Contents for Subpart 4

Subpart 4: General Audio (GA) Coding: AAC/TwinVQ

4.1 Scope

The General Audio (GA) coding subpart of MPEG-4 Audio is mainly intended to be used for generic audio coding at all but the lowest bitrates. Typically, GA encoding is used for complex music material in mono from 6 kbit/s per channel and for stereo signals from 12 kbit/s per stereo signal up to broadcast quality audio at 64 kbit/s or more per channel. MPEG-4 coded material can be represented either by a single set of data, like in MPEG-1 and MPEG-2 Audio, or by several subsets which allow the decoding at different quality levels, depending on the number of subsets being available at the decoder side (bitrate scalability).

MPEG-2 Advanced Audio Coding (AAC) syntax (including support for multi-channel audio) is fully supported by MPEG-4 Audio GA coding. All the features and possibilities of the MPEG-2 AAC standard also apply to MPEG-4. AAC has been tested to allow for ITU-R ‘indistinguishable’ quality according to [4] at data rates of 320 kb/s for five full-bandwidth channel audio signals. In MPEG-4 the tools derived from MPEG-2 AAC are available together with other MPEG-4 GA coding tools which provide additional functionalities, like bit rate scalability and improved coding efficiency at very low bit rates. Bit rate scalability is either achieved with only GA coding tools, or by using a combination with an external (non-GA, e.g. CELP) core coder.

MPEG-4 GA coding is not restricted to some fixed bitrates but supports a wide range of bitrates and variable rate coding. While efficient mono, stereo and multi-channel coding is possible using extended, MPEG-2 AAC derived tools, the document also provides extensions to this tool set which allow mono/stereo scalability, where a mono signal can be extracted by decoding only subsets of the encoded stereo stream.

4.1.1 Technical Overview

4.1.1.1 Encoder and Decoder Block Diagrams

The block diagrams of the GA encoder and decoder reflect the structure of MPEG-4 GA coding. In general, there are the MPEG-2 AAC related tools with MPEG-4 add-ons for some of them and the tools related to the Twin-VQ quantization and coding. The Twin-VQ is an alternative module for the AAC-type quantization and it is based on an interleaved vector quantization and LPC (Linear Predictive Coding) spectral estimation. It operates from 6 kbit/s/ch and is recommended to be used below 16 kbit/s/ch with constant bitrate.

The basic structure of the MPEG-4 GA system is shown in Figures 4.1.1 and 4.1.2. 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.

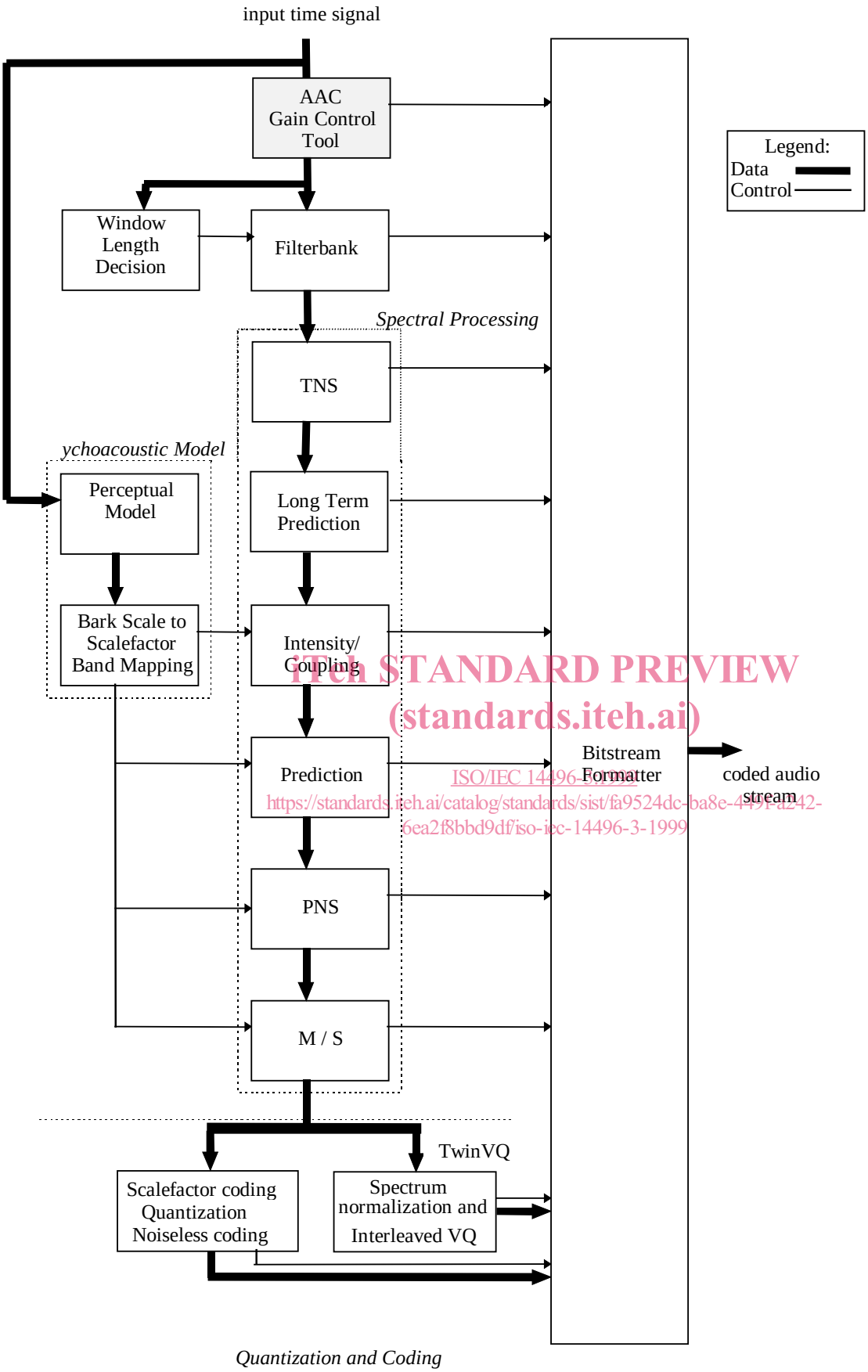

Figure 4.1.1 - Block diagram GA non scalable encoder

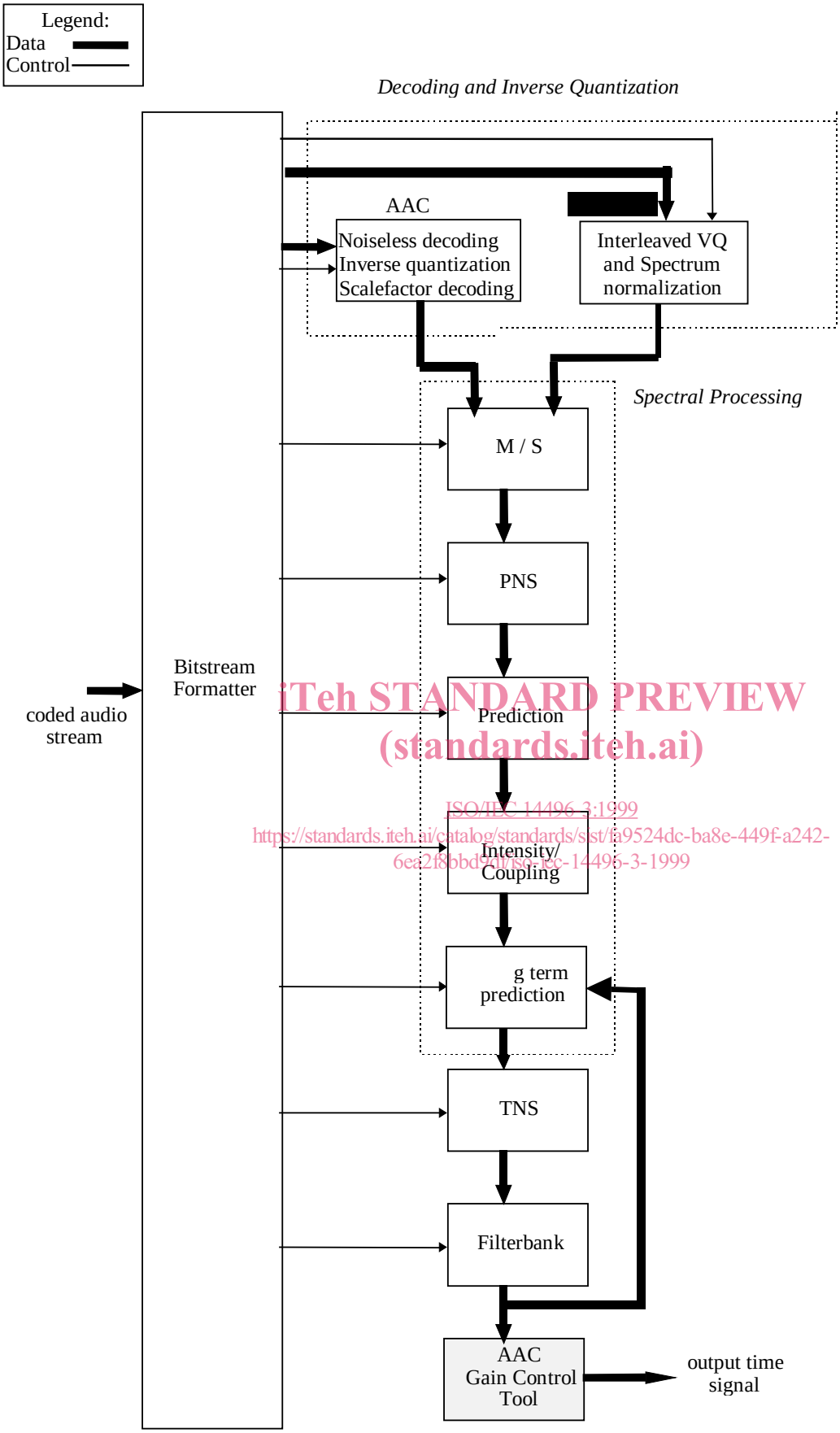

Figure 4.1.2 - Block diagram of the GA non scalable decoder

4.1.1.2 Overview of the Encoder and Decoder Tools

The input to the bitstream demultiplexer tool is the MPEG-4 GA bitstream. The demultiplexer separates the bitstream 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 quantized (and optionally noiselessly coded) spectra represented by either
  - the sectioning information and the noiselessly coded spectra (AAC) or
  - a set of indices of code vectors (TwinVQ)
- The M/S decision information (optional)
- The predictor side information (optional)
- The perceptual noise substitution (PNS) 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)
- Bitrate scalability related side information (optional)

The AAC 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, improving stereo imaging quality and sometimes providing 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 predictor side 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.

Alternatively, there is a forward adaptive long term prediction tool provided. The inputs to the long term prediction tool are:

- The reconstructed time domain output of the decoder
- The scaled, inversely quantized spectra

The output from the long term 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 perceptual noise substitution (PNS) tool implements noise substitution decoding on channel spectra by providing an efficient representation for noise-like signal components.
The inputs to the perceptual noise substitution tool are:

- The inversely quantized spectra
- The perceptual noise substitution control information

The output from the perceptual noise substitution tool is:

- The inversely quantized spectra

Note: If either part of this block is disabled, the scaled, inversely quantized spectra are passed directly through this part without modification. If the perceptual noise substitution block is not active, all spectra are passed through this block unmodified.

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 this part 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 (depending on **window_sequence**) spectral coefficients (if **frameLengthFlag** is set to '0'), or of 960 or 120 spectral coefficients (if **frameLengthFlag** is set to '1'), respectively. 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).

Two alternative, but very similar versions of this tool are available. The version with a frame length of 960 samples, which is not available in ISO/IEC 13818-7, allows for an integer frame length. For example at 48 kHz sampling rate, the frame length is exactly 20 ms with this version. This is especially useful for the CELP/AAC bitrate scalability combinations, where this allows the construction of combined CELP layer frames, which have a length of a multiple of 10 ms, and AAC enhancement layer frames. However, this feature can be used for configurations with only AAC or TwinVQ coding as well.

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 scalable sampling rate (SSR) object type only.

The spectrum normalization tool converts the reconstructed flat spectra to the actual values at the decoder. The spectral envelope is specified by LPC coefficients, a Bark scale envelope, periodic peak components, and gain.

The input to the spectral normalization tool are

- The reconstructed flat spectra
- The information of LPC coefficients, a Bark scale envelope, periodic peak components and gain

The output from the spectral normalization tool is

- The reconstructed actual spectra

The interleaved VQ tool converts the vector index to the flattened spectra at the TwinVQ decoder by means of table look-up of the codebook and inverse interleaving of the spectra. Quantization noise is minimized by a weighted distortion measure at the encoder instead of an adaptive bit allocation. This is an alternative to the AAC quantization tool.

The input to the interleaved VQ tool is:

- A set of indices of the code vector.

The output from the TwinVQ tool is:

- The reconstructed flattened spectra

The Frequency Selective Switch (FSS) tool is used to control the combination of the AAC coding layer with both, TwinVQ, and CELP coding layer, if these are used as base layer coder in scalable configurations. In a second function this tool is applied to control the combination of mono and stereo coding layer in scalable configurations where both mono, and stereo coding layer are used to code a stereo input signal.

The Up-sampling Filter tool adapts the sampling rate of a CELP core coder, which can be used as base layer coder in scalable configurations, to the sampling rate of the AAC extension layer.

The input to the Upsampling Filter tool is:

- The output of a CELP core coder running at a lower sampling rate than the AAC extension layer

The output from the Up-sampling Filter tool is:

- The up-sampled CELP core coder output, matching the sampling rate of the AAC extension layer, transformed into the frequency domain with exactly the same frequency and time resolution as the AAC extension layer.

4.2 Normative References

[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] ITU-T Rec.H.222.0(1995) | ISO/IEC 13818-1:1996, *Information technology - Generic coding of moving pictures and associated audio information: – Part 1: Systems.*

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

[4] ISO/IEC 13818-7:1997, *Information technology - Generic coding of moving pictures and associated audio information: - Part 7: Advanced Audio Coding (AAC).*

4.3 GA-specific definitions

4.3.1 alias: Mirrored spectral component resulting from sampling

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

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

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

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

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

4.3.7 core coder: The term core coder is used to denote a base layer coder in certain scalability configurations. A core coder does not code the spectral samples of the MDCT filterbank of the subsequent AAC coding layers, but operates on a time domain signal. The output of the core decoder has to be up-sampled and transformed into the spectral domain, before it can be combined with the output of the AAC coding layers. Within the MPEG-4 Audio standard only the MPEG-4 CELP coder is a valid core coder. However, in principal, also another AAC coding layer, operating at a lower sampling rate, could be used on the time domain signal, and then combined wit the other coding layer in exactly the same way as described for the CELP coder, and would therefore be called a core coder.

4.3.8 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.

4.3.9 discrete cosine transform; DCT: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation.

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

4.3.11 FSS: Frequency Selective Switch. Module which selects one of two input signals independently in each scalefactor band.

4.3.12 hybrid filterbank: A serial combination of subband filterbank and MDCT. Used in MPEG-1 and MPEG-2 Audio.

4.3.13 IDCT: Inverse Discrete Cosine Transform.

4.3.14 IMDCT: Inverse Modified Discrete Cosine Transform.

4.3.15 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.

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

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

4.3.18 main audio channels: All single_channel_elements (see subclause 4.5.2.1) or channel_pair_elements (see subclause 4.5.2.1) in one program.

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

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

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

4.3.22 modified discrete cosine transform (MDCT): A transform which hasthe property of time domain aliasing cancellation. An analytical expression of the MDCT can be found in subclause 4.6.14.3.3.

4.3.23 M/S stereo: A method of removing imaging artifacts as well as exploiting stereo irrelevance or redundancy in stereophonic audio programs based on coding the sum and difference signal instead of the left and right channels.

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

4.3.25 PNS: Perceptual Noise Substitution.

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

4.3.27 program: A set of main audio channels, coupling_channel_elements (see subclause 4.5.2.1), lfe_channel_elements (see subclause 4.5.2.1), and associated data streams intended to be decoded and played back simultaneously A program may be defined by default, or specifically by a program_configuration_element (see subclause 4.5.1.2). A given single_channel_element (see subclause 4.5.2.1) , channel_pair_element (see subclause 4.5.2.1), coupling_channel_element, lfe_channel_element or data channel may accompany one or more programs in any given bitstream.

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

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

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

4.3.31 scalefactor index: A numerical code for a scalefactor.

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

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

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

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

4.3.36 TNS: Temporal Noise Shaping

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

4.4 Syntax

Two types of data are part of the MPEG-4 GA coder syntax. These are

1. Configuration information
2. Actual Payload

The payload is intended to be transported via the MPEG-4 Systems layer. These data contain all information variing on a frame to frame basis, and therefore carry the actual audio information.

The Configuration information is also transported via MPEG-4 systems. These elements contain configuration information, which is necessary for the decoding process and parsing of the Payload. However, an update is only necessary if there are changes in the configuration.

The configuation information and the payload are abstract elements which define all information for the decoding and parsing of the bitstream. However, for real applications these streams need a transport layer which cares for the delivery of these elements. Normally, this transport mechanism will be handled by MPEG-4 Systems. However, the interface format streams defined in the Annex A of subpart 1 define a simple way of multiplexing the header and the raw data streams.

4.4.1 GA Specific Configuration

Table 4.4.1 - Syntax of GA_SpecificConfig ()

Syntax	No. of bits	Mnemonic
GA_SpecificConfig(samplingFrequency, ChannelConfiguration, AudioObjectType)		
{		
FrameLengthFlag /* 1024 or 960 */	**1**	**uimsbf**
DependsOnCoreCoder;	**1**	**uimsbf**
If (dependsOnCoreCoder == 1){		
CoreCoderDelay	**14**	**uimsbf**
}		
ExtensionFlag	**1**	**uimsbf**
If (channelConfiguration == 0){		
Program_config_element();		
}		
If (extensionFlag==1) {		
<to be defined in mpeg4 phase 2>		
}		
}		

4.4.1.1 Program config element

Table 4.4.2 - Syntax of program_config_element()

Syntax	No. of bits	Mnemonic
Program_config_element()		
{		
element_instance_tag	**4**	**uimsbf**
object_type	**2**	**uimsbf**
sampling_frequency_index	**4**	**uimsbf**
num_front_channel_elements	**4**	**uimsbf**
num_side_channel_elements	**4**	**uimsbf**
num_back_channel_elements	**4**	**uimsbf**
num_lfe_channel_elements	**2**	**uimsbf**
num_assoc_data_elements	**3**	**uimsbf**
num_valid_cc_elements	**4**	**uimsbf**
mono_mixdown_present	**1**	**uimsbf**
if (mono_mixdown_present == 1)		
mono_mixdown_element_number	**4**	**uimsbf**
stereo_mixdown_present	**1**	**uimsbf**
if (stereo_mixdown_present == 1)		
stereo_mixdown_element_number	**4**	**uimsbf**
matrix_mixdown_idx_present	**1**	**uimsbf**
if (matrix_mixdown_idx_present == 1) {		
matrix_mixdown_idx	**2**	**uimsbf**
pseudo_surround_enable	**1**	**uimsbf**
}		
for (i = 0; i < num_front_channel_elements; i++) {		
front_element_is_cpe[i];	**1**	**bslbf**
front_element_tag_select[i];	**4**	**uimsbf**
}		
for (i = 0; i < num_side_channel_elements; i++) {		
side_element_is_cpe[i];	**1**	**bslbf**
side_element_tag_select[i];	**4**	**uimsbf**
}		
for (i = 0; i < num_back_channel_elements; i++) {		
back_element_is_cpe[i];	**1**	**bslbf**
back_element_tag_select[i];	**4**	**uimsbf**
}		
for (i = 0; i < num_lfe_channel_elements; i++)		
lfe_element_tag_select[i];	**4**	**uimsbf**
for (i = 0; i < num_assoc_data_elements; i++)		
assoc_data_element_tag_select[i];	**4**	**uimsbf**
for (i = 0; i < num_valid_cc_elements; i++) {		
cc_element_is_ind_sw[i];	**1**	**uimsbf**
valid_cc_element_tag_select[i];	**4**	**uimsbf**
}		
byte_alignment()		
comment_field_bytes	**8**	**uimsbf**
for (i = 0; i < comment_field_bytes; i++)		
comment_field_data[i];	**8**	**uimsbf**
}		

4.4.2 GA Bitstream Payloads

4.4.2.1 Payloads for the audio object types AAC_main, AAC_SSR, AAC_LC and AAC_LTP

Table 4.4.3 - Syntax of top level payload for audio object types AAC Main, SSR, LC, andLTP (raw_data_block())

Syntax	No. of bits	Mnemonic

Raw_data_block()
{
  while((id = **id_syn_ele**) != ID_END){
  switch (id) {
  case ID_SCE: single_channel_element()
  break;
  case ID_CPE: channel_pair_element()
  break;
  case ID_CCE: coupling_channel_element()
  break;
  case ID_LFE: lfe_channel_element()
  break;
  case ID_DSE: data_stream_element()
  break;
  case ID_PCE: program_config_element()
  break;
  case ID_FIL: fill_element()
  break;
  }
  }
}
	3	**uimsbf**

Table 4.4.4 - Syntax of single_channel_element()

Syntax	No. of bits	Mnemonic

single_channel_element()
{
  element_instance_tag
  individual_channel_stream(0,0)
}
	4	**uimsbf**

Table 4.4.5 - Syntax of channel_pair_element()

Syntax	No. of bits	Mnemonic
channel_pair_element() { **element_instance_tag**	**4**	**uimsbf**
common_window	**1**	**uimsbf**

if(common_window) {
  ics_info()
  ms_mask_present
	2	**uimsbf**

if(ms_mask_present == 1) {
  for(g=0; g < num_window_groups; g++) {
  for(sfb=0; sfb < max_sfb; sfb++) {
  ms_used[g][sfb**]**
	1	**uimsbf**

}
  }
  }
  }
  individual_channel_stream(common_window,0)
  individual_channel_stream(common_window,0)
}		