel for low frequency audio effects in a multichannel system.

48. main audio channels: All single_channel_elements (see clause 3.2.1) or channel_pair_elements (see clause 3.2.1) in one program.

49. mapping: Conversion of an audio signal from time to frequency domain by subband filtering and/or by MDCT.

50. masking: A property of the human auditory system by which an audio signal cannot be perceived in the presence of another audio signal.

51. masking threshold: A function in frequency and time below which an audio signal cannot be perceived by the human auditory system.

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 clause B 2.3.1.2.

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.

54. multichannel: A combination of audio channels used to create a spatial sound field.

55. multilingual: A presentation of dialogue in more than one language.

56. non-tonal component: A noise-like component of an audio signal.

57. Nyquist sampling: Sampling at or above twice the maximum bandwidth of a signal.

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.

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.

60. parser: Functional stage of a decoder which extracts from a coded bitstream a series of bits representing coded elements.

61. polyphase filterbank: A set of equal bandwidth filters with special phase interrelationships, allowing for an efficient implementation of the filterbank.

62. prediction error: The difference between the actual value of a sample or data element and its predictor.

63. prediction: The use of a predictor to provide an estimate of the sample value or data element currently being decoded.

64. predictor: A linear combination of previously decoded sample values or data elements.

65. presentation channel: An audio channel at the output of the decoder.

66. program: A set of main audio channels, coupling_channel_elements (see clause 3.2.1), lfe_channel_elements (see clause 3.2.1), and associated data streams intended to be decoded and played back simultaneously A program may be defined by default (see clause 3.5.1) or specifically by a program_configuration_element (see clause 3.2.1). A given single_channel_element (see clause 3.2.1) , channel_pair_element (see clause 3.2.1), coupling_channel_element, lfe_channel_element or data channel may accompany one or more programs in any given bitstream..

67. psychoacoustic model: A mathematical model of the masking behaviour of the human auditory system.

68. random access: The process of beginning to read and decode the coded bitstream at an arbitrary point.

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.

70. Sampling Frequency (Fs): Defines the rate in Hertz which is used to digitise an audio signal during the sampling process.

71. scalefactor: Factor by which a set of values is scaled before quantization.

72. scalefactor band: A set of spectral coefficients which are scaled by one scalefactor.

73. scalefactor index: A numerical code for a scalefactor.

74. side information: Information in the bitstream necessary for controlling the decoder.

75.spectral coefficients: Discrete frequency domain data output from the analysis filterbank.

76. spreading function: A function that describes the frequency spread of masking effects.

77. stereo-irrelevant: A portion of a stereophonic audio signal which does not contribute to spatial perception.

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.

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.

80. syncword: A 12-bit code embedded in the audio bitstream that identifies the start of a adts_frame() (see 1.2, Table 6.4).

81. synthesis filterbank: Filterbank in the decoder that reconstructs a PCM audio signal from subband samples.

82. tonal component: A sinusoid-like component of an audio signal.

83. variable bitrate: Operation where the bitrate varies with time during the decoding of a coded bitstream.

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.

85. variable length code (VLC): A code word assigned by variable length encoder (See variable length coding)

86. variable length decoder: A procedure to obtain the symbols encoded with a variable length coding technique.

87. variable length encoder: A procedure to assign variable length codewords to symbols.

1.2 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.

1.2.1 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.

1.2.2 Logical operators

|| Logical OR.

&& Logical AND.

! Logical NOT

1.2.3 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.

1.2.4 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.

1.2.5 Assignment

= Assignment operator.

1.2.6 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.

1.2.7 Constants

π 3,14159265358...

e 2,71828182845...

1.3 Method of describing bit stream syntax

The bit stream retrieved by the decoder is described in 1 (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 3.3.. 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.

2 Technical Overview

to be added

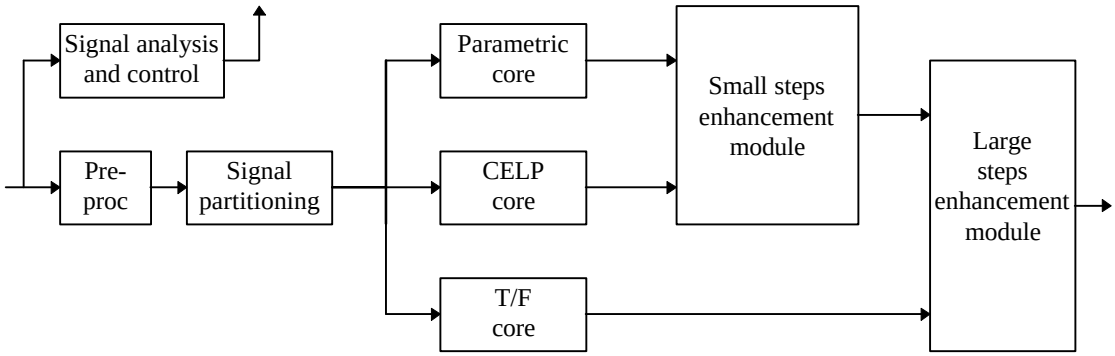

Figure 2

2.1 MPEG-4 Audio Object Profiles and Levels

2.1.1 Object Profiles

Profile	Hierarchy	Tools supported:	Object Profile ID
reserved			0
AAC Main	contains AAC LC	13818-7 main profile PNS	1
AAC LC		13818-7 LC profile PNS	2
AAC SSR		13818-7 SSR profile PNS	3
T/F		13818-7 LC PNS LTP	4
T/F Main scalable	contains T/F LC scalable	13818-7 main PNS LTP BSAC tools for large step scalability (TLSS) core codecs: CELP, TwinVQ, HILN	5
T/F LC scalable		13818-7 LC PNS LTP BSAC	6

		tools for large step scalability (TLSS) core codecs. CELP, TwinVQ, HILN	
TwinVQ core		TwinVQ	7
CELP		CELP	8
HVXC		HVXC	9
HILN		HILN	10
TTSI		Text-To-Speech Interface	11
Main Synthetic	contains Wavetable Synthesis	all structured audio tools	12
Wavetable Synthesis		SASBF MIDI	13
reserved			14
reserved			15

2.1.2 Combination Profiles

[make reference to systems 7.3.3.3 DecoderConfigDescriptor, profileAndLevelIndication Values !]

Combination Profile	Hierarchy	Audio Object Profiles supported:
Main	Contains Scalable, Speech and Low Rate Synthetic	AAC Main, LC, SSR T/F, T/F Main Scalable, T/F LC Scalable TwinVQ core CELP HVXC HILN Main Synthetic TTSI
Scalable	Contains Speech	T/F LC Scalable AAC-LC or/and T/F CELP HVXC TwinVQ core HILN Wavetable Synthesis TTSI
Speech		CELP HVXC TTSI
Low Rate Synthesis		Wavetable Synthesis TTSI

2.1.3 Complexity Units

Complexity units are defined to give an approximation of the decoder complexity in terms of processing power and RAM usage required for processing MPEG-4 Audio bitstreams in dependence of specific parameters.

The approximated processing power is given in “Processor Complexity Units” (PCU), specified in integer numbers of MOPS. The approximated RAM usage is given in “RAM Complexity Units” (RCU), specified in (mostly) integer numbers of kWords (1000 words). The RCU numbers do not include working buffers which can be shared between different objects and/or channels. Total complexity estimates for single-object / single-channel decoders which also include ROM requirements are given in Section 2.2.

If a level of a profile is specified by the maximum number of complexity units, a flexible configuration of the decoder handling different types of objects is allowed under the constraint that both values for the total complexity for decoding and sampling rate conversion (if needed) do not exceed this limit.

The following table gives complexity estimates for the different object profiles:

Object Profile	Parameters	PCU (MOPS)	RCU (kWords)	Remarks
AAC Main	fs = 48 kHz	5	5	1)
AAC LC	fs = 48 kHz	3	3	1)
AAC SSR	fs = 48 kHz	4	3	1)
T/F	fs = 48 kHz	4	3	1)
T/F Main Scalable	fs = 48 kHz	6	5	1), 2)
T/F LC Scalable	fs = 48 kHz	5	3	1), 2)
TwinVQ core	fs = 24 kHz	2	3	1)
CELP	fs = 8 kHz	1	1	
CELP	fs = 16 kHz	2	1	
CELP	fs = 8/16 kHz (bandwidth scalable)	4	1	
HVXC	fs = 8 kHz	2	1	
HILN	fs = 8 kHz, ns = 40	5	2	1), 3)
HILN	fs = 8 kHz, ns = 90	10	2	1), 3)
TTSI		-	-	4)
Wavetable Synthesis	fs= ???, nt = ???	under study	under study, 5)	
Main Synthetic		under study	under study	
Sampling Rate Conversion	rf = 2, 3, 4, 6	2	(.5)	

Definitions:

- fs = sampling frequency
- ns = max. number of sinusoids to be synthesized
- nt = max. number of tones to be synthesized simultaneously
- rf = ratio of sampling rates

Notes:

1) PCU proportional to sampling frequency
2) Includes core decoder
3) RCU proportional to sampling frequency
4) The complexity for speech synthesis is not taken into account
5) Size of wavetable

2.1.4 Levels within the Combination Profiles

Levels for Main Combination Profile

Four levels are defined by complexity units:

1. PCU < 40, RCU < 20
2. PCU < 80, RCU < 64
3. PCU < 160, RCU < 128
4. PCU < 320, RCU < 64000

Levels for Scalable Combination Profile

Four levels are defined by configuration, the fourth level is defined by complexity units:

1. 24kHz 1 channel or 1 object (all object profiles) or 2 objects (overlay of a speech object and one low complexity SA or HILN object at 16 kHz)
2. 24kHz 2 channels or 2 objects
3. 48kHz 2 channels or 2 objects (48 kHz t/f or speech coding)
4. 48 kHz/24 kHz 5 channels or multiple objects, max. one integer factor sampling rate conversion for a maximum of two channels. Flexible configuration is allowed. PCU < 30, RCU < 19.

Levels for Speech Combination Profile

Only one level is defined for one speech object.

Levels for Low Rate Synthesis Combination Profile

Two levels are defined by the maximum number of tones to be synthesized simultaneously, the size of the sample RAM and by the maximum number of TTS objects which can be transmitted:

1. max. 8 tones to be synthesized simultaneously, 64 kWords of sample RAM
   max. 2 TTS objects
2. max. 24 tones to be synthesized simultaneously, 256 kWords of sample RAM
   max. 8 TTS objects

2.2 Complexity Figures for the Natural Audio Coding Algorithms

Overview of the complexity of the MPEG-4 Natural Audio Coding Algorithms

Complexity	AAC	AAC-LC	BSAC	Twin VQ	HILN	HVXC	NB-CELP	WB-CELP
Memory Requirements for 1 Audio Channel and Minimum Word Length								
RAM (Words)	4256	2232	4346	4240	3000	1500	650	830
ROM (Words)	3545	3545	3618	43000	4000	7700	2300	1000

min. Word Length	>=20	>=20	>=20	>=20	16	16	16	24 (16)
Computational Complexity calculated for a Sampling Rate of								
kHz	48	48	48	48	8	8	8	16
MOPS/ MIPS	5	2.5	5.7	3	4 typ. 10 max	5.1	1.5	3

The above numbers relate to a decoder for one audio channel. For a multi-channel system the numbers given for the RAM size and the computational complexity have to be multiplied with the number of audio channels to get a first estimation. The figures given for AAC have been calculated during the MPEG-2 AAC standardization process. The Wide-Band CELP figures are derived from an implementation on a Motorola DSP 56002. The figures given for the computational complexity may have been calculated with different methods, and may therefore not be directly comparable, but only give a first impression. Also, all the figures are not guaranteed to be the absolute maximum.

3 Interface to MPEG-4 Systems

3.1 Syntax

3.1.1 Audio DecoderSpecificInfo

In case that DecoderConfigDescriptor() (see ISO/IEC 14496-1 MPEG4 Systems) is used for MPEG-4 audio decoders, the array specificInfoByte[] shall contain AudioSpecificInfo() defined as follows:

```
aligned (8) class AudioSpecificConfig() {
        uint(4) objectProfile; /* see also Clause 2.1.1, „Object Profiles“ in this document; remark :
                                  composition profile and  level are already signaled by the
                                 profileAndLevelIndication value in DecoderConfigDescriptor */
        uint(4) samplingFrequencyIndex;
        if (samplingFrequencyIndex==0xf){
                uint(24) samplingFrequency;/* in Hz */
        }

        uint(4) channelConfiguration;

        if (objectProfile<8) { /* this is T/F  (AAC/TwinVq)*/
                TFSpecificConfig tfConfig(  uint(4) samplingFrequencyIndex);
        }
        if (objectProfile==8) { /* this is Celp */
                CelpSpecificConfig celpConfig( uint(4) samplingFrequencyIndex);
        }
        if (objectProfile==9 or 10 ) { /* this is Parametric , HVXC or HILN*/
                ParametricSpecificConfig paramConfig( );
        }
        if (objectProfile==11) { /* this is TextToSpeech */
                TTSSpecificConfig ttsConfig( );
        }
```

```
        if (objectProfile==12 or 13) { /* this is structured audio , Main Synthetic or Wavetable
                                          Synthesis*/
            StructuredAudioSpecificConfig strucConfig( );
        }

}
```

3.1.2 ParametricSpecificConfig

Defined in subpart 2.

3.1.3 CelpSpecificConfig

Defined in subpart 3.

3.1.4 TFSpecificConfig

Defined in subpart 4.

3.1.5 StructuredAudioSpecificConfig

Defined in subpart 5.

3.1.6 TTSSpecificConfig

Defined in subpart 6.

3.1.7 Payloads

The actual payloads for each configuration are defined in the corresponding subparts. These are the basic entities to be carried by the systems transport layer.

3.2 Semantics

3.2.1 objectProfile

A four bit field indicating the object profile. This is the master switch which selects the actual bit stream syntax of the audio data. In general, different object profiles use a different bit stream syntax. The interpretation of this field is given in the Object Profiles table above.

3.2.2 SamplingFrequencyIndex

A four bit field indicating the sampling rate used.

Value	samplingFrequencyIndex
0x0	**96000**
0x1	**88200**
0x2	**64000**
0x3	**48000**
0x4	**44100**
0x5	**32000**
0x6	**24000**
0x7	**22050**

0x8	**16000**
0x9	**12000**
0xa	**11025**
0xb	**8000**
0xc	**reserved**
0xd	**reserved**
0xe	**reserved**
0xf	**escape value**

3.2.3 channelConfiguration

A four bit field indicating the channel configuration

value	number of channels	channel to speaker mapping
0	**-**	**defined in audioDecderSpecificConfig**
1	**1**	**center front speaker**
2	**2**	**left, right front speakers**
3	**3**	**center front speaker,** **left, right front speakers**
4	**4**	**center front speaker,** **left, right center front speakers,** **rear surround speakers**
5	**5**	**center front speaker,** **left, right front speakers,** **left surround, right surround rear speakers**
6	**5+1**	**center front speaker,** **left, right front speakers,** **left surround, right surround rear speakers,** **front low frequency effects speaker**

7	**7+1**	**center front speaker** **left, right center front speakers,** **left, right outside front speakers,** **left surround, right surround rear speakers,** **front low frequency effects speaker**
8-15	**-**	**reserved**

4 Tools for Other Functionalities

This section describes the functionalities that are applicable across the natural audio coding schemes (parametric, CELP and t/f). Functionalities that are applicable to the individual coding schemes can be found in the corresponding file.

4.1 Speed change

4.1.1 Tool description

PICOLA (Pointer Interval Controlled OverLap Add) speed control tool supports speed change functionality for mono-channel signals sampled at 8kHz or 16kHz. The speed control is achieved by replacing a part of the input signal with an overlap-added waveform or by inserting the overlap-added waveform into the input signal.

4.1.2 Definitions

InputSignal[]: This array contains the input signal from the decoder.

NumInputSignal: This field contains the number of samples of the input signal InputSignal[].

OutputSignal[]: This array contains the speed-changed signal and is of a fixed length.

SpeedControlFactor: This field contains the speed control ratio. (SpeedControlFactor = 1.0 indicates normal speed.)

DecodeFlag: This field contains the flag which indicates whether the input signal of the next frame is required for speed change processing.

DecodeFlag	Description
0	Input signal of the next frame is not required
1	Input signal of the next frame is required

OutputFlag: This field contains the flag which indicates whether the output array OutputSignal[] contains speed-changed samples, the size of which is less than the fixed output array size, the outstanding difference being filled by the next speed-changed signal.

OutputFlag	Description
0	Speed-changed signal has size less than the fixed-size output array
1	Speed-changed signal has size equal to the fixed-size output array and therefore be outputted.

4.1.3 Speed control process

The block diagram of the speed controller is shown in Figure 1. The input signal InputSignal[], which is an output from the decoder with a given frame length NumInputSample, is stored in the buffer memory. Adjacent waveforms with the same length are extracted in pairs from the memory buffer and the pair with the minimum distortion between the two waveforms is selected. The selected waveforms are overlap-added. The speed control is achieved by replacing a part of the input signal with the overlap-added waveform or by inserting the overlap-added waveform into the input signal. The speed controller outputs the speed changed signal with a certain fixed length frame. Normally, the frame length is to be the same length as that of the associated decoder. In the case where the frame length of the associated decoder is variable, the maximum possible frame length is applied. Since the number of samples of the output signal differs from that of the input signal, the input and the output are controlled using flags, namely, the DecodeFlag and OutputFlag. In the case where the samples of the input signal in the buffer are not enough to carry out the speed change, DecodeFlag is set to 1 and the decoded signal from the associated decoder is inputted. On the other hand, in the case where samples of the speed-changed signal has size equal to the fixed-size output array OutputSignal[], OutputFlag is set to 1 and the speed-changed signal is outputted. Details of the processing are described below.

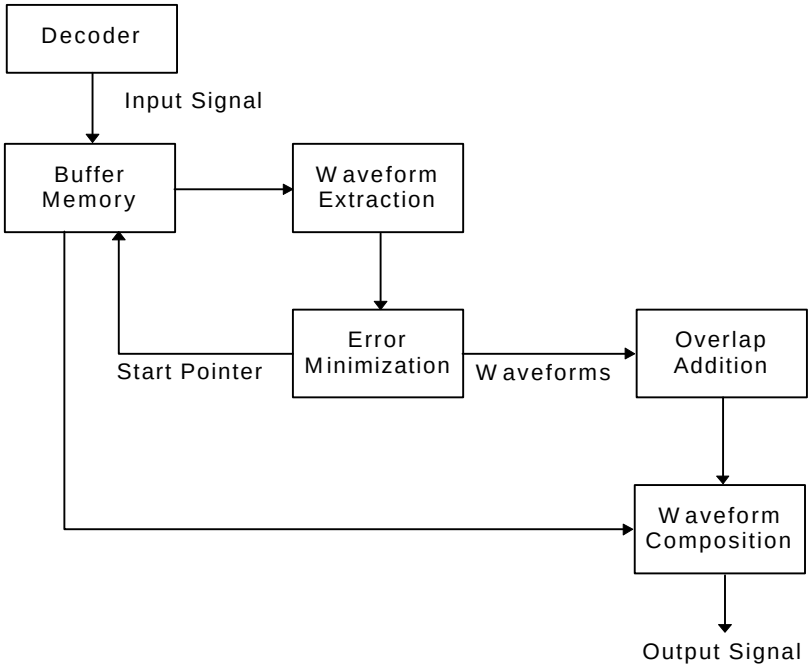

Figure 1. Block Diagram of the Speed Controller

4.1.3.1 Time scale compression (High speed replay)

The compression principle is shown in Figure 2. P0 is the pointer which indicates the starting sample of the current processing frame in the memory buffer. The processing frame has a length of LF samples and comprises adjacent waveforms each of length LW samples. The average distortion per sample between the first half of the processing frame (waveform A) and the second half (waveform B) is calculated as shown below.

$$\mathrm{D}(\mathrm{LW}) = \frac{1}{\mathrm{LW}} \sum_{n=0}^{\mathrm{LW}-1} \{x(n) - y(n)\}^2 \quad (\mathrm{PMIN} \le \mathrm{LW} \le \mathrm{PMAX})$$

where, D(LW) is the average distortion between the two waveforms when the waveform length is LW, $x(n)$ is the waveform A, $y(n)$ is the waveform B, PMIN is the minimum length of the candidate waveform and PMAX is the maximum length of the candidate waveform. Typically PMIN=32 and PMAX=160 for 8kHz sampling rate, PMIN=80 and PMAX=320 for 16kHz sampling.

The length LW which minimizes the distortion D(LW) is selected, and corresponding waveforms A and B are determined. If the cross-correlation between the selected waveforms A and B is negative, the pointer P0 is shifted forward by the frame length of the decoder and the length LW is determined again with the processing frame starting from the updated pointer P0. After the waveform length LW is determined, the waveform A is windowed by a half triangular window with a descending slope, and the waveform B is windowed by a half triangular window with an ascending slope. The overlap-added waveform C is obtained by linearly adding the windowed waveform A and waveform B. Then, the pointer P0 moves to the point P1. The distance L from the beginning of waveform C to the pointer P1 is given by;

$$\mathrm{L} = \mathrm{LW} \bullet \frac{1}{\mathrm{SpeedControlFactor} - 1} \qquad (1 < \mathrm{SpeedControlFactor} \le 2)$$

L samples from the beginning of waveform C are outputted as the compressed signal. If L is greater than LW, the original waveform D which follows the waveform B is outputted. Therefore the length of the signal is shortened from LW+L samples to L samples. The updated pointer P1 indicates the starting sample P0' of the next processing frame.

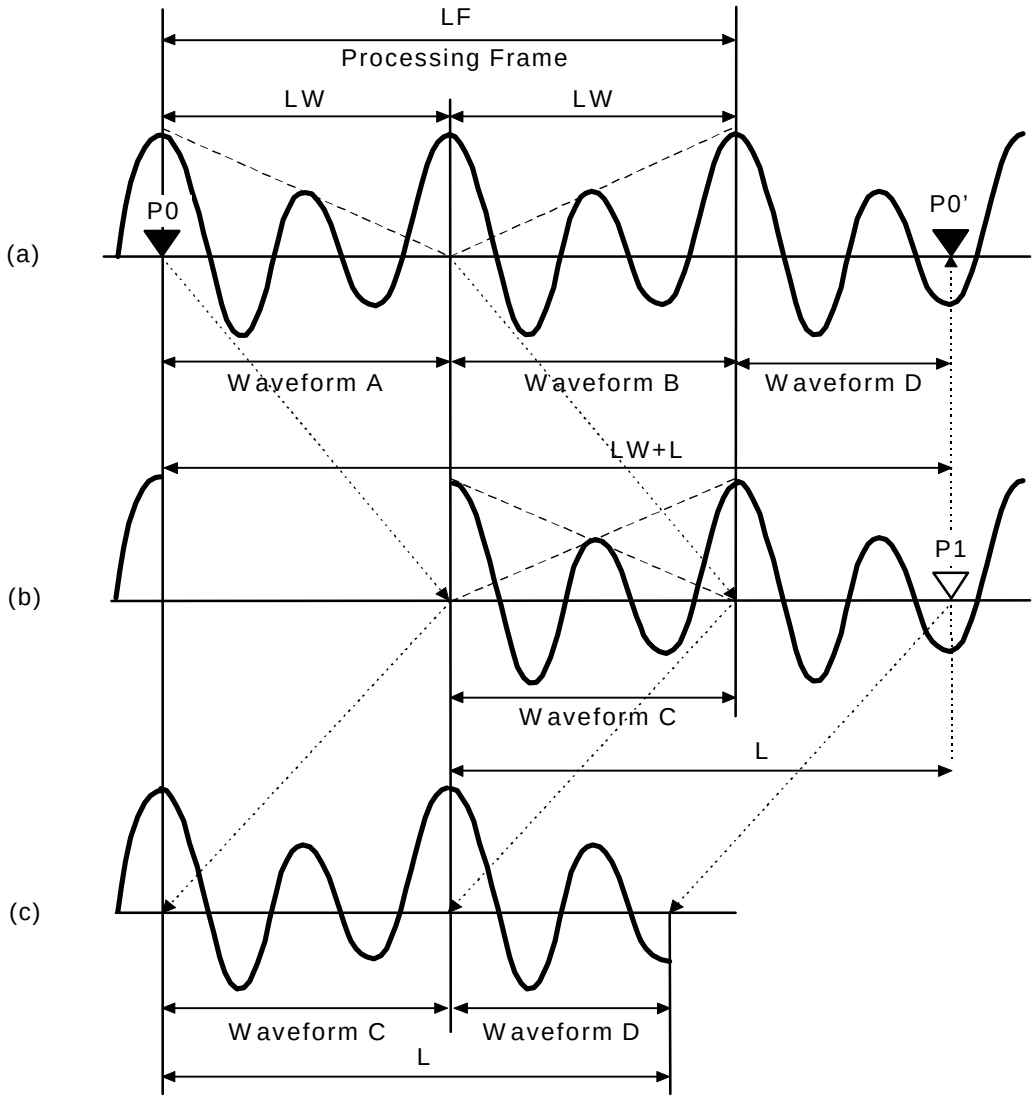

Figure 2. Principle of Time Scale Compression

(a) Orginal signal; (b) Overlap-added waveform; (c) Compressed signal

4.1.3.2 Time scale expansion (Low speed replay)

The expansion principle is shown in Figure 3. P0 is the pointer which indicates the starting sample of the current processing frame in the memory buffer. The processing frame has a length of LF samples and includes adjacent waveforms each of length LW samples. After the waveform length LW is determined using the same method as described in the time scale compression, the first half of the processing frame (waveform A) is outputted without any modification. Next, the first half (waveform

A) is windowed by a half triangular window with an ascending slope, and the second half (waveform B) is windowed by a half triangular window with a descending slope. The overlap-added waveform C is obtained by linearly adding the windowed waveform A and waveform B. Then, the pointer P0 moves to the point P1. The distance L from the beginning of waveform C to the pointer P1 is given by;

$$L = LW \bullet \frac{SpeedControlFactor}{1 - SpeedControlFactor} \qquad (0.5 \le SpeedControlFactor < 1)$$

L samples from the beginning of waveform C are outputted as the expanded signal. If L is greater than LW, the original waveform B is repeated as the output. The length of the signal is therefore expanded from L samples to LW+L samples. The updated pointer P1 indicates the starting sample P0' of the next processing frame.

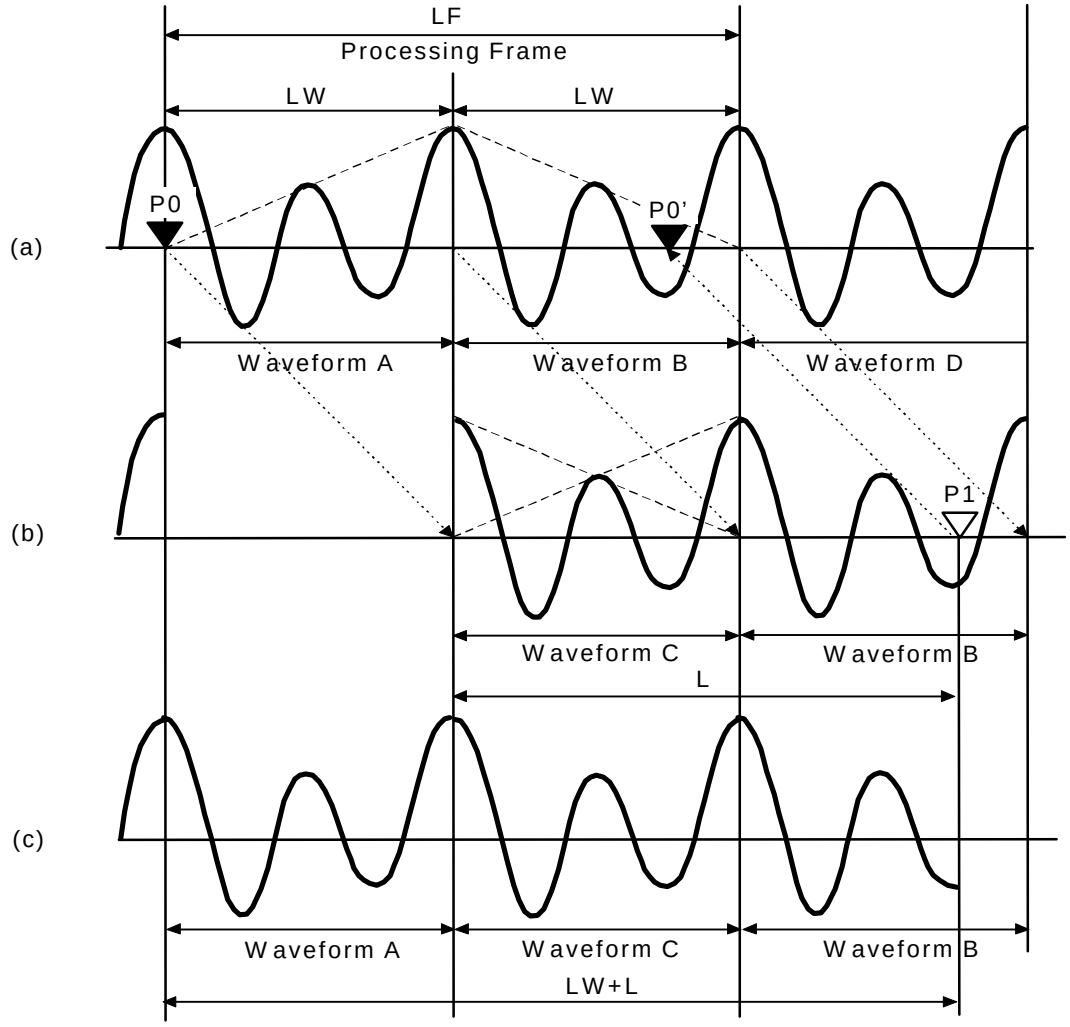

Figure 3. Principle of Time Scale Expansion

(a) Orginal signal; (b) Overlap-added waveform; (c) Expanded signal

Annex A (Informative)

1 PICOLA functional interface description

```
void mod_picola(
    float InputSignal[],                    /* input */
    int NumInputSample,                     /* input */
    float OutputSignal[],                   /* output */
    float SpeedControlFactor,               /* input */
    int *DecodeFlag,                        /* output */
    int *OutputFlag                         /* output */
)
```

Annex B (Informative)

2 Patent statements

(Table of organizations indicating to possess patent/intellectual property related to this specification to be added)

WG11 requests companies who believe they hold rights on patents that are necessary to implement MPEG-4 parts 1-2-3-5-6 to deliver a statements on company letterhead of compliance with ISO policy concerning use of patented items in International Standards. The patent statement can take a form similar to the statement given below:
<Company name> is pleased that the standardisation in relation to "Very low bitrate audio-visual coding" (known as MPEG-4) has reached Committee Draft level as documents ISO/IEC JTC1/SC29/WG11 N1901, N1902, N1903, N1905, N1906.
<Company name> hereby declares that it is prepared to license its patents, both granted and pending, which are necessary to manufacture, use, and sell implementations of the proposed MPEG-4 Systems, Visual, Audio, Reference Software and DMIF standards or combinations thereof.
<Company name> hereby also declares that it is aware of the rules governing inclusion of patented items in international standards, as described by Section 5.7, part 2 of the ISO/IEC Directives, and in particular that it is willing to grant a license to an unlimited number of applicants throughout the world under reasonable terms and conditions that are demonstrably free of any unfair competition.
This statement is intended to apply to the following parts of the proposed MPEG-4 standard (use the ones which apply):
Systems, Visual, Audio, Reference Software, DMIF".

I.