INTERNATIONAL STANDARD

ISO/IEC 23003-1

First edition
2007-02-15

Information technology — MPEG audio technologies —

Part 1: MPEG Surround

Technologies de l'information — Technologies audio MPEG —

Partie 1: Ambiance MPEG

Reference number
ISO/IEC 23003-1:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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.

ISO/IEC 23003-1 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 23003 consists of the following parts, under the general title *Information technology — MPEG audio technologies*:

— *Part 1: MPEG Surround*

Introduction

The following MPEG International Standard describes the coding of multi-channel signals based on a downmixed signal of the original multi-channel signal, and associated spatial parameters. It offers lowest possible data rate for coding of multi-channel signals, as well as an inherent mono or stereo downmix signal included in the data stream. Hence, a mono or stereo signal can be expanded to multi-channel by a very small additional data overhead. Furthermore, the International Standard describes binaural decoding of the MPEG Surround stream, enabling a surround sound experience over headphones. The International Standard furthermore defines an Enhanced Matrix Mode that enables a multi-channel upmix from a stereo signal without any spatial parameters.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

The ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO and IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the ISO and IEC. Information may be obtained from the companies listed in Annex J.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified in Annex J. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — MPEG audio technologies —

Part 1:
MPEG Surround

1 Scope

This International Standard describes the MPEG Surround standard (Spatial Audio Coding, SAC), that is capable of re-creating N channels based on M<N transmitted channels, and additional control data. In the preferred modes of operating the spatial audio coding system, the M channels can either be a single mono channel or a stereo channel pair. The control data represents a significant lower data rate than required for transmitting all N channels, making the coding very efficient while at the same time ensuring compatibility with both M channel devices and N channel devices.

This International Standard incorporates a number of tools enabling a number of features that allow for broad application of the International Standard. A key feature is the ability to scale the spatial image quality gradually from very low spatial overhead towards transparency. Another key feature is that the compatible decoder input can be made compatible to existing matrix surround technologies. All tools are grouped to cover certain profiles.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

ISO/IEC 14496-3, *Information technology — Coding of audio-visual objects — Part 3: Audio*

3 Terms and definitions

3.1 Definitions

For the purpose of this document, the following terms and definitions apply.

3.1.1
channel
input or output audio channel corresponding to a specific speaker, as given by Table 1 — and illustrated in Figure 1

Table 1 — Channel abbreviation and loudspeaker position

Channel abbreviation	Loudspeaker position
L	Left Front
R	Right Front
C	Center Front
LFE	Low Frequency Enhancement
Ls	Left Surround
Rs	Right Surround
Lc	Left Front Center
Rc	Right Front Center
Lsr	Rear Surround Left
Rsr	Rear Surround Right
Cs	Rear Center
Lsd	Left Surround Direct
Rsd	Right Surround Direct
Lss	Left Side Surround
Rss	Right Side Surround
Lw	Left Wide Front
Rw	Right Wide front
Lv	Left Front Vertical Height
Rv	Right Front Vertical Height
Cv	Center Front Vertical Height
Lvr	Left Surround Vertical Height Rear
Rvr	Right Surround Vertical Height Rear
Cvr	Center Vertical Height Rear
Lvss	Left Vertical Height Side Surround
Rvss	Right Vertical Height Side Surround
Ts	Top Center Surround
LFE2	Low Frequency Enhancement 2

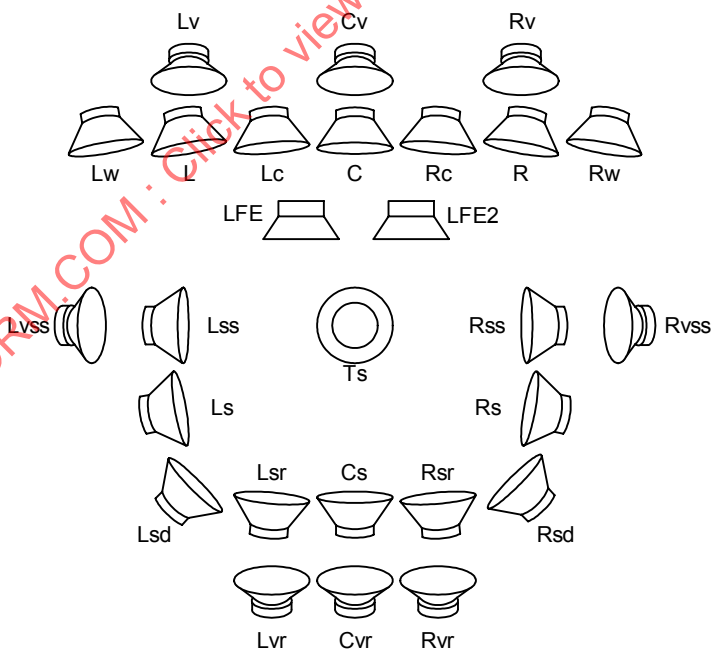

Figure 1 — Loudspeaker positions

3.1.2
Channel Level Difference
CLD
energy difference between two channels

3.1.3
CLD band
one or more hybrid subbands for which a single CLD parameter applies

3.1.4
Channel Prediction Coefficient
CPC
prediction coefficient used for re-creating three channels from two channels

3.1.5
CPC band
one or more hybrid subbands for which a single CPC parameter applies

3.1.6
hybrid filterbank
hybrid filter bank structure, consisting of a QMF bank and oddly modulated Nyquist filter banks, used to transform time domain signals into hybrid subband samples

3.1.7
hybrid filtering
filtering step on a QMF subband signal resulting in multiple hybrid subbands

NOTE The resulting hybrid subbands can be non-consecutive in frequency.

3.1.8
hybrid subband
subband obtained after hybrid filtering of a QMF subband

NOTE The hybrid subband can have the same time/frequency resolution as a QMF subband.

3.1.9
Inter Channel Correlation
ICC
correlation or coherence between two channels

3.1.10
ICC band
one or more hybrid subbands for which a single IIC parameter applies

3.1.11
NA
Not Applicable

3.1.12
M-N-M configuration
configuration of the spatial audio coding system that re-creates M channels from N downmixed channel and the corresponding spatial parameters, e.g. 5-1-5 configuration or 5-2-5 configuration

3.1.13
OTT box
conceptual one-to-two box that takes one channel as input and produces two channels as output

3.1.14
parameter band
one or more hybrid subbands applicable to one parameter

3.1.15
parameter band border
parameter band delimiter, expressed as a specific hybrid subband

3.1.16
parameter time slot
specific time slot for which the parameter is defined

3.1.17
parameter set
parameters associated with a specific parameter time slot

3.1.18
parameter subset
parameters associated with a specific parameter time slot and a specific OTT box or TTT box

3.1.19
processing band
one or more hybrid subbands defining the finest frequency resolution that could be controlled by the parameters

3.1.20
QMF bank
bank of complex exponentially modulated filters

3.1.21
QMF subband
subband obtained after QMF filtering of a time-domain signal, without any additional hybrid filtering stage

3.1.22
SAC
Spatial Audio Coder

3.1.23
SAC frame
time segment to which processing is applied according to the data conveyed in the corresponding SpatialFrame() syntax element

3.1.24
time segment
group of consecutive time slots

3.1.25
time slot
finest resolution in time for SAC time borders

NOTE One time slot equals one subsample in the hybrid QMF domain.

3.1.26
TTT box
conceptual two-to-three box that takes two channels as input and produces three channels as output

3.2 Notation

The description of the Spatial Audio Coder uses the following notation:

- Vectors are indicated by bold lower-case names, e.g. $\mathbf{vector}$.
- Matrices (and vectors of vectors) are indicated by bold upper-case single letter names, e.g. $\mathbf{M}$.

- Variables are indicated by italic, e.g. *variable*.
- Functions are indicated as *func(x)*.
- Real numbers are denoted by R
- Complex numbers are denoted by C

For equations in the text, normal mathematical interpretation is assumed (no rounding or truncation unless explicitly stated). For flowcharts, normal pseudo-code interpretation is assumed, with no rounding or truncation unless explicitly stated.

3.3 Operations

3.3.1 Scalar operations

X^* is the complex conjugate of X.

$y = \lfloor x \rfloor$ represents rounding down to the nearest integer, i.e., the largest integer number that is less than x.

or $y = \lceil x \rceil$ represents rounding up to the nearest integer, i.e., the smallest integer number that is not less than x.

$y = INT(x)$ represents truncation to integer (only keep the integer part), i.e., conversion to the integer number with the same sign as x and with an absolute value smaller than or equal to the absolute value of x.

$y = \log_2(x)$ is the base-2 logarithm of x.

$y = \log_{10}(x)$ is the base-2 logarithm of x.

$y = \min(,\ldots,)$ the minimum value in the argument list.

$y = \max(,\ldots,)$ the maximum value in the argument list.

$y = \mathrm{mod}(x, z)$ is the modulo operation y = (x - n*z) where n = ceil(x/z)-1 defined for z ~= 0..

$y = \mathrm{round}(x)$ represents rounding to the nearest integer. Halfway cases are rounded away from zero.

$y = \mathrm{sign}(x)$ the sign of x, hence defined as -1 for negative values of x, 1 for positive values and 0 for 0.

3.3.2 Vector operations

$\mathbf{y} = sort(\mathbf{x})$. $\mathbf{y}$ is equal to the sorted vector $\mathbf{x}$, where the elements of $\mathbf{x}$ are sorted in ascending order.

$y = length(\mathbf{x})$. y is the number of elements of the vector $\mathbf{x}$.

3.4 Constants

ε	A constant to avoid division by zero, e.g. 96 dB below maximum signal input.

3.5 Variables

$\mathbf{a}^{m}(l)$	aliasing condition vector defined for every parameter time slot l and all QMF subbands m that are the last subband (highest in frequency) within a parameter band.
ch	is the current audio channel.
$\mathbf{D}_{\mathrm{ATD}}$	is the three dimensional matrix holding arbitrary tree data, i.e. mapped CLD data, for every OTT box, every parameter set, and M_{proc} bands, for the arbitrary tree.
$\mathbf{D}_{\mathrm{CLD}}$	is the three dimensional matrix holding the dequantized, and mapped CLD data for every OTT box, every parameter set, and M_{proc} bands.
$\mathbf{D}_{\mathrm{ICC}}$	is the three dimensional matrix holding the dequantized, and mapped ICC data for every OTT or TTT box, every parameter set, and M_{proc} bands.
$\mathbf{D}_{\mathrm{CPC\_1}}$	is the three dimensional matrix holding the dequantized, and mapped first CPC data for every TTT box, every parameter set, and M_{proc} bands.
$\mathbf{D}_{\mathrm{CPC\_2}}$	is the three dimensional matrix holding the dequantized, and mapped second CPC data for every TTT box, every parameter set, and M_{proc} bands.
$\mathbf{D}_{\mathrm{CLD\_1}}$	is the three dimensional matrix holding the dequantized, and mapped first CLD data for every TTT box, every parameter set, and M_{proc} bands.
$\mathbf{D}_{\mathrm{CLD\_2}}$	is the three dimensional matrix holding the dequantized, and mapped second CLD data for every TTT box, every parameter set, and M_{proc} bands.
$\mathbf{D}^{\mathrm{Q}}_{YYY}$	is a three dimensional matrix similar to $\mathbf{D}_{YYY}$, the data is dequantized as for $\mathbf{D}_{YYY}$, however it has M_{par} bands of data. YYY can be any of CLD, ICC, CPC_1, CPC_2, CLD_1, CLD_2 or ATD.
$\mathbf{envRatio}_{X}$	is a vector with GES envelope data for each channel X.
Fs	is the sampling frequency of the Spatial Audio Tool.
$\mathbf{G}$	is a three dimensional matrix holding the dequantized and mapped gain correction data for all input channels, parameter set, and M_{proc} bands.
$\mathbf{G}^{\mathrm{Q}}$	is a three dimensional matrix holding the dequantized gain correction data for all input channels, parameter set, and M_{par} bands.
$\mathbf{idx}XXX(,,)$	is a three dimensional matrix holding the Huffman and delta decoded indices. XXX can be any of CLD, ICC, CPC_1, CPC_2, CLD_1, CLD_2 or ATD.
K	number of hybrid subbands, 71.
K_c	number of complex QMF subbands for Low Power MPEG Surround, $K_c = 8$.
L	number of parameter sets.

M_{proc}	is the number of processing bands, 28.
M_{proc}^{c}	number of complex processing bands for Low Power MPEG Surround, 12.
M_{par}	is the number of parameter bands signalled by *bsFreqRes*.
M_{QMF}	is the number of QMF subbands depending on sampling frequency as defined in subclause 6.3.3.
$\mathbf{m}_{\text{resPar}}$	is a vector with the number of parameter bands that each residual cover.
$\mathbf{m}_{\text{resProc}}$	is a vector with the number of processing bands that each residual cover.
$\mathbf{m}_{\text{tttLowProc}}$	is a vector with the number of processing bands for the low range in the TTT boxes.
$\mathbf{m}_{\text{tttHighProc}}$	is a vector with the number of processing bands for the high range in the TTT boxes.
$\mathbf{M}_{1}^{n,m}$	is the time and frequency variant pre- matrix, defined for all time slots n and all hybrid subbands m.
$\mathbf{M}_{2}^{n,m}$	is the time and frequency variant mix-matrix, defined for all time slots n and all hybrid subbands m.
$\mathbf{r}^{m}(l)$	weighted correlation sum based on the input downmix signal, defined for every parameter time slot l and all QMF subbands m that have an adjoining parameter border, used for Low Power MPEG surround.
reset	a variable (in the encoder and the decoder) set to one if certain data elements have changed from the previous frame, otherwise set to zero.
$\mathbf{S}_{\text{proc}}$	a matrix indicating for every parameter set and processing band if smoothing is applied.
$\mathbf{s}_{\text{delta}}$	a vector indicating for every time-slot the smoothing filter coefficient.
t	is of length *L* and contains parameter time slots for all CLD, ICC, and CPC parameter sets in the current frame.
$\mathbf{Tree}(ch,,)$	a 3 dimensional matrix, which for each input channel to the Arbitrary Tree have a column for each output signal of the sub-tree indexing the OTT modules the input signal must pass before the output is reached.
$\mathbf{Tree}_{\text{sign}}(ch,,)$	a 3 dimensional matrix, which for each input channel to the Arbitrary Tree have a column for each output signal of the sub-tree indicating whether the upper (1) or the lower (-1) output of an OTT module should be followed to reach the output signal.
$\mathbf{Tree}_{\text{depth}}(ch,)$	a matrix which for each input channel to the Arbitrary Tree have the number of OTT modules that are passed for every output channel.
$\mathbf{Tree}_{\text{outChan}}(ch)$	is a vector with *numOutChan* elements and each element contain the number of output channels for each Arbitrary Sub-tree.

$\mathbf{v}^{n,m}$ is a vector with the hybrid subband output from the pre gain matrix $\mathbf{M}_1^{n,m}\mathbf{x}^{n,m}$, defined for all time slots n and all hybrid subbands m.

$\mathbf{w}^{n,m}$ is a vector with the hybrid subband output from the decorrelators, the pre-gain matrix and residuals , defined for all time slots n and all hybrid subbands m.

$\mathbf{w}_{\text{diffuse}}^{n,m}$ is a vector with the hybrid subband output from the decorrelators, defined when temporal shaping is used, defined for all time slots n and all hybrid subbands m.

$\mathbf{w}_{\text{direct}}^{n,m}$ is a vector with the hybrid subband output from the decorrelators, the pre-gain matrix and residuals, defined when temporal shaping is used, defined for all time slots n and all hybrid subbands m.

$\mathbf{x}^{n,m}$ is a vector with the hybrid subband input signals (down-mix and residuals), defined for all time slots n and all hybrid subbands m.

$\mathbf{y}^{n,m}$ is a vector with the output hybrid subband signals, which are feed into the hybrid synthesis filter banks, defined for all time slots n and all hybrid subbands m.

$\mathbf{y}_{\text{diffuse}}^{n,m}$ is a vector with the output hybrid subband signals for the diffuse part of the output signal, which is defined when temporal processing is applied, defined for all time slots n and all hybrid subbands m.

$\mathbf{y}_{\text{direct}}^{n,m}$ is a vector with the output hybrid subband signals for the direct part of the output signal, which is defined when temporal processing is applied, defined for all time slots n and all hybrid subbands m.

4 MPEG Surround overview

4.1 Introduction

MPEG Surround is based on a principle called Spatial Audio Coding (SAC). Spatial Audio Coding is a multi-channel compression technique incorporating a backwards compatible downmix, and exploiting the perceptual inter-channel redundancy in multi-channel audio signals to achieve higher compression rates. In order to efficiently represent the spatial image of the multi-channel signal, the MPEG Surround encoder employs a number of spatial parameters:

- Channel level Differences (CLD) describing level differences between two channels,
- Inter channel Correlation / Coherences (ICC) describing the amount of correlation or coherence between two channels and
- Channel Prediction Coefficients (CPC) enabling the recreation of a third channel out of two channels by means of prediction.

During encoding, spatial cues are extracted from the multi-channel audio signal and a downmix is generated. Any number of channels can be used for the downmix, provided that it is less than that used for the original audio signal. The MPEG Surround system supports multi-channel signals of up to 27 channels.

All parameters are efficiently quantized and coded into the spatial bitstream. The quantization process is flexible to allow for a gradual quality versus bit-rate trade-off. For the extreme case where no spatial parameters are employed, the spatial decoder supports an Enhanced Matrix Mode, in which the parameters are derived directly from the down-mix. In addition, support for residual coding is included, that allows coding of the difference between the multi-channel signal as reproduced by the parametric model and the original. Hence, the MPEG Surround system, includes several tools to scale gracefully on the rate/distortion curve, as outlined by the following figure.

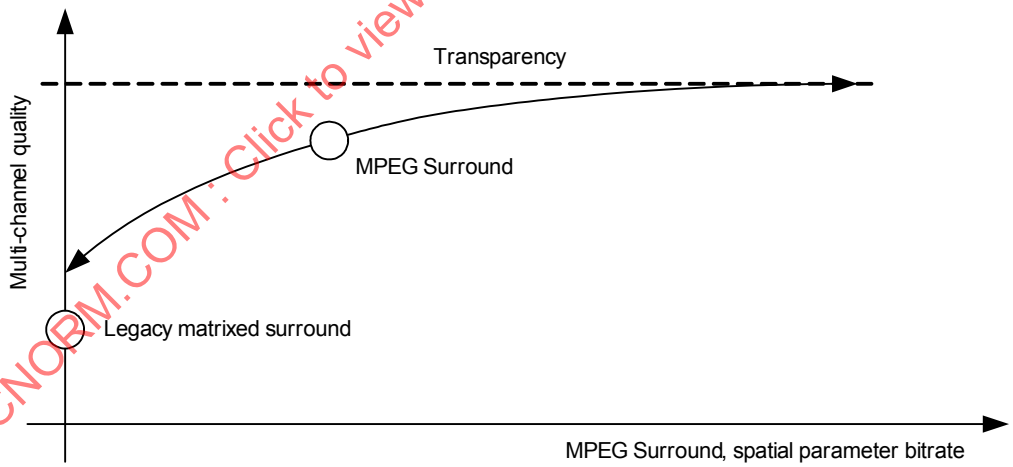

Figure 2 — MPEG Surround bit-rate quality scaling

The downmix can be compressed and transmitted without the need to update existing coders and infrastructures. The spatial cues, or spatial side information, are transmitted in a low bitrate side channel, e.g. the ancillary data portion of the downmix bit-stream.

For most audio productions both a stereo as well as a 5.1 multi-channel downmix is produced from the original multi-track recording by an audio engineer. Naturally, the internal (i.e. automated) downmix produced by the MPEG Surround encoder can differ significantly from the external (artistic) stereo downmix signal as intended by the audio engineer. The MPEG Surround encoder can therefore operate in two modes, "Internal Downmix Mode" or "External Downmix Mode". In the External Downmix Mode the artistic external downmix is transmitted instead of the internal downmix. The difference between this artistic stereo downmix and the automated stereo downmix signal, required by the decoder for optimal multi-channel reconstruction, is coded as part of the spatial side information stream either in a parametric fashion for low bitrate applications or as a wave-form coded difference signal, by means of residual coding.

On the decoder side, a multi-channel up-mix is created from the transmitted downmix signal and spatial side information. In this respect, the Spatial Audio Coding concept can be used as a pre- and post-processing step to upgrade existing systems. Figure 3 illustrates this for a 5.1 original with a stereo downmix. The MPEG Surround decoder supports two different modes, "Normal Mode" in which the spatial parameters obtained from the data stream is used for decoding, and a "Enhanced Matrix Mode" in which the decoder operates on the downmix signal only, estimating the required parameters from the received downmix.

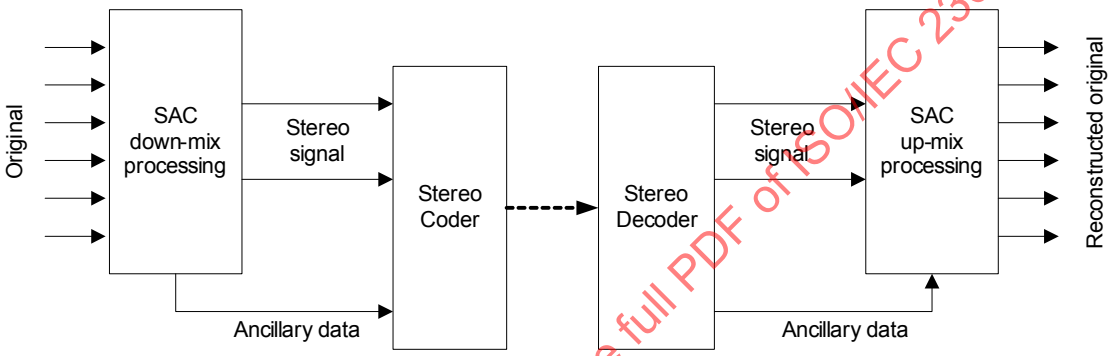

Figure 3 — Overview of the MPEG Surround decoder outlined for a stereo downmix

The spatial decoder combines the spatial parameters with the downmix channels into a multichannel output signal. Further, the MPEG Surround downmix can be provided in two different modes, "Normal Mode" and "Matrix Compatible Mode". If Matrix Compatible Mode is used, the MPEG Surround encoder provides a downmix signal representation that is suitable for playback via legacy matrix surround decoders.

4.2 Basic structure

The basic structure of the MPEG Surround system is shown in Figure 4. The data stream is de-quantized and decoded in the "Bit-stream decoder" modules, as outlined in subclause 6.1. The decoded spatial data is input to the modules calculating the pre-matrix and the mix-matrix as outlined in subclause 6.5. The residual coding data is input to the residual signal decoder as outlined in subclause 6.9. The input signal is analyzed by the QMF hybrid filterbank as outlined in subclause 6.3. The subband signal vector is multiplied with the pre-matrix as outlined in subclause 6.4. Part of the resulting signal vector is input to decorrelator units as outlined in subclause 6.6. The residual signals are also combined with the input signal vector as outlined in subclause 6.4. The mix matrix is applied according to subclause 6.4, and the temporal shaping is applied according to subclause 6.7 and 6.8. The output signal is synthesized according to subclause 6.3.

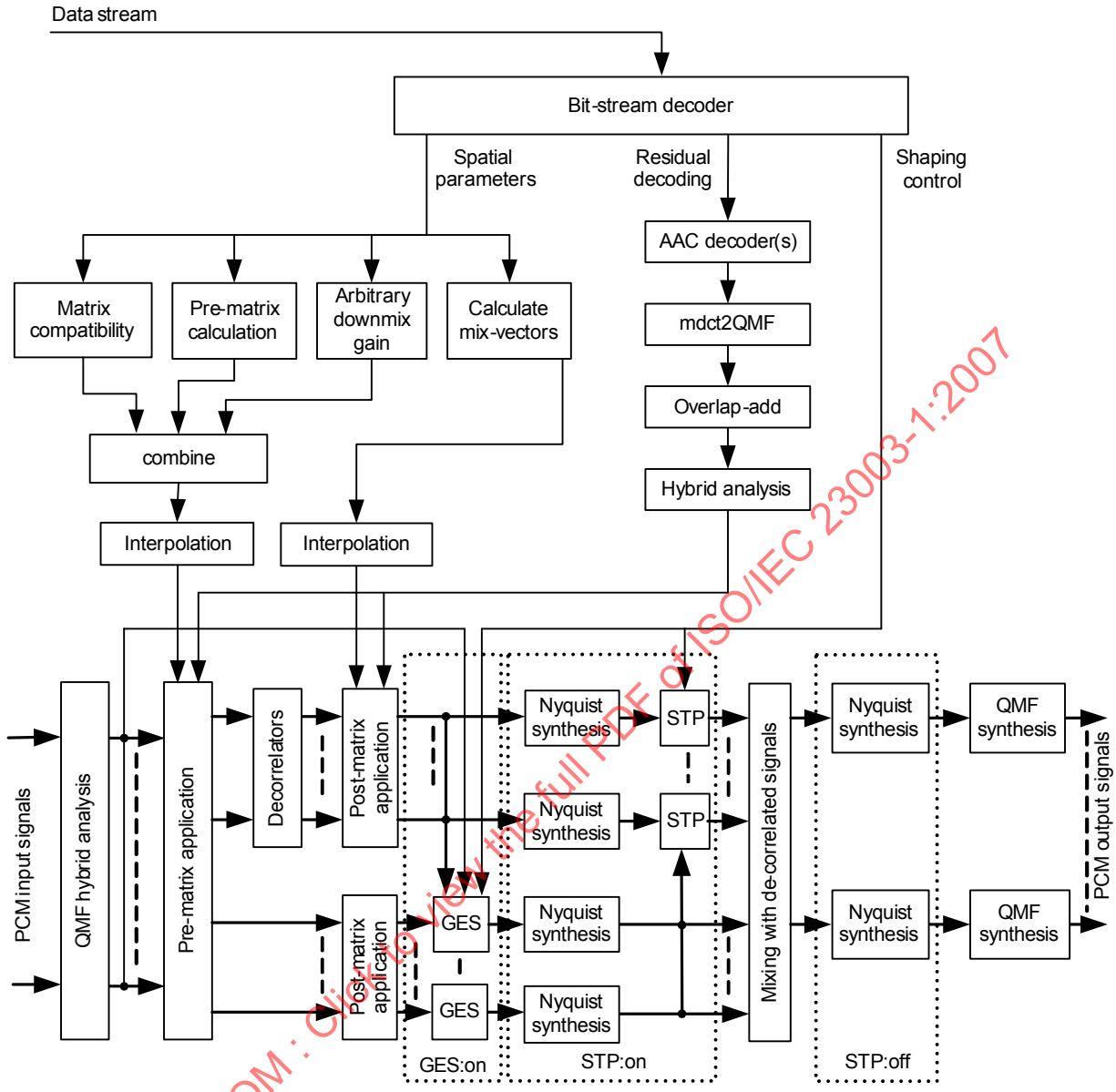

Figure 4 — Overview of the MPEG Surround decoder

4.3 Tools and functionality

4.3.1 General MPEG Surround tools

4.3.1.1 Introduction

The MPEG Surround system incorporates a number of tools that allow for flexible complexity and/or quality trade-off, as well as a diverse set of functionality. In the following subclauses some key-features of MPEG Surround are briefly outlined.

4.3.1.2 Residual coding

As the de-correlated signal is based on certain model assumptions, it is an incomplete representation that allows to scale towards transparent quality for increasing spatial bit-rate. By means of residual coding of the de-correlated signal this issue is addressed. For certain frequency regions where the de-correlated signal is not sufficient, the de-correlated signal is substituted by a residual waveform-type representation. One or more residual signals are coded using the MPEG-2 AAC Low Complexity syntax.

4.3.1.3 Temporal Shaping

Guided envelope shaping (GES) and Subband domain Temporal Processing (STP) are two tools that specifically address the preservation of the temporal structure in the output signal.

4.3.1.4 Enhanced Matrixed Mode of MPEG Surround

The MPEG Surround system includes an enhanced matrixed mode that creates a multi-channel signal based on the downmix without the transmission of a MPEG Surround bitstream. The parameters required in the MPEG Surround decoder are estimated from the received downmix signal. The basic principle is outlined in the following figure.

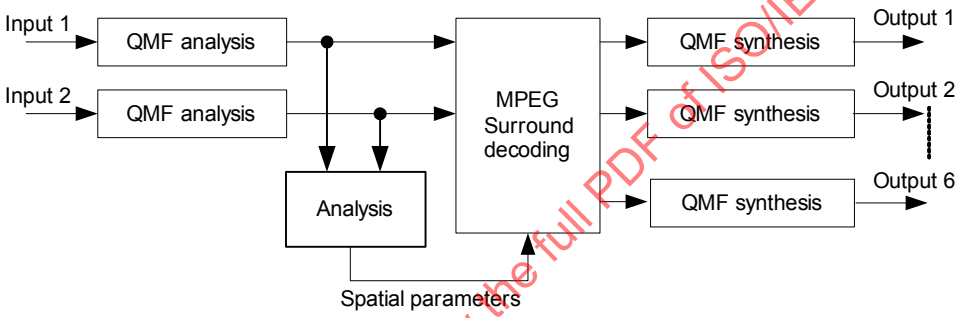

Figure 5 — Enhanced Matrix Mode of MPEG Surround

4.3.1.5 External downmix

The MPEG Surround system supports the use of external downmixes. The MPEG Surround encoder analyzes the difference between the internal downmix created by the MPEG Surround encoder and the external downmix. The difference is compensated for at the MPEG Surround decoder side. The basic principle is outlined in the following figure.

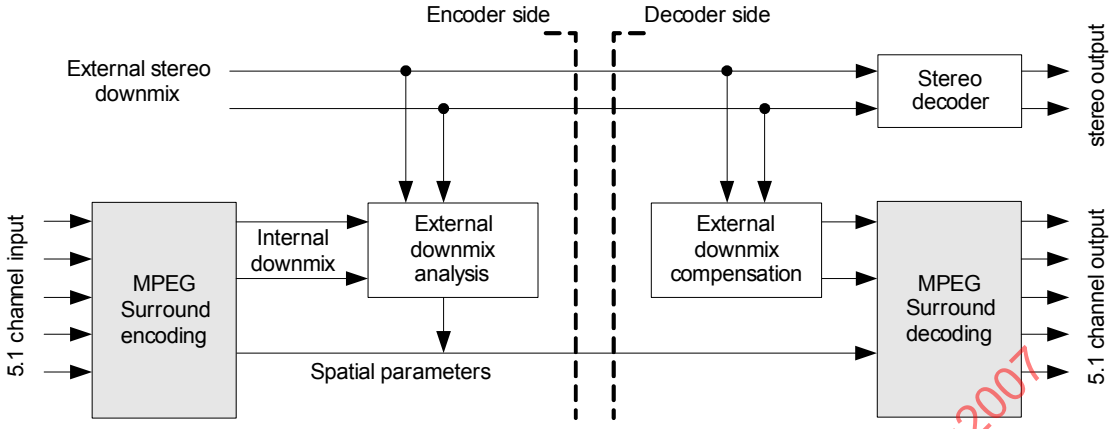

Figure 6 — Support for external downmixes within MPEG Surround

4.3.1.6 Matrix inversion (Matrix Compatible Mode)

In order to supply the highest level of backwards compatibility with legacy systems, the MPEG Surround system supports matrixed encoded downmixes. The MPEG Surround encoder can create a stereo downmix that is matrixed encoded, and can thus be decode by legacy matrixed surround decoders. The MPEG Surround decoder, will invert the matrixed encoded downmix, and produce the multi-channel signal based on the inverted downmix and the spatial parameters, without any degradation in quality due to the matrixed encoded downmix. The basic principle is outlined in the following figure.

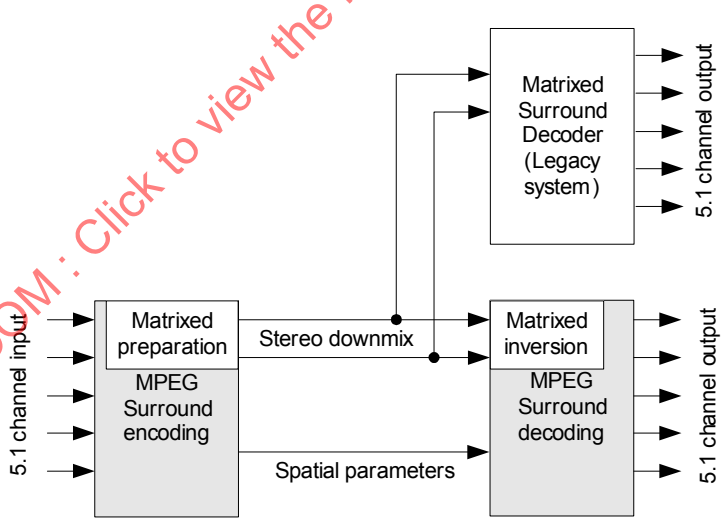

Figure 7 — Matrixed compatible MPEG Surround

4.3.1.7 Binaural and 3D stereo decoding

The MPEG Surround system can be operated in a binaural / 3D stereo mode. This enables a multi-channel impression over headphones by means of Head Related Transfer Function (HRTF) filtering. Two modes of operation can be distinguished, 3D stereo and binaural decoding. In the former, the binauralization is done on the MPEG Surround encoder side, and hence no decoder processing is required for stereo listening. For

normal MPEG Surround decoding, the binaural encoded downmix is inverted similarly to the inversion of the matrixed compatible downmix. The basic principle is outlined in the following figure.

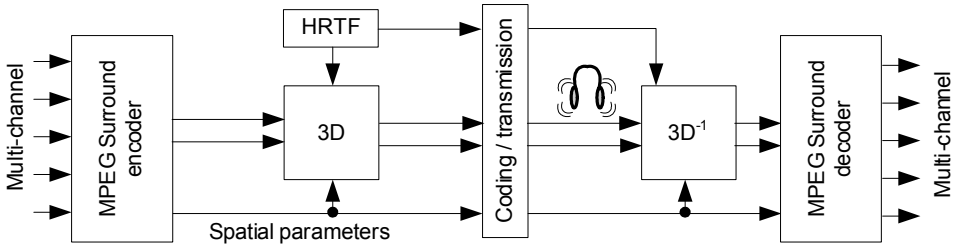

Figure 8 — 3D stereo decoding of MPEG Surround

For binaural decoding on the decoder side, the MPEG Surround downmix and spatial parameters are used in combination with HRTF filters supplied to the decoder. There are two modes of operation, a parametric approach, for lowest complexity, and a filtering approach for highest quality. The basic principle is outlined in the following figure.

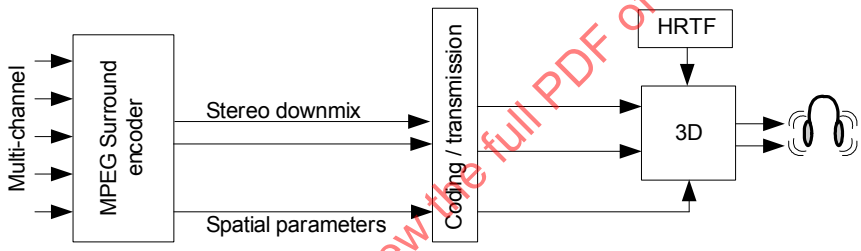

Figure 9 — Binaural decoding of MPEG Surround

4.3.2 High Quality and Low Power MPEG Surround

The MPEG Surround decoder can be implemented in a High Quality version and a Low Power version. The main difference are outlined by the following Table 2, and given in detail in subclause 6.10.

Table 2 — Outline of difference between the High Quality and the Low Power MEG Surround system

Tool or functionality	High Quality version	Low Power version
Filterbank	Complex valued QMF	Partially complex valued QMF
Residual coding	Supported over the entire frequency range	Only supported over the complex valued part of the frequency range
Binaural and 3D stereo	Parametric and HQ filtering supported	Only parametric HRTF filtering supported
Decorrelators		
Aliasing reduction	Not Applicable	Tool operates on the real-valued part of the frequency range

Both versions operate on the same data stream, albeit with different output signals.

4.4 Inter-connection of MPEG Surround with audio coders

The MPEG Surround decoder can be interfaced in either the time-domain or the QMF domain. In general, the MPEG Surround coder interfaces to the downmix channels by means of a 64 bands QMF domain frequency representation, identical to that standardized in ISO/IEC 14496-3 for High Quality Parametric Audio Coding. In the case the spatial coder is combined with HE-AAC, this QMF representation is directly available as an intermediate signal in the HE-AAC coder. In combination with alternative core coders, additional QMF analysis or synthesis modules, as defined in ISO/IEC 14496-3, are required for the spatial decoder and encoder, respectively. A Low Power MPEG Surround decoder cannot be connected in the QMF domain with a High Quality HE-AAC decoder, and vice versa a High Quality MPEG Surround decoder cannot be connected in the QMF domain with a Low Power HE-AAC decoder. For both combinations they shall be combined in the time-domain.

The 64 bands QMF representation of the downmix channels is further refined into hybrid subbands and grouped into parameter bands. By means of a pre-matrix operation on the hybrid subbands, generation of de-correlated signals and mixing, through application of the spatial parameters, multichannel hybrid subband signals are generated. In the last step, these are transformed to the time domain through an inverse QMF hybrid synthesis procedure resulting in the multi-channel time domain signal representation.

4.5 Delay and synchronization

The MPEG Surround decoder introduces a delay when processing the time domain signal from a downmix decoder. Two cases can be distinguished, connecting the MPEG Surround decoder with an arbitrary downmix coder (including High Efficiency AAC) via the time domain, and connecting the MPEG Surround decoder with a High Efficiency AAC decoder directly in the QMF domain. The direct connection in the QMF domain is not possible in case of upsampled or downsampled operation of MPEG Surround, as defined in subclause 6.3.3.

The MPEG Surround decoder introduces a total delay of 1281 samples for the High Quality decoder and 1601 samples for the Low Power decoder, as shown in Figure 10. The analysis filterbank as outlined in subclause 6.3.2 introduces a delay corresponding to 704 samples in the time domain. This is divided between 320 samples for the QMF analysis filtering and 384 samples for the Nyquist filtering. The synthesis filtering as outlined in subclause 6.3.2 introduces a delay corresponding to 257 samples in the time domain. This is divided into 0 samples for the hybrid synthesis filtering and 257 samples for the QMF synthesis filtering.

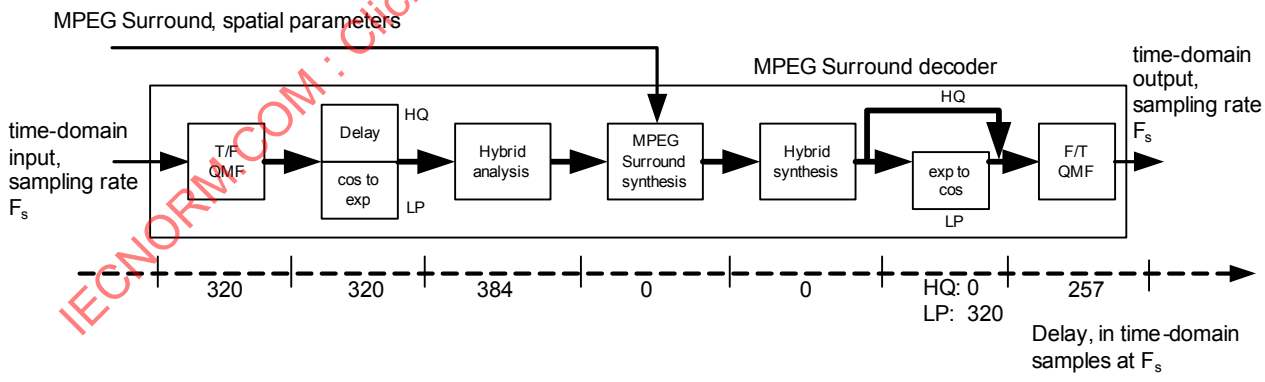

Figure 10 — Delay for the different parts of the MPEG Surround decoder

If the MPEG Surround decoder is connected with an arbitrary downmix coder (including High Efficiency AAC) via the time domain, as shown in Figure 11, the additional delay introduced by the MPEG Surround decoding process will be as outlined above.

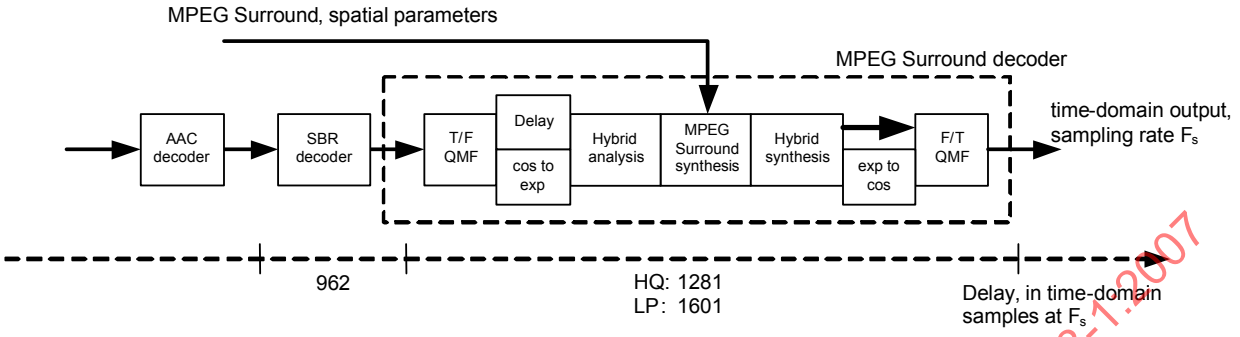

Figure 11 — Delay when connecting MPEG Surround in the time-domain for arbitrary core codec (including HE-AAC)

If the MPEG Surround decoder is directly connected with a High Efficiency AAC decoder via the QMF domain, as shown in Figure 12, the only additional delay is caused by the delay from the real to complex converter in the case of Low Power MPEG Surround and corresponding delay in case of High Quality MPEG Surround. This is because the QMF synthesis filtering is the same for High Efficiency AAC and MPEG Surround, and because the look-ahead of 384 samples needed for the Nyquist filtering is already available in the SBR tool of High Efficiency AAC, as described in ISO/IEC 14496-3 subclause 8.A.3.

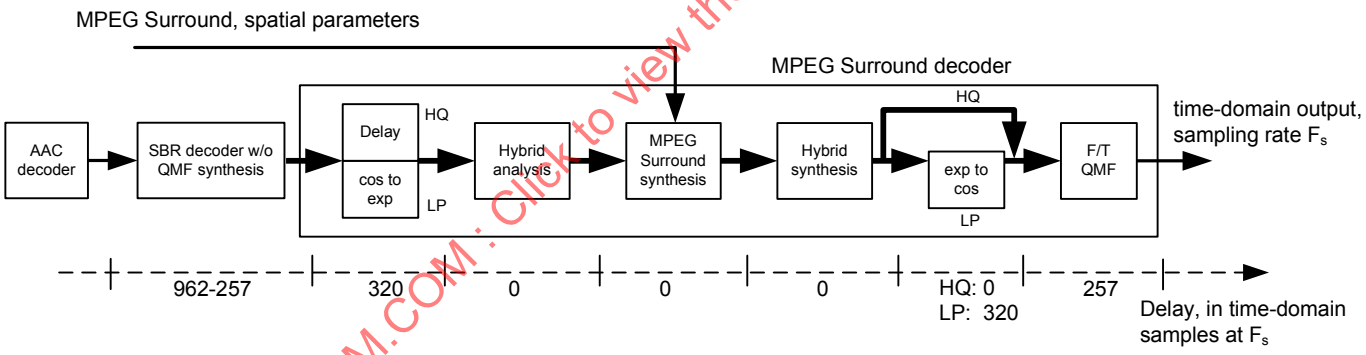

Figure 12 — Delay when connecting MPEG Surround with HE-AAC in the QMF-domain

Transmission of MPEG Surround side information with respect to transmission of the coded downmix signal is done in such a manner that there is no need to delay the downmix signal before it is processed by the MPEG Surround decoder. This means that MPEG Surround data is conveyed such that it is available when needed by the MPEG Surround decoding process. The temporal relationship between downmix data and spatial data is defined in clause 6. Note that special consideration is required if an MPEG Surround decoder and an HE-AAC decoder are connected in the time domain while a connection in the QMF domain would have been possible. In this case, the spatial parameters have to be delayed by 961 time samples, which is the sum of 257 samples for HE-AAC QMF synthesis and 704 samples for MPEG Surround QMF and Nyquist analysis.

In the case of Enhanced Matrix mode operation of the Low Power MPEG Surround decoder, the delay is the same as described above. In the case of Enhanced Matrix mode operation of the High Quality MPEG Surround decoder, the delay is 320 time-domain samples less than described above because the delay corresponding to the real to complex converter is not present.

In the case of single-slot parametric binaural decoding, the delay is the same as described above for both High Quality and Low Power MPEG Surround. In the case of multi-slot HRTF convolution approach, an additional delay is introduced depending on the initial delay of the HRTF used.

All delay figures given in this subclause up to now assume that MPEG Surround is operated in its normal mode using 64 band QMF banks. In the case of upsampled or downsampled operation of MPEG Surround, as defined in subclause 6.3.3, all delay figures measured in samples have to be multiplied by 2 (for upsampled operation) or multiplied by 0.5 (for downsampled operation).

4.6 Downmix gain

The downmix gain is applied to the mono or stereo downmix in the time domain prior to the QMF analysis. If an MPEG Surround decoder is directly connected to the output of an HE-AAC decoder in the QMF domain, the downmix gain is applied to the QMF domain input of MPEG Surround instead. The surround gain is applied to each surround channel (i.e., left surround and right surround for a 5.1 configuration) in the PCM domain after QMF synthesis. The LFE gain is applied to the LFE channel (as present in a 5.1 configuration) in the time domain after QMF synthesis.

4.7 MPEG Surround Profiles and Levels

4.7.1 Introduction

This subclause defines profiles and their levels for MPEG Surround.

Complexity units are defined to give an approximation of the decoder complexity in terms of processing power and RAM usage required for the MPEG Surround decoding process. The approximated processing power is given in “Processor Complexity Units” (PCU), specified in MOPS. The approximated RAM usage is given in “RAM Complexity Units” (RCU), specified in kWords (1000 words).

4.7.2 Baseline MPEG Surround Profile

The Baseline MPEG Surround Profile comprises the tools as indicated in the following table.

Table 3 — MPEG Surround Tools relation to the Baseline MPEG Surround Profile

MPEG Surround Tool	Included in the Baseline MPEG Surround Profile
Upmix (M1,M2 and M3, as defined by OTT and TTT modules, including decorrelators)	X
Parameterized External Downmix compensation (subclause 6.5.2.3.3)	X
Matrix compatibility (subclause 6.5.2.4)	X
Enhanced Matrix Mode decoding	X
Temporal Shaping	X
Residual Coding	X
Residual coding based External Downmix compensation (subclause 6.5.2.3.4)	
Binaural decoding (parametric and filtering)	X
3D Audio decoding	X
Low Power Decoding (as alternative to High Quality decoding)	X

Five different hierarchical levels are defined, which allow for different numbers of input and output channels, for different ranges of sampling rates, and for a different bandwidth of the residual signal decoding. The level of the decoder must be equal to or larger than the level of the bitstream in order to ensure proper decoding. In addition, decoders of level 1 and 2 are capable of decoding all bitstreams of level 2 and 3, though at a possibly slightly reduced quality due to the limitations of the decoder. The quality and format of the output of an MPEG Surround decoder furthermore depends on the specific decoder configuration, as detailed below. However, these decoder configuration aspects are completely orthogonal to the different levels of this profile.

An MPEG Surround decoder can be implemented in two different alternative versions:

- High Quality decoder (specified throughout the present document).
- Low Power decoder (difference to the HQ decoder specified in subclause 6.10).

An MPEG Surround decoder can be operated in two different modes, and support for both modes is mandatory for all levels:

- Normal mode (specified throughout the present document).
  In this mode, the decoder takes a downmix signal and a side information bitstream as input and generates a multi-channel output signal (or a binaural output signal).
- Enhanced matrix mode (differences to the normal mode specified in subclause 6.2).
  In this mode, the decoder takes a stereo downmix signal as input, e.g. a matrix surround stereo downmix, and generates a multi-channel output signal (or a binaural output signal).

An MPEG Surround decoder can provide three different output configurations, and support for all three configurations is mandatory for all levels except for level 1, where multi-channel output is not mandatory:

- Multi-channel output (specified throughout the present document).
  Typically a 5.1 channel configuration for loudspeaker presentation. Also more complex configurations like 7.1 channels and beyond are supported by some levels.
- Stereo output (specified in subclause 6.4.7.3).
  2-channel stereo output emulating a downmix of the multi-channel signal. This is only applicable in case of a 515 tree configuration because for a 525 tree configuration, the downmix is already a 2-channel stereo signal.
- Binaural output (specified in clause 7).
  Stereo output for headphones presentation, generated by integrated binaural rendering based on an HRTF data set provided by the user to the decoder.

The definition of the five levels of the Baseline MPEG Surround Profile is given in Table 4.

Table 4 — Levels of the Baseline MPEG Surround profile

Level	Tree configurations	Max. number output channels	Max. sampling rate [kHz]	Max. bandwidth residual coding [QMF bands]	Max. PCU High Quality decoder	Max. RCU High Quality decoder	Max. PCU Low Power decoder	Max. RCU Low Power decoder
1	515, 525 (Note 1) (Note 4)	2.0	48	0 (Note 2)	12	5	6	4
2	515, 525 (Note 4)	5.1	48	0 (Note 2)	25	15	12	11
3	515, 525 (Note 4)	5.1	48	64 (Note 3)	25	15	12	11
4	515, 525, 757, 727 (Note 4)	7.1	48	64 (Note 3)	34	21	17	15
5	515, 525, 757, 727, plus arbitrary tree extension	32 incl. LFE	96	64 (Note 3)	123 (max. 70 at 48 kHz sampling)	61 (max. 38 at 48 kHz sampling)	80 (max. 44 at 48 kHz sampling)	53 (max. 32 at 48 kHz sampling)

Note 1: This level provides a 2-channel stereo output.
Note 2: Residual coding data, if present in the bitstream, is not utilized, hence the residual decoding tool is not required.
Note 3: A low power decoder utilizes only residual coding data for the first 8 QMF bands, corresponding to approximately 2.7 kHz bandwidth.
Note 4: Arbitrary tree extension data, if present, is not utilized.

The following applies to all levels of the Baseline MPEG Surround Profile:

- Support of stereo output for 515 (subclause 6.4.7.3) is mandatory for all levels.
- Support of “Enhanced matrix mode” (subclause 6.2) is mandatory for all levels, and also in combination with binaural decoding.
- Support of binaural decoding in “Single slot parametric low complexity approach” (subclause 6.11.4.1 and subclause 6.11.4.2.2) and 3DaudioMode (subclause 6.11.5) is mandatory for all levels.
- Support for “Multi-slots HRTF convolution approach” (subclause 6.11.4.2.3) is mandatory for all levels of a high quality decoder (since this mode is not applicable in case of a low power decoder), and the complexity figures assume that a set of HRTF filters with a length of 128 samples (level 1) or 512 samples (levels 2 to 5) in the time domain is used.
- The complexity figures assume that normal decorrelators and not fractional delay decorrelators are used (subclause 6.6.2.3).
- The complexity figures do not include the complexity of the decoding process for the downmix signal itself that is input to the MPEG Surround decoder.
- When an MPEG Surround decoder is connected to an HE-AAC downmix decoder in the QMF domain, the combined PCU and RCU values need to be corrected to take into account that QMF synthesis and analysis filtering of the downmix signals is removed. A High Quality QMF bank has 1.5 PCU and 0.5 RCU, and a Low Power QMF bank has 1.0 PCU and 0.5 RCU. For a 2-channel downmix coded using the HE-AAC Profile at level 2 combined with the Baseline MPEG Surround Profile at level 2, the resulting complexity is:
  - High Quality: 9+25–2*2*1.5 = 28 PCU 10+15–2*2*0.5 = 23 RCU
  - Low Power: 7+12–2*2*1.0 = 15 PCU 8+11–2*2*0.5 = 17 RCU

5 Syntax

5.1 Payloads for Spatial Audio Coding

Table 5 — Syntax of SpatialSpecificConfig()

Syntax	No. of bits	Mnemonic
SpatialSpecificConfig()		
{		
bsSamplingFrequencyIndex;	**4**	**uimsbf**
if (bsSamplingFrequencyIndex == 0xf) {		
bsSamplingFrequency;	**24**	**uimsbf**
}		
bsFrameLength;	**7**	**uimsbf**
bsFreqRes;	**3**	**uimsbf**
bsTreeConfig;	**4**	**uimsbf**
bsQuantMode;	**2**	**uimsbf**
bsOneIcc;	**1**	**uimsbf**
bsArbitraryDownmix;	**1**	**uimsbf**
bsFixedGainSur;	**3**	**uimsbf**
bsFixedGainLFE;	**3**	**uimsbf**
bsFixedGainDMX;	**3**	**uimsbf**
bsMatrixMode;	**1**	**uimsbf**
bsTempShapeConfig;	**2**	**uimsbf**
bsDecorrConfig;	**2**	**uimsbf**
bs3DaudioMode;	**1**	**uimsbf**
for (i=0; i<numOttBoxes; i++) {		Note 1
OttConfig(i);		
}		
for (i=0; i<numTttBoxes; i++) {		Note 1
TttConfig(i);		
}		
if (bsTempShapeConfig == 2) {		
bsEnvQuantMode	**1**	**uimsbf**
}		
if (bs3DaudioMode) {		
bs3DaudioHRTFset;	**2**	**uimsbf**
if (bs3DaudioHRTFset==0) {		
ParamHRTFset();		
}		
}		
ByteAlign();		
SpatialExtensionConfig();		
}		
Note 1: numOttBoxes and numTttBoxes are defined by Table 40 dependent on bsTreeConfig.		

Table 6 — Syntax of OttConfig()

Syntax	No. of bits	Mnemonic

OttConfig(i)
{
  if (ottModeLfe[i]) {
  bsOttBands[i];
  }
  else {
  bsOttBands[i] = numBands;
  }
}
	5	Note 1
		uimsbf
		
		Note 2

Note 1: ottModeLfe[i] are defined by Table 40 dependent on bsTreeConfig. Note 2: numBands is defined in Table 39 and depends on bsFreqRes.		

Table 7 — Syntax of TttConfig()

Syntax	No. of bits	Mnemonic

TttConfig(i)
{
  bsTttDualMode[i];
  bsTttModeLow[i];
  if (bsTttDualMode[i]) {
  bsTttModeHigh[i];
  bsTttBandsLow[i];
  bsTttBandsHigh[i] = numBands;
  }
  else {
  bsTttBandsLow[i] = numBands;
  }
}
	1
	3
	
	3
	5
	
		uimsbf
		uimsbf
		
		uimsbf
		uimsbf
		Note 1
		
		Note 1

Note 1: numBands is defined in Table 39 and depends on bsFreqRes.		

Table 8 — Syntax of ParamHRTFset()

Syntax	No. of bits	Mnemonic

ParamHRTFset()
{
  bsHRTFfreqRes;
 	
  bsHRTFasymmetric;
  for (hc=0; hc<HRTFnumChan; hc++) {
  for (hb =0; hb <HRTFnumBands; hb ++) {
  bsHRTFlevelLeft[hc][hb];
  }
  if (bsHRTFasymmetric) {
  for (hb =0; hb <HRTFnumBands; hb ++) {
  bsHRTFlevelRight[hc][hb];
  }
  }
  bsHRTFphase[hc];
  if (bsHRTFphase[hc]) {
  for (hb =0; hb <HRTFnumPhase; hb ++) {
  bsHRTFphaseLR[hc][hb];
  }
	3
	
	1
	
	6
	
	6
	
	1
	
	6
		uimsbf
		
		uimsbf
		Note 1
		Note 2
		uimsbf
		
		Note 2
		uimsbf
		
		uimsbf
		
		Note 3
		uimsbf

}
bsHRTFicc[hc];
if (bsHRTFicc[hc]) {
for (hb =0; hb <HRTFnumBands; hb ++) {
bsHRTFiccLR[hc][hb];
}
}
}
}
	1
	3
		uimsbf
		Note 2
		uimsbf

Note 1: HRTFnumChan= 5. Note 2: HRTFnumBands is defined in Table 53 and depends on bsHRTFfreqRes. Note 3: HRTFnumPhase is defined in Table 53 and depends on bsHRTFfreqRes.		

Table 9 — Syntax of SpatialExtensionConfig()

Syntax	No. of bits	Mnemonic

SpatialExtensionConfig()
{
sacExtNum = 0;
while (BitsAvailable() >= 8) {
bsSacExtType;
sacExtType[sacExtNum] = bsSacExtType;
sacExtNum++;
cnt = **bsSacExtLen**;
if (cnt==15) {
cnt += **bsSacExtLenAdd**;
}
if (cnt==15+255) {
cnt += **bsSacExtLenAddAdd**;
}
bitsRead = SpatialExtensionConfigData(bsSacExtType)
nFillBits = 8*cnt-bitsRead;
bsFillBits;
}
}
}
	
	4
	
	4
	
	8
	
	16
	
	nFillBits
		
		Note 1
		uimsbf
		
		uimsbf
		
		uimsbf
		
		uimsbf
		
		Note 2
		
		bslbf

Note 1: The function BitsAvailable() returns the number of bits available to be read. Note 2: SpatialExtensionConfigData() returns the number of bits read.		

Table 10 — Syntax of SpatialExtensionConfigData(0)

Syntax	No. of bits	Mnemonic

SpatialExtensionConfigData(0)
{
bsResidualSamplingFrequencyIndex;
bsResidualFramesPerSpatialFrame;
for (i=0; i<numOttBoxes+numTttBoxes; i++) {
ResidualConfig(i);
}
}
	
	4
	2
		
		uimsbf
		uimsbf
		Note 1

Note 1: numOttBoxes and numTttBoxes are defined by Table 40 and depend on bsTreeConfig.		

Table 11 — Syntax of ResidualConfig()

Syntax	No. of bits	Mnemonic
ResidualConfig(i)		
{		
bsResidualPresent[i];	**1**	**uimsbf**
if (bsResidualPresent[i]) {		
bsResidualBands[i];	**5**	**uimsbf**
}		
}		

Table 12 — Syntax of SpatialExtensionConfigData(1)

Syntax	No. of bits	Mnemonic
SpatialExtensionConfigData(1)		
{		
bsArbitraryDownmixResidualSamplingFrequencyIndex;	**4**	**uimsbf**
bsArbitraryDownmixResidualFramesPerSpatialFrame;	**2**	**uimsbf**
bsArbitraryDownmixResidualBands;	**5**	**uimsbf**
}		

Table 13 — Syntax of SpatialExtensionConfigData(2)

Syntax	No. of bits	Mnemonic
SpatialExtensionConfigData(2)		
{		
TreeConfig();		
}		

Table 14 — Syntax of TreeConfig()

Syntax	No. of bits	Mnemonic
TreeConfig()		
{		
numOutChanAT = 0;		
numOttBoxesAT = 0;		
for (ch = 0; ch < numOutChan)		Note 1
tmpOpen = 1;		
idx = 0;		
while (tmpOpen > 0) {		
bsOttBoxPresent [ch][idx];	**1**	**uimsbf**
if (bsOttBoxPresent[ch][idx]) {		
numOttBoxesAT = numOttBoxesAT+1;		
tmpOpen = tmpOpen+1;		
}		
else {		
numOutChanAT = numOutChanAT + 1;		
tmpOpen = tmpOpen-1;		
}		
idx = idx + 1;		
}		
}		

for (i=0; i<numOttBoxesAT; i++) {		
defaultATD[i] = **bsOttDefaultCld**;	**1**	**uimsbf**
ottModeLfeAT[i] = **bsOttModeLfe**;	**1**	**uimsbf**
if (ottModeLfeAT[i]) {		
bsOttBandsAT[i] = **bsOttBands**;	**5**	**uimsbf**
}		
else {		
bsOttBandsAT[i] = numBands;		Note 2
}		
}		
for (i=0; i<numOutChanAT; i++) {		
bsOutputChannelPos[i]	**5**	**uimsbf**
}		
}		
Note 1: numOutChan is defined in Table 40 and depends on bsTreeConfig.		
Note 2: numBands is defined in Table 39 and depends on bsFreqRes.		

Table 15 — Syntax of SpatialFrame()

Syntax	No. of bits	Mnemonic
SpatialFrame()		
{		
FramingInfo();		
bsIndependencyFlag;	**1**	**uimsbf**
OttData();		
TttData();		
SmgData();		
TempShapeData();		
if (bsArbitraryDownmix != 0) {		
ArbitraryDownmixData();		
}		
ByteAlign();		
SpatialExtensionFrame();		
}		

Table 16 — Syntax of FramingInfo()

Syntax	No. of bits	Mnemonic

FramingInfo()
{
  bsFramingType;
  bsNumParamSets;
  if (bsFramingType) {
  for (ps=0; ps<numParamSets; ps++) {
  if(ps==0){
  bsParamSlot[0];
 	
  else{
  bsDiffParamSlot[ps];
 	
  bsParamSlot[ps] = bsParamSlot[ps-1] + bsDiffParamSlot[ps] + 1;
  }
  }
  }
}
	
	1
	3
	
	nBitsPara mSlot(0)
	
	nBitsPara mSlot(ps)
		
		uimsbf
		uimsbf
		
		Note 1
		
		uimsbf Note 2
		
		uimsbf Note 2

Note 1: numParamSets is defined by numParamSets = bsNumParamSets + 1. Note 2: nBitsParamSlot(ps) is defined according to nBitsParamSlot(0) = ceil($\log_2$(numSlots–numParamSets+1)) and nBitsParamSlot(ps) = ceil($\log_2$(numSlots–numParamSets+ps–bsParamSlot[ps–1])) for 0 < ps < numParamSets.		

Table 17 — Syntax of OttData()

Syntax	No. of bits	Mnemonic

OttData()
{
  for (i=0; i<numOttBoxes; i++) {
  EcData(CLD, i, 0, bsOttBands[i]);
  }
  if (bsOneIcc) {
  EcData(ICC, 0, 0, numBands);
  }
  else {
  for (i=0; i<numOttBoxes; i++) {
  if (!ottModeLfe[i]) {
  EcData(ICC, i, 0, bsOttBands[i]);
  }
  }
  }
}
		
		Note 1
		
		Note 2
		
		Note 1
		Note 1

Note 1: numOttBoxes and ottModeLfe[i] are defined by Table 40 and depends on bsTreeConfig. Note 2: numBands is defined in Table 39 and depends on bsFreqRes.		

Table 18 — Syntax of TttData ()

Syntax	No. of bits	Mnemonic

TttData()
{
  tttOff = numOttBoxes;
  for (i=0; i<numTttBoxes; i++) {
  if (bsTttModeLow[i] < 2) {
  EcData(CPC, tttOff+4*i, 0, bsTttBandsLow[i]);
  EcData(CPC, tttOff+4*i+1, 0, bsTttBandsLow[i]);
  EcData(ICC, tttOff+4*i, 0, bsTttBandsLow[i]);
  }
  else {
  EcData(CLD, tttOff+4*i, 0, bsTttBandsLow[i]);
  EcData(CLD, tttOff+4*i+1, 0, bsTttBandsLow[i]);
  }
  if (bsTttDualMode[i]) {
  if (bsTttModeHigh[i] < 2) {
  EcData(CPC, tttOff+4*i+2, bsTttBandsLow[i], bsTttBandsHigh[i]);
  EcData(CPC, tttOff+4*i+3, bsTttBandsLow[i], bsTttBandsHigh[i]);
  EcData(ICC, tttOff+4*i+2, bsTttBandsLow[i], bsTttBandsHigh[i]);
  }
  else {
  EcData(CLD, tttOff+4*i+2, bsTttBandsLow[i], bsTttBandsHigh[i]);
  EcData(CLD, tttOff+4*i+3, bsTttBandsLow[i], bsTttBandsHigh[i]);
  }
  }
  }
}
		Note 1

Note 1: numOttBoxes is defined by Table 40 and depends on bsTreeConfig.		

Table 19 — Syntax of SmgData()

Syntax	No. of bits	Mnemonic

SmgData()
{
  for (ps=0; ps<numParamSets; ps++) {
  bsSmoothMode[ps];
  if (bsSmoothMode[ps] >= 2) {
  bsSmoothTime[ps];
  }
  if (bsSmoothMode[ps] == 3) {
  bsFreqResStrideSmg[ps];
  dataBands = (numBands-1)/pbStride+1;
  for (pg=0; pg<dataBands; pg++) {
  bsSmgData[ps][pg];
  }
  }
  }
}
	
	2
	
	2
	
	2
	
	1
	
		Note 1
		uimsbf
		
		uimsbf
		
		uimsbf
		Note 2
		
		uimsbf

Note 1: numParamSets is defined by numParamSets = bsNumParamSets + 1. Note 2: pbStride is defined in Table 70 and depends on bsFreqResStrideSmg[]. Furthermore the division shall be interpreted as ANSI C integer division. numBands is defined in Table 39 and depends on bsFreqRes.		

Table 20 — Syntax of TempShapeData()

Syntax	No. of bits	Mnemonic

TempShapeData()
{
  if (bsTempShapeConfig != 0) {
  bsTempShapeEnable;
	1	**uimsbf**

if (bsTempShapeEnable) { for (ch=0; ch<numTempShapeChan; ch++) {		Note 1
bsTempShapeEnableChannel[ch];	**1**	**uimsbf**

}
  if (bsTempShapeConfig == 2) {
  EnvelopeReshapeHuff(bsTempShapeEnableChannel);
  }
  else if (bsTempShapeConfig == 3) {
  /* reserved */
  }
  }
  }
}		

Note 1: numTempShapeChan is defined by Table 40 and depends on bsTempShapeConfig and bsTreeConfig.		

Table 21 — Syntax of EnvelopeReshapeHuff()

Syntax	No. of bits	Mnemonic

EnvelopeReshapeHuff()
{
  for (ch=0; ch<numTempShapeChan; ch++) {
		Note 1

if (bsTempShapeEnableChan[ch]) {
  numValRcvd = 0;
  while (numValRcvd < numTimeSlots) {
		Note 2

(erVal, erLen) = 2Dhuff_dec(hcod2D_EnvRes, **bsCodeW**);	**1..x**	**vlclbf**

for (k=numValRcvd; k<numValRcvd+erLen; k++) {
  bsEnvShapeData[ch][k] = erVal;
  }
  numValRcvd += erLen;
  }
  }
  }
}		

Note 1: numTempShapeChan is defined by Table 40 and depends on bsTempShapeConfig and bsTreeConfig. Note 2: numSlots is defined by numSlots = bsFrameLength + 1.		

Table 22 — Syntax of ArbitraryDownmixData()

Syntax	No. of bits	Mnemonic

ArbitraryDownmixData()
{
  adOff = numOttBoxes+4*numTttBoxes;
  for (ic=0; ic<numInChan; ic++) {
  EcData(CLD, adOff+ic, 0, numBands);
  }
}
		Note 1

Note 1: numBands is defined in Table 39 and depends on bsFreqRes.		

Table 23 — EcData()

Syntax	No. of bits	Mnemonic

EcData(dataType, paramIdx, startBand, stopBand)
{
  dataSets = 0;
  for (ps=0; ps<numParamSets; ps++) {
  bsXXXdataMode[paramIdx][ps];
  if (bsXXXdataMode[paramIdx][ps] == 3) {
  dataSets++;
  }
  }
  setIdx = 0;
  while (setIdx < dataSets) {
  bsDataPairXXX[paramIdx][setIdx];
  bsQuantCoarseXXX[paramIdx][setIdx];
  bsFreqResStrideXXX[paramIdx][setIdx];
  dataBands = (stopBand-startBand-1)/pbStride+1;
  EcDataPair(dataType, paramIdx, setIdx, dataBands,
  bsDataPairXXX[paramIdx][setIdx],
  bsQuantCoarseXXX[paramIdx][setIdx]);
  if (bsDataPairXXX[paramIdx][setIdx]) {
  bsXXXQuantCoarse[paramIdx][setIdx+1] =
  bsXXXQuantCoarse[paramIdx][setIdx];
  bsFreqResStrideXXX[paramIdx][setIdx+1] =
  bsFreqResStrideXXX[paramIdx][setIdx];
  }
  setIdx += bsDataPairXXX[paramIdx][setIdx]+1;
  }
  startBandXXX[paramIdx] = startBand;
  stopBandXXX[paramIdx] = stopBand;
}
	2
	
	1
	1
	2
		Note 1
		
		Note 2
		uimsbf
		
		uimsbf
		uimsbf
		uimsbf
		Note 3

Note 1: XXX is to be replaced by the value of dataType (CLD, ICC, CPC, ATD). Note 2: numParamSets is defined by numParamSets = bsNumParamSets + 1. Note 3: pbStride is defined in Table 70 and depends on bsFreqResStride[][]. Furthermore the division shall be interpreted as ANSI C integer division.		

Table 24 — Syntax of EcDataPair()

Syntax	No. of bits	Mnemonic
EcDataPair(dataType, paramIdx, setIdx, dataBands, pairFlag, coarseFlag)		Note 1
{		
mixedTimePair_flag = 0;		
bsPcmCodingXXX[paramIdx][setIdx];	**1**	**uimsbf**
bsPilotCodingXXX[paramIdx][setIdx] = 0;		
if (bsPcmCodingXXX[paramIdx][setIdx] && dataBands>4) {		
bsPilotCodingXXX[paramIdx][setIdx];	**1**	**uimsbf**
}		
if (bsPcmCoding[paramIdx][setIdx] &&!bsPilotCodingXXX[paramIdx][setIdx]) {		
if (coarseFlag) {		
numQuantSteps = numQuantStepsXXXCoarse;		Note 2
Else {		
numQuantSteps = numQuantStepsXXXFine;		Note 2
}		
aaDataPair = GroupedPcmData(dataType, pairFlag, numQuantSteps, dataBands);		
}		
else {		
allowDiffTimeBack = (!bsIndependencyFlag) || (setIdx>0);		
(aaDataPairMsbDiff, aPgOffset, pilot, mixedTimePair_flag) = DiffHuffData(dataType, pairFlag, allowDiffTimeBack, bsPilotCodingXXX[paramIdx][setIdx], dataBands);		
aaDataPairLsb[0] = LsbData(dataType, coarseFlag, dataBands);		
if (pairFlag) {		
aaDataPairLsb[1] = LsbData(dataType, coarseFlag, dataBands);		
}		
}		
/* copy information read by EcDataPair() and its subfunctions into non-ambiguous variables for later delta decoding etc. */		
bsDiffTypeXXX[paramIdx][setIdx] = bsDiffType[0];		
bsDiffTimeDirectionXXX[paramIdx][setIdx] = bsDiffTimeDirection[0];		
bsXXXpilot[paramIdx][setIdx] = pilot;		
mixedTimePairXXX[paramIdx][setIdx] = mixedTimePair_flag;		
if (pairFlag) {		
bsDiffTypeXXX[paramIdx][setIdx+1] = bsDiffType[1];		
bsDiffTimeDirectionXXX[paramIdx][setIdx+1] = bsDiffTimeDirection[1];		
bsPcmCodingXXX[paramIdx][setIdx+1] = bsPcmCodingXXX[paramIdx][setIdx];		
bsXXXpilot[paramIdx][setIdx+1] = pilot;		
bsPilotCodingXXX[paramIdx][setIdx+1] = bsPilotCodingXXX[paramIdx][setIdx];		
mixedTimePairXXX[paramIdx][setIdx+1] = mixedTimePair_flag;		
}		
for (pg=0; pg<dataBands; pg++) {		
if (bsPcmCodingXXX[paramIdx][setIdx] && !bsPilotCodingXXX[paramIdx][setIdx]) {		
bsXXXpcm[paramIdx][setIdx][pg] = aaDataPair[0][pg];		
else {		
bsXXXmsbDiff[paramIdx][setIdx][pg] = aaDataPairMsbDiff[0][pg];		
bsXXXlsb[paramIdx][setIdx][pg] = aaDataPairLsb[0][pg];		
}		

```
        if (pairFlag) {
            if (bsPcmCodingXXX[paramIdx][setIdx+1] &&
            !bsPilotCodingXXX[paramIdx][setIdx+1]) {
                bsXXXpcm[paramIdx][setIdx+1][pg] = aaDataPair[1][pg];
            }
            else {
                bsXXXmsbDiff[paramIdx][setIdx+1][pg] =aaDataPairMsbDiff[1][pg];
                bsXXXlsb[paramIdx][setIdx+1][pg] = aaDataPairLsb[1][pg];
            }
        }
    }
}
```

Note 1: XXX is to be replaced by the value of dataType. (CLD, ICC, CPC, ATD).
Note 2: numQuantStepsXXXCoarse and numQuantStepsXXXFine is defined in Table 71 and depends on dataType.

Table 25 — Syntax of GroupedPcmData()

Syntax	No. of bits	Mnemonic

GroupedPcmData(dataType, pairFlag, numQuantSteps, dataBands)
{
numPcmDataBands = dataBands;
if (pairFlag) {
numPcmDataBands *= 2;
}
 	
for (i=0 ; i<numPcmDataBands ; i+= maxGrpLen) {
grpLen = min(maxGrpLen, numPcmDataBands - i);
nGrpBits = ceil(grpLen*log$_2$(numQuantSteps));
bsPcmWord;
for (j = 0; j < grpLen; j++) {
idx = i + (grpLen–1) – j;
pcmValue = bsPcmWord % numQuantSteps;
if (pairFlag) {
aaDataPair[idx%2][idx/2] = pcmValue;
}
else {
aaDataPair[0][idx] = pcmValue;
}
bsPcmWord /= numQuantSteps;
}
}
 	
return (aaDataPair);
}
	
	nGrpBits
	
		Note 1
		
		uimsbf
		
		Note 2
		Note 2

Note 1: maxGrpLen is defined in Table 72 and depends on numQuantSteps.
Note 2: % denotes the modulo operator (ANSI C integer math) and returns the remainder of the division.

Table 26 — Syntax of DiffHuffData()

Syntax	No. of bits	Mnemonic
DiffHuffData(dataType, pairFlag, allowDiffTimeBackFlag, pilotCodingFlag, dataBands)		
{		
mixedTimePair_flag = 0;		
bsDiffType[0] = DIFF_FREQ;		
bsDiffType[1] = DIFF_FREQ;		
if (pilotCodingFlag) {		
pilot = 1Dhuff_dec(hcodPilot_XXX, **bsCodeW**);	**1..x**	**vlclbf** Note1,Note2
}		
else {		
if (pairFlag || allowDiffTimeBackFlag) {		
bsDiffType[0];	**1**	**Uimsbf**
}		
if (pairFlag && ((bsDiffType[0] == DIFF_FREQ) || allowDiffTimeBackFlag)) {		
bsDiffType[1];	**1**	**Uimsbf**
}		
}		
bsCodingScheme;	**1**	**Uimsbf**
if (bsCodingScheme == HUFF_1D) {		
(aaHuffData[0]) = HuffData1D(dataType, aDiffType[0], pilotCodingFlag, dataBands);		
if (pairFlag) {		
(aaHuffData[1]) = HuffData1D(dataType, aDiffType[1], pilotCodingFlag, dataBands);		
}		
}		
else { /* HUFF_2D */		
if (pairFlag) {		
bsPairing;	**1**	**Uimsbf**
}		
else {		
bsPairing = FREQ_PAIR;		
}		
if (bsPairing == FREQ_PAIR) {		
(aaHuffData[0]) = HuffData2DFreqPair(dataType, aDiffType[0], pilotCodingFlag, dataBands);		
if(pairFlag) {		
(aaHuffData[1]) = HuffData2DFreqPair(dataType, aDiffType[1], pilotCodingFlag, dataBands);		
}		
}		
else { /* TIME_PAIR */		
(aaHuffData) = HuffData2DtimePair(dataType, aDiffType, pilotCodingFlag, dataBands);		
if (bsDiffType[0] != bsDiffType[1]) {		
mixedTimePair_flag = 1;		
}		

}
}
 	
/* Inverse differential coding */
if ((bsDiffType[0] == DIFF_TIME) || (bsDiffType[1] == DIFF_TIME)) {
if (!allowDiffTimeBackFlag && (bsDiffType[0] == DIFF_TIME)) {
bsDiffTimeDirection[0] = FORWARDS;
}
else if (!pairFlag || (pairFlag && (bsDiffType[1] == DIFF_TIME))) {
bsDiffTimeDirection[0] = BACKWARDS;
}
else {
bsDiffTimeDirection[0];
	1	**Uimsbf**

}
if (pairFlag) {
bsDiffTimeDirection[1] = BACKWARDS;
}
}
 	
return (aaHuffData, aPgOffset, pilot, mixedTimePair_flag);
}		

Note 1: XXX is to be replaced by the value of dataType (CLD, ICC or CPC, ATD). Note 2: 1Dhuff_dec() is defined in Annex A.1.		

Table 27 — Syntax of HuffData1D()

Syntax	No. of bits	Mnemonic

HuffData1D(dataType, diffType, pilotCodingFlag, dataBands)
{
pgOffset = 0;
 	
if (diffType == DIFF_FREQ &&!pilotCodingFlag) {
aHuffData1D[0] = 1Dhuff_dec(hcodFirstBand_XXX, **bsCodeW**);
	1..x	**vlclbf**
		Note1,3

pgOffset = 1;
}
 	
for (i=pgOffset; i<dataBands; i++) {
aHuffData1D[i] = 1Dhuff_dec(hcod1D_XXX_YY, **bsCodeW**);
	1..x	**vlclbf**
		Note1,2,3

if (aHuffData1D[i] != 0) {
bsSign;
if (bsSign) {
aHuffData1D[i] = -aHuffData1D[i];
}
}
}
 	
return (aHuffData1D);
}
	1	**Uimsbf**

Note 1: XXX is to be replaced by the value of dataType (CLD, ICC, CPC, ATD). Note 2: YY is to be replaced by “PC”, “DF”, or “DT”, depending on the value of pilotCodingFlag and diffType. Note 3: 1Dhuff_dec() is defined in Annex A.1.		

Table 28 — Syntax of HuffData2DFreqPair()

Syntax	No. of bits	Mnemonic
HuffData2DFreqPair(dataType, diffType, pilotCodingFlag, dataBands)		
{		
LavIdx = 1Dhuff_dec(hcodLavIdx, **bsCodeW**);	**1..3**	**Vlclbf**
lav = lavTabXXX[LavIdx];		Note 1
pgOffset = 0;		
if (diffType == DIFF_FREQ &&!pilotCodingFlag) {		
aHuffData2D[0] = 1Dhuff_dec(hcodFirstBand_XXX, **bsCodeW**);	**1..x**	**vlclbf** Note6
pgOffset = 1;		
}		
escapeCode = hcod2D_XXX_YY_FP_LL_escape;		Note2,3,4,5
/* specific escape code belonging to this Huffman table */		
escCntr = 0;		
for (i=pgOffset; i<dataBands; i+=2) {		
(aTmp[0], aTmp[1]) = 2Dhuff_dec(hcod2D_XXX_YY_FP_LL, **bsCodeW**);	**1..x**	**vlclbf,** Note3,4,5,6
if (bsCodeWord != escapeCode) {		
aTmpSym = SymmetryData(aTmp);		
aHuffData2D[i] = aTmpSym[0];		
aHuffData2D[i+1] = aTmpSym[1];		
}		
else {		
aEscList[escCntr++] = i;		
}		
}		
if (escCntr > 0) {		
aaEscData = GroupedPcmData(dataType, 1, 2*lav+1, escCntr);		
for (i=0; i<escCntr; i++) {		
aHuffData2D[aEscList[i]] = aaEscData[0][i] - lav;		
aHuffData2D[aEscList[i]+1] = aaEscData[1][i] - lav;		
}		
}		
if ((dataBands-pgOffset) % 2) {		Note 7
aHuffData2D[dataBands-1] = 1Dhuff_dec(hcod1D_XXX_YY, **bsCodeW**);	**1..x**	**vlclbf,** Note3,4,6
if (aHuffData2D[dataBands-1] != 0) {		
bsSign;	**1**	**Uimsbf**
if (bsSign) {		
aHuffData2D[dataBands-1] = -aHuffData2D[dataBands-1];		
}		
}		
}		
return (aHuffData2D);		
}		

Note 1: lavTabXXX is defined in Table 76.
Note 2: The escape code tables are defined in Table A.8, Table A.9, and Table A.10. For some Huffman tables no escape code is needed since all possible values are covered by the huffman table.
Note 3: XXX is to be replaced by the value of dataType (CLD, ICC, CPC, ATD).
Note 4: YY is to be replaced by "PC", "DF", or "DT", depending on the value of pilotCodingFlag and diffType.
Note 5: LL is to be replaced by the value of lav.
Note 6: 1Dhuff_dec() and 2Dhuff_dec() are defined in Annex A.1.
Note 7: % denotes the modulo operator (ANSI C integer math) and returns the remainder of the division.

Table 29 — Syntax of HuffData2DTimePair()

Syntax	No. of bits	Mnemonic
HuffData2DTimePair(dataType, aDiffType, pilotCodingFlag, dataBands)		
{		
LavIdx = 1Dhuff_dec(hcodLavIdx, **bsCodeW**);	**1..3**	**VlcIbf**
lav = lavTabXXX[LavIdx];		Note 1
pgOffset = 0;		
if (((aDiffType[0] == DIFF_FREQ) || (aDiffType[1] == DIFF_FREQ))		
&&!pilotCodingFlag) {		
aaHuffData2D[0][0] = 1Dhuff_dec(hcodFirstBand_XXX, **bsCodeW**);	**1..x**	**vlcIbf**, Note3,6
aaHuffData2D[1][0] = 1Dhuff_dec(hcodFirstBand_XXX, **bsCodeW**);	**1..x**	**vlcIbf**, Note3,6
pgOffset = 1;		
}		
escapeCode = hcod2D_XXX_YY_TP_LL_escape;		Note2,3,4,5
/* specific escape code belonging to this Huffman table */		
escCntr = 0;		
if ((aDiffType[0] == DIFF_TIME) || (aDiffType[1] == DIFF_TIME)) {		
diffType = DIFF_TIME;		
}		
Else {		
diffType = DIFF_FREQ;		
}		
for (i=pgOffset; i<dataBands; i++) {		
(aTmp[0], aTmp[1]) = 2Dhuff_dec(hcod2D_XXX_YY_TP_LL, **bsCodeW**);	**1..x**	**vlcIbf**, Note3,4,5,6
if (bsCodeW != escapeCode) {		
aTmpSym = SymmetryData(aTmp);		
aaHuffData2D[0][i] = aTmpSym[0];		
aaHuffData2D[1][i] = aTmpSym[1];		
}		
else {		
aEscList[escCntr++] = i;		
}		
}		

```
    if ( escCntr > 0 ) {
        aaEscData = GroupedPcmData(dataType, 1, 2*lav+1, escCntr);
        for ( i=0; i<escCntr; i++ ) {
            aaHuffData2D[0][aEscList[i]] = aaEscData[0][i] - lav;
            aaHuffData2D[1][aEscList[i]] = aaEscData[1][i] - lav;
        }
    }

    return (aaHuffData2D);
}
```

Note 1: lavTabXXX is defined in Table 76.
Note 2: The escape code tables are defined in Table A.8, Table A.9, and Table A.10. For some Huffman tables no escape code is needed since all possible values are covered by the huffman table.
Note 3: XXX is to be replaced by the value of dataType (CLD, ICC, CPC, ATD).
Note 4: YY is to be replaced by “PC”, “DF”, or “DT”, depending on the value of pilotCodingFlag and diffType.
Note 5: LL is to be replaced by the value of lav.
Note 6: 1Dhuff_dec() and 2Dhuff_dec() are defined in Annex A.1.

Table 30 — Syntax of SymmetryData()

Syntax	No. of bits	Mnemonic

SymmetryData(aDataPair)
{
  sumVal = aDataPair[0] + aDataPair[1];
  diffVal = aDataPair[0] - aDataPair[1];
  if (sumVal > lav) {
  aDataPair[0] = (2*lav+1) - sumVal;
  aDataPair[1] = - diffVal;
  }
  else {
  aDataPair[0] = sumVal;
  aDataPair[1] = diffVal;
  }
  if (aDataPair[0] + aDataPair[1] != 0) {
  bsSymBit[0];
	1	**uimsbf**

if (bsSymBit[0]) {
  aDataPair[0] = - aDataPair[0];
  aDataPair[1] = - aDataPair[1];
  }
  }
  if (aDataPair[0] - aDataPair[1] != 0) {
  bsSymBit[1];
	1	**uimsbf**

if (bsSymBit[1]) {
  tmpVal = aDataPair[0];
  aDataPair[0] = aDataPair[1];
  aDataPair[1] = tmpVal;
  }
  }
}		

Table 31 — Syntax of LsbData()

Syntax	No. of bits	Mnemonic

LsbData(dataType, coarseFlag, dataBands)
{
for(i=0; i<dataBands; i++) {
bsLsb = 0;
if (dataType == CPC && !coarseFlag) {
bsLsb;
	1	**uimsbf**

}
aDataOut[i] =bsLsb;
}
return (aDataOut);
}		

Table 32 — Syntax of SpatialExtensionFrame()

Syntax	No. of bits	Mnemonic

SpatialExtensionFrame()
{
for (ec=0; ec<sacExtNum; ec++) {
if (sacExtType[ec]<12) {
cnt = **bsSacExtLen**;
	8	**uimsbf**

if (cnt==255) { cnt += **bsSacExtLenAdd**;	**16**	**uimsbf**
} bitsRead = SpatialExtensionFrameData(sacExtType[ec])		Note 1
nFillBits = 8*cnt-bitsRead; **bsFillBits**;	**nFillBits**	**bslbf**
} } }		
Note 1: SpatialExtensionFrameData() returns the number of bits read.		

Table 33 — Syntax of SpatialExtensionFrameData(0)

Syntax	No. of bits	Mnemonic

SpatialExtensionFrameData(0)
{
ResidualData();
}		

Table 34 — Syntax of ResidualData()

Syntax	No. of bits	Mnemonic

ResidualData()
{
for (i=0; i<numOttBoxes+numTttBoxes; i++) {
if (bsResidualPresent[i]) {
if (i<numOttBoxes) {
for (ps=0; ps<numParamSets; ps++) {
		Note 2

bsIccDiffPresent[i][ps];	**1**	**Uimsbf**

if (bsIccDiffPresent[i][ps]) {
for (pb=0; pb<bsResidualBands[i]; pb++) {
IccDiff[i][ps][pb] =
	1..7	**vlclbf**

1Dhuff_dec(hcod1D_ICC_Diff,**bsCodeW**);		Note 3

}
}
}
}
tempExtraFrame=numSlots/(bsResidualFramesPerSpatialFrame+1);
		Note 4

for (rf=0; rf<bsResidualFramesPerSpatialFrame; rf++) individual_channel_stream(0);		Note 1

if (window_sequence == EIGHT_SHORT_SEQUENCE) &&
((tempExtraFrame == 18) || (tempExtraFrame == 24) ||
(tempExtraFrame == 30)) {
		Note 5

individual_channel_stream(0);		Note 1
} } } } }		
Note1: individual_channel_stream(0) according to MPEG-2 AAC Low Complexity profile bitstream syntax described in subclause 6.3 of ISO/IEC 13818-7. Note 2: numParamSets is defined by numParamSets = bsNumParamSets + 1. Note 3: 1Dhuff_dec() is defined in Annex A.1. Note 4: numSlots is defined by numSlots = bsFrameLength +1. Furthermore the division shall be interpreted as ANSI C integer division. Note 5: individual_channel_stream(0) determines the value of window_sequence.		

Table 35 — Syntax of SpatialExtensionFrameData(1)

Syntax	No. of bits	Mnemonic

SpatialExtensionDataFrame(1)
{
ArbitraryDownmixResidualData();
}		

Table 36 — Syntax of ArbitraryDownmixResidualData()

Syntax	No. of bits	Mnemonic
ArbitraryDownmixResidualData()		
{		
resFrameLength = numSlots /		Note 1
(bsArbitraryDownmixResidualFramesPerSpatialFrame + 1);		
for (i = 0; i < numAacEl; i++) {		Note 2
bsArbitraryDownmixResidualAbs[i]	**1**	**Uimsbf**
bsArbitraryDownmixResidualAlphaUpdateSet[i]	**1**	**Uimsbf**
for (rf = 0; rf < bsArbitraryDownmixResidualFramesPerSpatialFrame + 1;		
rf++)		
if (AacEl[i] == 0) {		Note 3
individual_channel_stream(0);		Note 4
else{		
channel_pair_element();		Note 5
}		
if (window_sequence == EIGHT_SHORT_SEQUENCE) &&		Note 6
((resFrameLength == 18) || (resFrameLength == 24) ||		
(resFrameLength == 30)) {		
if (AacEl[i] == 0) {		
individual_channel_stream(0);		Note 4
else{		
channel_pair_element();		Note 5
}		
}		
}		
}		
}		

Note 1: numSlots is defined by numSlots = bsFrameLength + 1. Furthermore the division shall be interpreted as ANSI C integer division.
Note 2: numAacEl indicates the number of AAC elements in the current frame according to Table 81.
Note 3: AacEl indicates the type of each AAC element in the current frame according to Table 81.
Note 4: individual_channel_stream(0) according to MPEG-2 AAC Low Complexity profile bitstream syntax described in subclause 6.3 of ISO/IEC 13818-7.
Note 5: channel_pair_element(); according to MPEG-2 AAC Low Complexity profile bitsream syntax described in subclause 6.3 of ISO/IEC 13818-7. The parameter common_window is set to 1.
Note 6: The value of window_sequence is determined in individual_channel_stream(0) or channel_pair_element().

Table 37 — Syntax of SpatialExtensionFrameData(2)

Syntax	No. of bits	Mnemonic
SpatialExtensionFrameData(2)		
{		
ArbitraryTreeData();		
}		

Table 38 — Syntax of ArbitraryTreeData()

Syntax	No. of bits	Mnemonic

ArbitraryTreeData()
{
for (i=0; i<numOttBoxesAT; i++) {
EcData(ATD, i, 0, bsOttBandsAT[i]);
}
}
		Note 1

Note 1: numOttBoxesAT is defined by TreeConfig().		

5.2 Definition

ByteAlign() Up to 7 fill bits to achieve byte alignment with respect to the beginning of the syntactic element in which ByteAlign() occurs.

SpatialSpecificConfig() Syntactic element that contains the SAC configuration data (header).

bsSamplingFrequencyIndex
see ISO/IEC 14496-3, subclause 1.6.3.4.

bsSamplingFrequency see ISO/IEC 14496-3, subclause 1.6.3.3.

bsFrameLength Defines the number of time slots in a spatial frame according to:
numSlots = **bsFrameLength** + 1.

bsFreqRes Defines the number of parameter bands according to:

Table 39 — bsFreqRes

bsFreqRes	**numBands**
0	Reserved
1	28
2	20
3	14
4	10
5	7
6	5
7	4

bsTreeConfig Defines the tree configuration according to:

Table 40 — bsTreeConfig

bsTreeConfig	Meaning
0	5151 configuration numOttBoxes = 5 defaultCld[0] = 1 defaultCld[1] = 1 defaultCld[2] = 0 defaultCld[3] = 0 defaultCld[4] = 1 defaultCld[5] = 0 ottModeLfe[0] = 0 ottModeLfe[1] = 0 ottModeLfe[2] = 0 ottModeLfe[3] = 0 ottModeLfe[4] = 1 numTttBoxes = 0 numInChan = 1 numOutChan = 6 output channel ordering: L, R, C, LFE, Ls, Rs
1	5152 configuration numOttBoxes = 5 defaultCld[0] = 1 defaultCld[1] = 0 defaultCld[2] = 1 defaultCld[3] = 1 defaultCld[4] = 1 defaultCld[5] = 0 ottModeLfe[0] = 0 ottModeLfe[1] = 0 ottModeLfe[2] = 1 ottModeLfe[3] = 0 ottModeLfe[4] = 0 numTttBoxes=0 numInChan = 1 numOutChan = 6 output channel ordering: L, Ls, R, Rs, C, LFE
2	525 configuration numOttBoxes = 3 defaultCld[0] = 1 defaultCld[1] = 1 defaultCld[2] = 1 defaultCld[3] = 1 defaultCld[4] = 0 defaultCld[5] = 1 defaultCld[6] = 0 defaultCld[7] = 0 defaultCld[8] = 0 ottModeLfe[0] = 1 ottModeLfe[1] = 0 ottModeLfe[2] = 0 numTttBoxes=1 numInChan = 2 numOutChan = 6 output channel ordering: L, Ls, R, Rs, C, LFE

3	7271 configuration (5/2.1) numOttBoxes = 5 defaultCld[0] = 1 defaultCld[1] = 1 defaultCld[2] = 1 defaultCld[3] = 1 defaultCld[4] = 1 defaultCld[5] = 1 defaultCld[6] = 0 defaultCld[7] = 1 defaultCld[8] = 0 defaultCld[9] = 0 defaultCld[10] = 0 ottModeLfe[0] = 1 ottModeLfe[1] = 0 ottModeLfe[2] = 0 ottModeLfe[3] = 0 ottModeLfe[4] = 0 numTttBoxes = 1 numInChan = 2 numOutChan = 8 output channel ordering: L, Lc, Ls, R, Rc, Rs, C, LFE
4	7272 configuration (3/4.1) numOttBoxes = 5 defaultCld[0] = 1 defaultCld[1] = 1 defaultCld[2] = 1 defaultCld[3] = 1 defaultCld[4] = 1 defaultCld[5] = 1 defaultCld[6] = 0 defaultCld[7] = 1 defaultCld[8] = 0 defaultCld[9] = 0 defaultCld[10] = 0 ottModeLfe[0] = 1 ottModeLfe[1] = 0 ottModeLfe[2] = 0 ottModeLfe[3] = 0 ottModeLfe[4] = 0 numTttBoxes = 1 numInChan = 2 numOutChan = 8 output channel ordering: L, Lsr, Ls, R, Rsr, Rs, C, LFE

5	7571 configuration (5/2.1) numOttBoxes = 2 defaultCld[0] = 1 defaultCld[1] = 1 defaultCld[2] = 0 defaultCld[3] = 0 defaultCld[4] = 0 defaultCld[5] = 0 defaultCld[6] = 0 defaultCld[7] = 0 ottModeLfe[0] = 0 ottModeLfe[1] = 0 numTttBoxes = 0 numInChan = 6 numOutChan = 8 output channel ordering: L, Lc, Ls, R, Rc, Rs, C, LFE
6	7572 configuration (3/4.1) numOttBoxes = 2 defaultCld[0] = 1 defaultCld[1] = 1 defaultCld[2] = 0 defaultCld[3] = 0 defaultCld[4] = 0 defaultCld[5] = 0 defaultCld[6] = 0 defaultCld[7] = 0 ottModeLfe[0] = 0 ottModeLfe[1] = 0 numTttBoxes = 0 numInChan = 6 numOutChan = 8 output channel ordering: L, Lsr, Ls, R, Rsr, Rs, C, LFE
7…15	Reserved

bsQuantMode Defines quantization and CLD energy-dependent quantization (EdQ) according to:

Table 41 — bsQuantMode

bsQuantMode	Meaning
0	default fine quantization for CLD, ICC, CPC
1	as above, but with first CLD EdQ curve
2	as above, but with second CLD EdQ curve
3	Reserved

bsOneIcc Indicates if only a single left/right ICC parameter is conveyed common to all OTT boxes.

bsArbitraryDownmix Indicates the presence of arbitrary down-mix gains according to Table 42.

Table 42 — bsArbitraryDownmix

bsArbitraryDownmix	Arbitrary down-mix gains
0	Not present
1	Present

bsFixedGainsSur Defines the gains used for the surround channels according to:

Table 43 — bsFixedGainsSur

bsFixedGainSur	Surround gain
0	1
1	2^(1/4)
2	2^(2/4)
3	2^(3/4)
4	2^(4/4)
5..7	reserved

bsFixedGainsLFE Defines the gains used for the LFE channels according to:

Table 44 — bsFixedGainsLFE

bsFixedGainLFE	LFE gain
0	1
1	10^(1/2)
2	10^(2/2)
3	10^(3/2)
4	10^(4/2)
5..7	reserved

bsFixedGainsDMX Defines the gains used for the downmix according to:

Table 45 — bsFixedGainsDMX

bsFixedGainDMX	Downmix gain
0	1
1	2^(1/4)
2	2^(2/4)
3	2^(3/4)
4	2^(4/4)
5	2^(5/4)
6	2^(6/4)
7	2^(8/4)

bsMatrixMode Indicates if a matrix compatible stereo downmix has been generated in the encoder according to:

Table 46 — bsMatrixMode

bsMatrixMode	Meaning
0	normal downmix
1	matrix compatible stereo downmix

bsTempShapeConfig Indicates operation mode of temporal shaping (STP or GES) in the decoder according to:

Table 47 — bsTempShapeConfig

bsTempShapeConfig	Meaning
0	do not apply temporal shaping
1	apply STP
2	apply GES
3	reserved

bsDecorrConfig Indicates operation mode of the decorrelator in the decoder according to:

Table 48 — bsDecorrConfig

bsDecorrConfig	Meaning
0	QMF split frequencies: 3, 15, 24, 65
1	QMF split frequencies: 3, 50, 65, 65
2	QMF split frequencies: 0, 15, 65, 65
3	Reserved

bs3DaudioMode Indicates that the stereo downmix was 3D audio encoded and that inverse HRTF processing is to be applied.

bsEnvQuantMode Defines the quantization mode of the envelope shaping data according to:

Table 49 — bsEnvQuantMode

bsEnvQuantMode	Meaning
0	5-step quantizer
1	reserved

bs3DaudioHRTFset Indicates the set of HRTF parameters according to the following table:

Table 50 — bs3DaudioHRTFset

bs3DaudioHRTFset	Meaning
0	set of HRTF parameters defined by ParamHRTFset()
1	reserved
2	reserved
3	reserved

OttConfig() Syntactic element that contains the configuration of an OTT box.

bsOttBands Defines the number of parameter bands 0 <= pb < **bsOttBands** for which OTT information is present.

TttConfig() Syntactic element that contains the configuration of a TTT box.

bsTttDualMode Indicates if the TTT box operates in different modes for a low and high band range according to:

Table 51 — bsTttDualMode

bsTttDualMode	Meaning
0	same TTT mode for full band range (i.e. no separate high band range)
1	different TTT modes for low and high band ranges

bsTttModeLow Indicates the mode of operation of the TTT box according to:

Table 52 — bsTttModeLow

bsTttModeLow	Meaning
0	Prediction mode (2 CPC, ICC) with decorrelation
1	Prediction mode (2 CPC, ICC) without decorrelation
2	energy-based mode (2 CLD) with subtraction, matrix compatibility enabled
3	energy-based mode (2 CLD) with subtraction, matrix compatibility disabled
4	energy-based mode (2 CLD) without subtraction, matrix compatibility enabled
5	energy-based mode (2 CLD) without subtraction, matrix compatibility disabled
6	Reserved
7	Reserved

bsTttBandsLow Defines the number of parameter bands for which TTT information is present. The full band range or low band range is 0 <= pb < **bsTttBandsLow**.

bsTttModeHigh Same as bsTttModeLow but for high band range.

bsTttBandsHigh Same as bsTttBandsLow but for high band range. The high band range is bsTttBandsLow <= pb < **bsTttBandsHigh**.

ParamHRTFset() Syntactic element that contains a set of HRTF parameters.

bsHRTFfreqRes Indicates number parameter bands for HRTF according to:

Table 53 — bsHRTFfreqRes

bsHRTFfreqRes	HRTFnumBands	HRTFnumPhase
0	Reserved	Reserved
1	28	13
2	20	13
3	14	8
4	10	7
5	7	4
6	5	3
7	4	3

bsHRTFasymmetric Indicates if HRTF set is asymmetric.

bsHRTFlevelLeft Level of left ear (quantization step size 1 dB).

bsHRTFlevelRight Level of right ear (quantization step size 1 dB).

bsHRTFphase Indicates if phase difference data is conveyed.

bsHRTFphaseLR Phase difference between left and right ear (uniform quantization).

bsHRTFicc Indicates if HRTF ICC data is conveyed.

bsHRTFiccLR HRTF ICC data between the left and right ear according to MPEG surround fine ICC quantization.

SpatialExtensionConfig() Syntactic element that acts as container to carry extensions to the spatial audio configuration. The presence of a spatial extension of type bsSacExtType (see Table 54) is signaled by the presence of a SpatialExtensionConfigData(bsSacExtType) element in SpatialExtensionConfig().

sacExtNum Helper variable storing the number spatial extension containers present.

bsSacExtType Indicates type of spatial extension data according to:

Table 54 — bsSacExtType

bsSacExtTyp	Meaning
0	Residual coding data
1	Arbitrary downmix residual coding data
2	Arbitrary tree extension data
3	User data (data delivered to applications outside the scope of this specification)
4..11	Reserved, SpatialExtensionFrameData() present
12..15	Reserved, SpatialExtensionFrameData() not present

sacExtType[ec] Helper variable storing the type of spatial extension data carried in the extension container ec.

bsSacExtLen Number of bytes in SpatialExtensionConfigData() or SpatialExtensionFrameData().

bsSacExtLenAdd Additional number of bytes in SpatialExtensionConfigData() or SpatialExtensionFrameData().

bsSacExtLenAddAdd Further additional number of bytes in SpatialExtensionConfigData().

bsFillBits Fill bits, to be ignored.

SpatialExtensionConfigData(bsSacExtType)
Instance of the SpatialExtensionConfigData that carries configuration data for spatial extension of type bsSacExtType (see Table 54).

SpatialExtensionConfigData(0)
Syntactic element that, if present, indicates that residual coding information is available. Residual coding is only supported for the spatial frame lengths according to Table 55. numSlots is defined by numSlots = **bsFrameLength** + 1.

Table 55 — Values of numSlots for which residual coding is supported

numSlots
15
16
18
24
30
32
36
48
60
64
72

bsResidualSamplingFrequencyIndex
Determines the sampling frequency assumed when decoding the AAC individual_channel_stream(), according to ISO/IEC 14496-4.

bsResidualFramesPerSpatialFrame
Indicates the number of residual frames per spatial frame, ranging from one to four according to

Table 56 — bsResidualFramesPerSpatialFrame

bsResidualFramesPerSpatialFrame	Meaning
0	1 frame
1	2 frames
2	3 frames
3	4 frames

ResidualConfig() Syntactic element that contains the configuration used to encode a residual signal.

bsResidualPresent Indicates if residual signal information is present.

bsResidualBands Defines the number of parameter bands 0 <= pb < **bsResidualBands** for which residual signal information is present.

SpatialExtensionConfigData(1)
Syntactic element that, if present, indicates that arbitrary downmix residual coding information is available. Arbitrary downmix residual coding is only supported for the spatial frame lengths according to Table 55. numSlots is defined by numSlots = **bsFrameLength** + 1.

bsArbitraryDownmixResidualSamplingFrequencyIndex
Determines the sampling frequency assumed when decoding the AAC individual channel streams or channel pair elements, according to ISO/IEC 14496-4.

bsArbitraryDownmixResidualFramesPerSpatialFrame
Indicates the number of arbitrary down-mix residual frames per spatial frame, ranging from one to four according to Table 57

Table 57 — bsArbitraryDownmixResidualFramesPerSpatialFrame

bsArbitraryDownmixResidualFramesPerSpatialFrame	number of arbitrary down-mix residual frames per spatial frame
0	1 frame
1	2 frames
2	3 frames
3	4 frames

bsArbitraryDownmixResidualBands
Defines the number of parameter bands 0 <= bsArbitraryDownmixResidualBands < numBands for which arbitrary down-mix residual signal information is present.

SpatialExtensionConfigData(2)
Syntactic element that, if present, indicates that arbitrary tree data is available.

TreeConfig() Syntactic element describing tree

bsOttBoxPresent Determines whether an intermediate signal in the tree is processed by an OTT box according to:

Table 58 — bsOttBoxPresent

bsOttBoxPresent	Meaning
0	signal connected to output channel
1	signal connected to an OTT box

bsOttDefaultCld Defines the default value for idxCLD[][][] according to:

Table 59 — ottDefaultCld

ottDefaultCld	Meaning
0	default idxCLD[][][] = 0
1	default idxCLD[][][] = 15

bsOttModeLfe Indicates if OTT box operates in normal or LFE mode according to:

Table 60 — ottModeLfe

ottModeLfe	Meaning
0	OTT box in normal mode
1	OTT box in LFE mode

bsOutputChannelPos Specifies the loudspeaker positions associated with each output channel of the tree as follows:

Table 61 — bsOutputChannelPos

bsOutputChannelPos	Channel abbreviation	Loudspeaker position
0	L	Left Front
1	R	Right Front
2	C	Center Front
3	LFE	Low Frequency Enhancement
4	Ls	Left Surround
5	Rs	Right Surround
6	Lc	Left Front Center
7	Rc	Right Front Center
8	Lsr	Rear Surround Left
9	Rsr	Rear Surround Right
10	Cs	Rear Center
11	Lsd	Left Surround Direct
12	Rsd	Right Surround Direct
13	Lss	Left Side Surround
14	Rss	Right Side Surround
15	Lw	Left Wide Front
16	Rw	Right Wide front
17	Lv	Left Front Vertical Height
18	Rv	Right Front Vertical Height
19	Cv	Center Front Vertical Height
20	Lvr	Left Surround Vertical Height Rear
21	Rvr	Right Surround Vertical Height Rear
22	Cvr	Center Vertical Height Rear
23	Lvss	Left Vertical Height Side Surround
24	Rvss	Right Vertical Height Side Surround
25	Ts	Top Center Surround
26	LFE2	Low Frequency Enhancement 2
27-31	reserved	

SpatialFrame() Syntactic element that contains the data of an SAC frame (payload).

bsIndependencyFlag Indicates if lossless coding of current SAC frame is done independently of previous SAC frame, i.e. whether the current SAC frame can be decoded without knowledge about the previous SAC frame.

FramingInfo() Syntactic element that contains information about the number of parameter sets and their associated time slots.

bsFramingType Indicates if parameter time slots information is available according to:

Table 62 — bsFramingType

bsFramingType	Meaning
0	fixed framing (equidistant parameter time slots)
1	variable framing

bsNumParamSets Defines the number of parameter sets in a frame according to: numParamSets = **bsNumParamSets** + 1;

bsParamSlot[ps] Defines the time slot to which each parameter set (0 <= ps < numParamSets) applies, where ps = 0 for the first time slot in a frame. If nBitsParamSlot(0) > 0, **bsParamSlot**[0] is directly read from the bitstream. If nBitsParamSlot(0) **==** 0, then **bsParamSlot**[0] = 0. **bsParamSlot**[ps] for 0 < ps < numParamSets is further determined by **bsDiffParamSlot**[ps].

bsDiffParamSlot[ps] Defines the difference between consecutive **bsParamSlot**[ps]'s according to **bsParamSlot**[ps] = **bsParamSlot**[ps-1] + **bsDiffParamSlot**[ps] + 1 for 0 < ps < numParamSets, where **bsDiffParamSlot**[ps] = 0 if nBitsParamSlot(ps) **==** 0.

OttData() Syntactic element that contains all parameters for all OTT boxes.

TttData() Syntactic element that contains all parameters for all TTT boxes.

SmgData() Syntactic element that contains the information about the temporal smoothing to be applied to the dequantized spatial parameters.

bsSmoothMode Defines the smoothing mode for a parameter set according to:

Table 63 — bsSmoothMode

bsSmoothMode	Meaning
0	set all smoothing flags to 0 (off)
1	keep previous smoothing parameters unchanged
2	set all smoothing flags to 1 (on)
3	read coded smoothing flags

bsSmoothTime Defines the time constant of the parameter smoothing filter by means of time slots for 1st order IIR smoothing. The number or time slots depends on **bsSmoothTime**.

Table 64 — number of time slots depending on bsSmoothTime

bsSmoothTime	smoothTime
0	64
1	128
2	256
3	512

bsFreqResStrideSmg Indicates whether and how parameter bands are grouped for transmission of **bsSmgData**. Grouping is performed according to Table 70.

bsSmgData Smoothing flag for a certain individual parameter set and parameter band.

smgData[ps][pb] Smoothing flag for the i:th parameter set (0 <= ps < numParamSets) and the pb:th parameter band (0 <= pb < numBands) according to:

Table 65 — smgData

smgData	Meaning
0	no smoothing
1	smoothing according to time constant

TempShapeData() Syntactic element that contains the information about the temporal envelope shaping to be applied to the upmixed signals.

bsTempShapeEnable Indicates if temporal envelope shaping data is present.

bsTempShapeEnableChannel[ch]
Indicates if a temporal envelope shaping tool is applied to an upmix channel ch according to:

Table 66 — bsTempShapeEnable

bsTempShapeEnableChannel[ch]	Meaning
0	temporal processing inactive
1	temporal processing active

The channel ordering is according to Table 67:

Table 67 — Channel ordering for TempShapeEnableChannel

bsTempShape Config	bsTreeConfig	numTemp ShapeChan	Channel ordering
1	0	5	L,R,C,Ls,Rs
1	1	5	L,R,C,Ls,Rs
1	2	4	L,R,Ls,Rs
1	3	6	L,R,Ls,Rs,Lc,Rc
1	4	6	L,R,Ls,Rs,Lsr,Rsr
1	5	4	L,R,Lc,Rc
1	6	4	Ls,Rs,Lsr,Rsr
2	0	5	L,R,C,Ls,Rs
2	1	5	L,R,C,Ls,Rs
2	2	5	L,R,C,Ls,Rs
2	3	7	L,R,C,Ls,Rs,Lc,Rc
2	4	7	L,R,C,Ls,Rs,Lsr,Rsr
2	5	4	L,R,Lc,Rc
2	6	4	Ls,Rs,Lsr,Rsr

EnvelopeReshapeHuff() Syntactic element that contains additional envelope information in case Guided Envelope Shaping (GES) is applied.

bsCodeW Huffman code word or escape code.

hcod2D_EnvRes Two-dimensional Huffman code (Table A.11) used for coding envelope reshaping information.

ArbitraryDownmixData() Syntactic element that contains the arbitrary downmix gain parameters.

EcData() Syntactic element that contains all parameter subsets of a given parameter in the SAC frame.

bsXXXdataMode Indicates whether and how new information for a parameter subset is encoded according to:

Table 68 — bsXXXdataMode

bsXXXdataMode	Meaning
0	set to default parameter values
1	keep previous parameter values unchanged
2	interpolate parameter values
3	read losslessly coded parameter values

bsDataPairXXX Indicates if two subsequent parameter subsets are coded jointly as a pair.

bsQuantCoarseXXX Indicates if coarse quantization is employed according to:

Table 69 — bsQuantCoarseXXX

bsQuantCoarseXXX	Meaning
0	parameter values coded with full quantizer resolution (fine quantization)
1	parameter values coded with half quantizer resolution (coarse quantization)

bsFreqResStrideXXX Indicates whether and how parameter bands are grouped for entropy coding of XXX data. Grouping is performed according to Table 70.

Table 70 — bsFreqResStrideXXX

bsFreqResStrideXXX	pbStride
0	1 (i.e., no grouping)
1	2
2	5
3	28

EcDataPair() Syntactic element that contains one or two temporally subsequent parameter subsets of a given parameter in the SAC frame.

bsPcmCodingXXX Indicates whether PCM coding is applied.

bsPilotCodingXXX Indicates whether pilot-based coding is applied in a single or in two temporally successive data sets.

GroupedPcmData() Syntactic element that contains one or two temporally subsequent parameter subsets of a given parameter in the SAC frame, where groups of quantized values are represented by a single PCM code. numQuantSteps depends on the data type XXX and whether coarse or fine quantization is used and is defined in Table 71. maxGrpLen depends on numQuantSteps and is defined in Table 72.

Table 71 — numQuantSteps

XXX (dataType)	numQuantStepsXXXCoarse	numQuantStepsXXXFine
CLD	16	31
ICC	4	8
CPC	26	51

Table 72 — maxGrpLen

numQuantSteps	maxGrpLen
3	5
6	5
7	6
11	2
13	4
19	4
25	3
51	4
any other value	1

bsPcmWord Single PCM code word representing a group of jointly coded quantized values.

DiffHuffData() Syntactic element that contains one or two temporally subsequent parameter subsets of a given parameter in the SAC frame, where the quantized values are coded using a combination of differential coding and Huffman coding.

hcodPilot_XXX One-dimensional Huffman code (Table A.2, Table A.3, and Table A.4) used for coding of pilot the data type of which is determined by the value of XXX. It is applied for coding of the pilot signal whenever pilot-based coding is applied.

bsDiffType One or two bits determining whether differential coding in frequency or in time direction is applied in a single or in two temporally successive data sets.

bsCodingScheme One bit determining whether 1D or 2D Huffman coding is applied:

Table 73 — bsCodingScheme

Mnemonic	Value	Meaning
HUFF_1D	0	1D Huffman coding
HUFF_2D	1	2D Huffman coding

bsPairing One bit determining the pairing direction of the 2D Huffman code:

Table 74 — bsPairing

Mnemonic	Value	Meaning
FREQ_PAIR	0	pairing in frequency direction
TIME_PAIR	1	pairing in time direction

bsDiffTimeDirection One bit determining whether differential coding in time direction is calculated relative to predecessor or successor frame.

Table 75 — bsDiffTimeDirection

Mnemonic	Value	Meaning
BACKWARDS	0	difference relative to previous parameter time slot data
FORWARDS	1	difference relative to following parameter time slot data

HuffData1D() Syntactic element that contains Huffman data making use of one-dimensional Huffman codes.

hcodFirstBand_XXX One-dimensional Huffman code (Table A.2, Table A.3, and Table A.4)used for coding of data the data type of which is determined by the value of XXX. It is applied for coding of the lowest frequency band whenever differential coding in frequency direction is applied.

hcod1D_XXX_YY One-dimensional Huffman code (Table A.5, Table A.6, and Table A.7) used for coding of data the data type of which is determined by the value of XXX. It is applied for pilot-based or differentially coded data with YY determining the usage of pilot-based coding or the direction of the difference calculation (PC: pilot-based coding;DF: differences in frequency direction; DT: differences in time direction). The Huffman table used for hcod1D_XXX_PC shall be the same as that used for hcod1D_XXX_DT.

bsSign One bit determining the sign of a 1D Huffman coded value (0: positive; 1: negative).

HuffData2DFreqPair() Syntactic element that contains Huffman data making use of two-dimensional Huffman codes representing pairs of values neighboring in frequency direction.

hcodLavIdx — One-dimensional Huffman code (Table A.24) used for coding of the LavIdx data. This determines the largest absolute value in one or two data sets coded with two-dimensional Huffman codes according to Table 76.

Table 76 — lavTabXXX

LavIdx	lavTabCLD [LavIdx]	lavTabICC [LavIdx]	lavTabCPC [LavIdx]
0	3	1	3
1	5	3	6
2	7	5	9
3	9	7	12

hcod2D_XXX_YY_ZZ_LL_escape — Single Huffman code (Table A.8, Table A.9, and Table A.10) out of the two-dimensional Huffman code table hcod2D_XXX_YY_ZZ_LL inducing an escape mechanism within which one or more pairs of values are transmitted by means of grouped PCM coding.

hcod2D_XXX_YY_ZZ_LL — Two-dimensional Huffman code (Table A.11 to Table A.22) used for coding of data the data type of which is determined by the value of XXX. It is applied for pilot-based or differentially coded data with YY determining the usage of pilot-based coding or the direction of the difference calculation (PC: pilot-based coding; DF: differences in frequency direction; DT: differences in time direction) and ZZ determining the direction of the 2D pairing (FP: pairing of values neighboring in frequency direction; TP: pairing of values neighboring in time direction). The value of LL determines one out of four possible LAV steps according to Table 76. The Huffman table used for hcod2D_XXX_PC_ZZ_LL shall be the same as that used for hcod2D_XXX_DF_TP_LL.

HuffData2DTimePair() — Syntactic element that contains Huffman data making use of two-dimensional Huffman codes representing pairs of values neighboring in time direction.

SymmetryData() — Syntactic element determining to which one out of four possible symmetry regions a pair of two jointly coded values belongs to.

bsSymBit[i] — Two bits determining one out of four symmetry regions.

LsbData() — Syntactic element that contains the least significant bits of each value in one or two temporally subsequent parameter subsets of a given parameter in the SAC frame.

bsLsb — One bit determining the mapping between quantization indices resulting from applying coarse or fine quantization scales, respectively.

bsXXXpcm[pi][ps][pb] — PCM coded indices where XXX is to be replaced by CLD, ICC, or CPC. The indices are offset so that they cannot be negative.

bsXXXmsbDiff[pi][ps][pb] — Differentially coded most significant bits of a quantization index of data type XXX (where XXX can be either CLD or ICC or CPC) belonging to parameter index pi, parameter set ps and parameter band pb.

bsXXXlsb[pi][ps][pb] — Least significant bit of a quantization index of data type XXX [...]. May only be 1 in case of data type CLD and fine quantization; otherwise always 0.

idxXXX[pi][ps][pb] — Quantized spatial parameter (as index, can be negative) for the pi:th XXX parameter for the ps:th parameter set (0 <= ps < numParamSets) and the pb:th parameter band (0 <= pb < numBands). XXX is to be replaced by CLD, ICC, or CPC.

SpatialExtensionFrame() — Syntactic element that acts as container to carry extensions to the spatial audio frame.

SpatialExtensionFrameData(bsSacExtType)
: Instance of the SpatialExtensionFrameData that carries frame data for spatial extension of type bsSacExtType (see Table 54).

SpatialExtensionFrameData(0)
: Syntactic element carrying residual coding data.

ResidualData()
: Syntactic element that contains the residual signal information.

bsIccDiffPresent[pi][ps]
: Signals the presence of differential ICC parameter indices for parameter instance pi and parameter set ps, which will be used to update **idxICC**[pi][ps][pb] for parameter bands 0 <= pb < **bsResidualBands**[pi]. A value of 0 indicates that no ICC difference is present for parameter instance pi and parameter set ps. A value of 1 indicates the presence of an ICC difference for parameter instance pi and parameter set ps.

hcod1D_ICC_Diff
: One-dimensional Huffman code (Table A.23) used for coding of IccDiff data.

IccDiff[pi][ps][pb]
: Differential ICC parameters to update **idxICC**[pi][ps][pb] for parameter instance pi, parameter set ps and parameter band pb. Values are encoded according to Table A.23.

individual_channel_stream()
: MPEG-2 AAC Low Complexity profile individual_channel_stream() elements according to the syntax defined in Table 16 (and related tables) of subclause 6.3 of ISO/IEC 13818-7. Decoding of an individual_channel_stream() element also determines the value of **window_sequence**, according to ISO/IEC 13818-7.

  A second individual_channel_stream() element is present when **window_sequence** (determined by the first individual_channel_stream() element) equals EIGHT_SHORT_SEQUENCE and **tempExtraFrame** equals 18, 24 or 30. In this case, the value of **window_sequence** determined by the second individual_channel_stream() element shall equal EIGHT_SHORT_SEQUENCE.

  Restrictions apply to the elements of the individual_channel_stream() syntax and the ics_info() syntax. The restrictions applied to the elements of the individual_channel_stream() syntax are given in Table 77.

Table 77 — Restrictions in syntax of individual_channel_stream()

Tool	Allowed value
pulse_data_present	0
gain_control_data_present	0

The restrictions applied to the elements of ics_info(), defined in Table 15 of subclause 6.3 of ISO/IEC 13818-7, are given in Table 78.

Table 78 — Restrictions in syntax of ics_info()

Tool	Allowed value
window_shape	0
predictor_data_present	0

In addition to the existing restriction on the maximum number of scalefactor bands to which Temporal Noise Shaping is applied, (the constant TNS_MAX_BANDS defined in Table 33 of subclause 7.1.6 of ISO/IEC 13818-7), the lowest scalefactor band where Temporal Noise Shaping is applied is restricted and depends on the sampling rate (derived from **bsResidualSamplingFrequencyIndex**) and whether **window_sequence** indicates a long or short window, as specified in Table 79.

Table 79 — Lowest scalefactor band where Temporal Noise Shaping is applied

Sampling Rate [Hz]	Sfb (long windows)	Sfb (short windows)
96000	10	2
88200	10	2
64000	14	2
48000	15	3
44100	15	3
32000	19	4
24000	23	5
22050	24	5
16000	22	7
12000	26	9
11025	27	9
8000	28	11

SpatialExtensionFrameData(1)
Syntactic element carrying arbitrary downmix residual coding data.

ArbitraryDownmixResidualData()
Syntactic element that contains the arbitrary downmix residual signal information.

bsArbitraryDownmixResidualAbs[i]
indicates whether the signal(s) of the arbitrary down-mix residual AAC element i (either an individual_channel_stream or a channel_pair_element) is generated either in the differential (0) or in the absolute (1) mode, disregarding signal modifications to allow for smooth transitions.

bsArbitraryDownmixResidualAlphaUpdateSet[i]
indicates first parameter set to which new value of α_k (see subclause 6.5.2.3) is applied according to Table 80.

Table 80 — bsArbitraryDownmixResidualAlphaUpdateSet

bsArbitraryDownmixResidualAlphaUpdateSet[i]	Meaning
0	The new value of α_k is applied to all parameter sets in the frame.
1	The value of α_k calculated in the previous frame is applied to the first parameter set. The new value is applied starting from the second parameter set.

numAacEl Specifies the number of AAC elements in the current frame according to Table 81.

AacEl Specifies the element type of each AAC element, according to Table 81. AacEl[i] represents the i^{th} entry in the AacEl string. '0' indicates an "individual_channel_stream" (ICS) whereas a value of '1' indicates a "channel_pair_element".

Table 81 — number of AAC elements and the element type depending on the number of down-mix channels

numInChan					
1		2		6	
numAacEl	AacEl	numAacEl	AacEl	numAacEl	AacEl
1	'0'	1	'1'	2	'111'

SpatialExtensionFrameData(2) Syntactic element carrying arbitrary tree data.

ArbitraryTreeData() Syntactic element that contains the ATD data for all OTT boxes in the arbitrary tree.

6 Decoding process

6.1 Compressed data stream decoding and de-quantization

6.1.1 Introduction

This subclause describes the decoding and dequantization of the bitstream payload into variables that are used by the different modules of the spatial audio decoder. The final subclause describes the derivation of the matrices $\mathbf{M}_1^{n,k}$ and $\mathbf{M}_2^{n,k}$, in the case of Enhanced Matrix Mode operation, where the matrix coefficients are directly calculated from the downmix.

6.1.2 Decoding of CLD ATD, ICC, CPC, and arbitrary downmix gain parameters

6.1.2.1 Overview

The decoding of the SpatialFrame() data result in the parameter indices idxXXX[][][] of the quantized CLD, ATD, ICC, and CPC parameters

idxCLD[pi][ps][pb] having values in the range -15 .. 15,

idxATD[pi][ps][pb] having values in the range -15 .. 15,

idxICC[pi][ps][pb] having values in the range 0 .. 7,

idxCPC[pi][ps][pb] having values in the range -20 .. 30,

where optional arbitrary downmix gain parameters are handled as CLD data, and where

pi = parameter instance having values in the range which, for CLD, ICC and CPC have the range 0 .. numOttBoxes+4*numTttBoxes+numInChan-1, and for ATD have the range 0 .. numOttBoxesAT-1

ps = parameter set having values in the range 0 .. numParamSets-1,

pb = parameter band having values in the range 0 .. numBands-1,

pg = parameter group having values in the range 0 .. dataBands-1,

is carried out as described in the following subclauses. The value ps = -1 refers to the last parameter set of the previous frame, which is required when applying time-differential decoding over frame boundaries.

The following process described in the subclauses below is carried out for all instances pi of all parameter types CLD, ATD, ICC, CPC, where XXX denotes the actual parameter type.

6.1.2.2 Loop over parameter sets

Prepare looping over parameter sets.

```
dataSets = 0;
for (ps=0, ps<numParamSets; ps++) {
    if (bsXXXdataMode[pi][ps] == 3) { /* coded */
        dataSetIdx[ps] = dataSets;
        paramSet[dataSets] = ps;
         paramHandled[ps] = 0;
        dataSets++;
    }
}
```

Decode all parameter sets ps according to their bsXXXdataMode[][] according to the subclause below.

```
while (ps=0; ps<numParamSet; ps++) {
    switch (bsXXXdataMode[pi][ps]) {
    case 0: /* default */
        for (pb=0; pb<numBands, pb++) {
            switch (XXX) {
            CLD,ATD:
                if (defaultXXX[pi])
                    idxXXX[pi][ps][pb] = 15;
                else
                    idxXXX[pi][ps][pb] = 0;
                break;
            ICC:
                idxXXX[pi][ps][pb] = 0;
                break;
            CPC:
                idxXXX[pi][ps][pb] = 10;
                break;
            }
        }
        break;
    case 1: /* keep */
    case 2: /* interpolate */
        for (pb=0; pb<numBands, pb++) {
            idxXXX[pi][ps][pb] = idxXXX[pi][ps-1][pb];
        }
        break;
    case 3: /* coded */
        if (!paramHandled[ps]) {
            DecodeDataPair(); /* see subclause 6.1.2.3 */
        }
        break;
    }
}
```

6.1.2.3 Decoding of a data pair element

6.1.2.3.1 DecodeDataPair overview

Dependent on the signaled grouping of data (**bsDataPairXXX**[][]) the delta decoding is either done for a single parameter set or for two parameter sets which is reflected by the variable paramHandled[ps]. An

overview of the data buffers involved in decoding of a data pair is shown in Figure 13. The actual delta decoding is preceded by a pre processing which has three purposes listed below:

- Map the already decoded indices to match the frequency grouping of the current data which is signaled by **bsFreqResStrideXXX**[][].
- Offset the indices to so that they never have negative values.
- If the current data has coarse quantization, then convert the previous data to coarse as well. Since the coarse quantization is a subset of the fine quantization, it is possible to do delta coding in between data sets even though they do not have the same quantization resolution.

The actual delta decoding includes the pilot delta decoding and the data delta decoding. The pilot delta decoding is done compared to a single value **bsXXXpilot[][]**. The data delta decoding is either done in the frequency direction from low frequencies to high frequencies or in the time direction generally compared to the previous parameter set. In the case that two parameter sets are decoded (**bsDataPairXXX**[][] == 1) it is possible to delta code compared to the next parameter set and hence that parameter sets needs to be decoded before.

Similarly to the pre processing the post processing has three purposes which are listed below:

- Map the decoded indices to the highest frequency resolution which is signaled by *bsFreqRes*.
- Remove the offset.
- Convert indices into fine quantization index range.

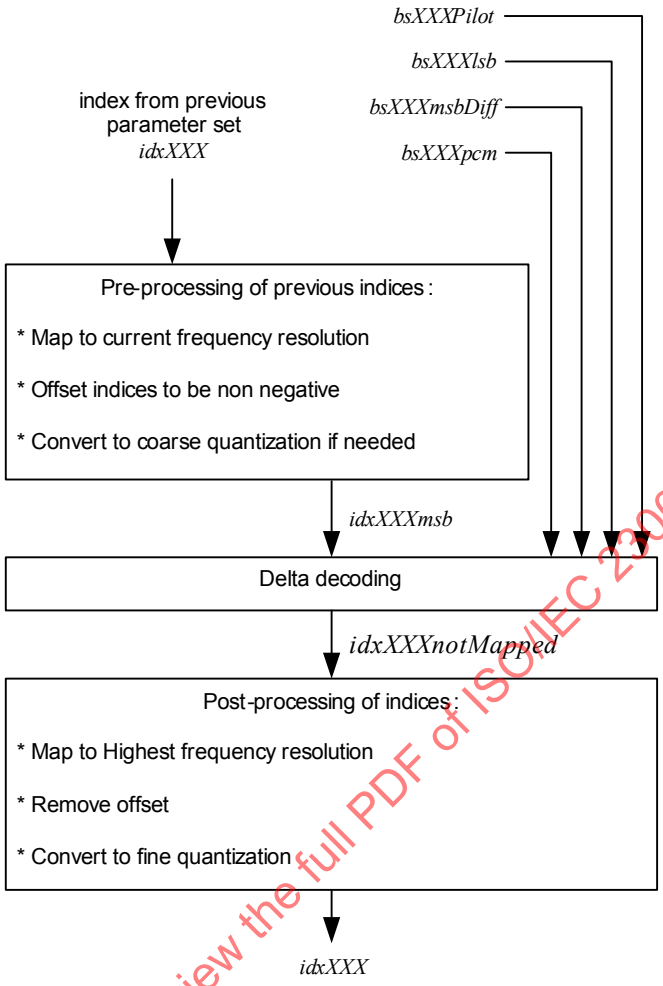

Figure 13 — DecodeDataPair data overview

6.1.2.3.2 DecodeDataPair syntax

The DecodeDataPair() process for a single parameter or a parameter pair of type XXX, instance pi, starting at parameter set ps is carried out as follows.

Firstly, the previous data is pre-processed for time-differential decoding.

```
setIdxStart = dataSetIdx[ps];
startBand = startBandXXX[pi];
stopBand = stopBandXXX[pi];
pbStride = pbStrideTable[bsFreqResStrideXXX[pi][setIdx]];  /* see Table 70 */
dataBands = (stopBand - startBand - 1)/pbStride + 1; /* ANSI C integer math
*/
aGroupToBand = createMapping(startBand, stopBand, pbStride);     /* see
subclause 6.1.2.4 */
for (pg=0; pg<dataBands; pg++) {
    pb = aGroupToBand[pg];
    tmp = idxXXX[pi][ps-1][pb];
    switch (XXX) {
    case CLD,ATD:
```

```
        if (bsQuantCoarseXXX[pi][setIdx]) {
            tmp = (tmp/2)+7;    /* ANSI C integer math */
        }
        else {
            tmp = tmp+15;
        }
        break;
    case ICC:
        if (bsQuantCoarseXXX[pi][setIdx]) {
            tmp = tmp/2;    /* ANSI C integer math */
        }
        break;
    case CPC:
        tmp = (tmp+20)/2;    /* ANSI C integer math */
        break;
    }
    idxXXXmsb[pi][setIdxStart-1][pg] = tmp;
}
```

Then, delta decoding is done in the following order

```
if (bsPilotCoding[pi][setIdx]) {
    decodePilotDeltaData(setIdxStart);
    if (bsDataPairXXX[pi][setIdxStart]) {
        decodePilotDeltaData(setIdxStart+1);
    }
}
else if (!bsPcmCodingXXX[pi][setIdx]) {
    if (bsDataPairXXX[pi][setIdxStart]) {
        if ((bsDiffTypeXXX[pi][setIdx]==DIFF_TIME) &&
            (bsDiffTimeDirectionXXX[pi][setIdx]==FORWARDS)) {
            decodeDeltaData(setIdxStart+1);
            decodeDeltaData(setIdxStart);
        }
        else {
            decodeDeltaData(setIdxStart);
            decodeDeltaData(setIdxStart+1);
        }
    }
    else {
        decodeDeltaData(setIdxStart);
    }
} else {
    idxXXXnotMapped[pi][setIdx][pg] = bsXXXpcm[pi][setIdx][pg];
}
```

where the decodePilotDeltaData(setIdx) process is carried out as follows.

```
for (pg= 0; pg< dataBands; pg++) {
    idxXXXmsb[pi][setIdx][pg] =
        bsXXXmsbDiff[pi][setIdx][pg] + bsXXXpilot[pi][setIdx];
    if (bsQuantCoarseXXX[pi][setIdx] || (XXX != CPC)) {
        idxXXXnotMapped[pi][setIdx][pg] = idxXXXmsb[pi][setIdx][pg];
    }
    else {
        idxXXXnotMapped[pi][setIdx][pg] =
            2*idxXXXmsb[pi][setIdx][pg] + bsXXXlsb[pi][setIdx][pg];
    }
}
```

and the decodeDeltaData(setIdx) process is carried out as follows.

```
for (pg= 0; pg< dataBands; pg++) {
    switch (bsDiffTypeXXX[pi][setIdx]) {
    case DIFF_FREQ:
        if( pg > 0 ) {
            idxXXXmsb[pi][setIdx][pg] =
                idxXXXmsb[pi][setIdx][pg-1] + bsXXXmsbDiff[pi][setIdx][pg];
        } else {
            idxXXXmsb[pi][setIdx][pg] = bsXXXmsbDiff[pi][setIdx][pg];
        }
        break;
    case DIFF_TIME:
        if ( (pg > 0) || (mixedTimePairXXX[pi][setIdx]) ) {
            switch (bsDiffTimeDirectionXXX[pi][setIdx]) {
            case BACKWARDS:
                idxXXXmsb[pi][setIdx][pg] =
                    idxXXXmsb[pi][setIdx-1][pg] + bsXXXmsbDiff[pi][setIdx][pg] ;
                break;
            case FORWARDS:
                /* assert that idxXXXmsb[pi][setIdx+1] is already available */
                idxXXXmsb[pi][setIdx][pg] =
                    idxXXXmsb[pi][setIdx+1][pg] - bsXXXmsbDiff[pi][setIdx][pg];
                break;
            }
        } else {
            idxXXXmsb[pi][setIdx][pg] = bsXXXmsbDiff[pi][setIdx][pb];
        }
    }
    if (bsQuantCoarseXXX[pi][setIdx] || (XXX != CPC)) {
        idxXXXnotMapped[pi][setIdx][pg] = idxXXXmsb[pi][setIdx][pg];
    }
    else {
        idxXXXnotMapped[pi][setIdx][pg] =
            2*idxXXXmsb[pi][setIdx][pg] + bsXXXlsb[pi][setIdx][pg];
    }
}
```

Finally, the following post-process is applied to the decoded data.

```
for (i=0; i<=bsDataPairXXX[pi][setIdxStart]; i++) {
    setIdx = setIdxStart+i;
    ps = paramSet[setIdx];
    paramHandled[ps] = 1;
    for (pg=0; pg<dataBands; pg++) {
        tmp = idxXXXnotMapped[pi][setIdx][pg];
        switch (XXX) {
        case CLD,ATD:
            if (bsQuantCoarseXXX[pi][setIdx]) {
                tmp = (tmp-7)*2;
                if (tmp==-14) tmp=-15;
                if (tmp==14) tmp=15;
            }
            else {
                tmp = tmp-15;
            }
            break;
        case ICC:
            if (bsQuantCoarseXXX[pi][setIdx]) {
                tmp = tmp*2;
```

```
            }
            break;
        case CPC:
            if (bsQuantCoarseXXX[pi][setIdx]) {
                tmp = tmp*2;
            }
            tmp = tmp-20;
            break;
        }
        pbStart = aGroupToBand[pg];
        pbStop = aGroupToBand[pg+1];
        for (pb=pbStart; pb<pbStop; pb++) {
            idxXXX[pi][ps][pb] = tmp;
        }
    }
}
```

6.1.2.4 Handling of parameter band stride

Handling of the stride for parameter band grouping is carried out by the following function which creates a vector with start and stop borders for the frequency grouping.

```
createMapping(startBand, stopBand, pbStride)
{
    inBands = stopBand-startBand;
    outBands = (inBands - 1) /pbStride + 1; /* ANSI C integer math */

    bandsAchieved = outBands*pbStride;
    bandsDiff = inBands - bandsAchieved;
    for( i = 0; i < outBands; i++ ) {
        vDk[i] = pbStride;
    }

    incr =1;
    k=0;

    while(bandsDiff < 0 ) {
        vDk[k] = vDk[k] - incr;
        k = k + incr;
        bandsDiff = bandsDiff + incr;
        if(k >= outBands) {
            k=0;
        }
    }
    aMap[0] = startBand;
    for( i = 0; i < outBands; i++ ) {
        aMap[i+1] = aMap[i] + vDk[i];
    }
    return( aMap )
}
```

6.1.3 Decoding of smoothing control parameters

The decoding of the smoothing control parameters to obtain **smgData**[*ps*][*pb*] is carried out as follows. The value ps = -1 refers to the last parameter set of the previous frame, which is required for time-differential decoding.

```
for (ps=0; ps<numParamSets; ps++) {
    switch (bsSmoothMode[ps]) {
    case 0:
        for (pb=0; pb<numBands; pb++) {
            smgData[ps][pb] = 0;
        }
        break;
    }
    case 1:
        for (pb=0; pb<numBands; pb++) {
            smgData[ps][pb] = smgData[ps-1][pb];
        }
        break;
    }
    case 2:
        for (pb=0; pb<numBands; pb++) {
            smgData[ps][pb] = 1;
        }
        break;
    }
    case 3:
        pbStride = pbStrideTable[bsFreqResStrideSmg[ps]]; /* see Table 70 */
        dataBands = (numBands - 1)/pbStride + 1;
        aGroupToBand=createMapping(0,numBands,pbStride););/*see subclause
                                                            6.1.2.4*/
        for (pg= 0; pg< dataBands; pg++) {
            pbStart = aGroupToBand[pg];
            pbStop = aGroupToBand[pg+1];
            for (pb=pbStart; pb<pbStop; pb++) {
                smgData[ps][pb] = bsSmgData[ps][pg];
            }
        }
        break;
    }
}
```

6.1.4 Decoding of number of parameter bands

The number of available parameter bands is defined according to:

$$M_{\text{par}} = numBands$$

where $numBands$ is defined in Table 39.

6.1.5 Decoding of residual coding elements

A residual coding element consists of one or two individual_channel_stream() elements (for OTT and TTT boxes) and differential ICC elements (for OTT boxes). The presence of a residual coding element for parameter instance *pi* is signaled through **bsResidualPresent**(*pi*) and the presence of differential ICC elements is signaled by **bsIccDiffPresent**(*pi,ps*).

The individual_channel_stream() elements are decoded according to the description in subclause 8.3 ("Decoding of a single_channel_element (SCE), channel_pair_element (CPE) and individual_channel_stream (ICS)") of ISO/IEC 13818-7 up to the inverse IMDCT, resulting in 1024 MDCT coefficients and the sequence of windows *window_sequence*.

In the case *window_sequence* equals EIGHT_SHORT_SEQUENCE and tempExtraFrame equals 18, 24 or 30, (see Table 34), the decoded MDCT coefficients of the second individual_channel_stream() element are appended to those of the first element, resulting in 2048 MDCT coefficients.

The number of residual bands $\mathbf{m}_{\text{resPar}}(pi)$ is set equal to **bsResidualBands**(*pi*) according to:

$$\mathbf{m}_{\text{resPar}}(pi) = \mathbf{bsResidualBands}(pi)$$

for $0 \le pi < numOttBoxes + numTttBoxes$.

The decoded MDCT coefficients are transformed to the QMF domain by the transformation described in subclause 6.9.2.

In the case **bsIccDiffPresent**(*pi,ps*) equals 1, differential ICC elements are decoded according to the Huffman code given in Table A.23, yielding **IccDiff**(*pi,ps,pb*). The ICC parameter indices **idxICC**(*pi,ps,pb*) are updated by adding **IccDiff**(*pi,ps,pb*) when the parameter band *pb* corresponds to a residual band, yielding **idxICCUpd**(*pi,ps,pb*) according to:

$$\mathbf{idxICCUpd}(pi,ps,pb) = \begin{cases} \mathbf{idxICC}(pi,ps,pb) + \mathbf{iccDiff}(pi,ps,pb) & ,if \begin{cases} pb < \mathbf{m}_{resPar}(pi) \\ \mathbf{bsIccDiffPresent}(pi,ps) = 1 \end{cases} \\ \mathbf{idxICC}(pi,ps,pb) & ,otherwise \end{cases}$$

for $0 \le pi < numOttBoxes$, $0 \le ps < numParamSets$ and $0 \le pb < M_{par}$.

6.1.6 Decoding of arbitrary down-mix residual coding elements

A residual coding element consists of a collection of individual_channel_stream() elements, and/or channel_pair_element() elements. The channel_pair_element() and/or individual_channel_stream() elements are decoded according to the description in subclause 8.3 ("Decoding of a single_channel_element (SCE), channel_pair_element (CPE) and individual_channel_stream (ICS)") of ISO/IEC 13818-7 up to the inverse IMDCT, resulting in 1024 MDCT coefficients and the sequence of windows *window_sequence*.

In the case *window_sequence* equals EIGHT_SHORT_SEQUENCE and resFrameLength equals 18, 24 or 30, (see Table 22), the decoded MDCT coefficients of the second channel_pair_element() or individual_channel_stream() element are appended to those of the first element, resulting in 2048 MDCT coefficients. The number of residual bands $\mathrm{m}_{ArtDmxRes}$ is set equal to *bsArbitraryDownmixResidualBands*. The decoded MDCT coefficients are transformed to the QMF domain by the transformation described in subclause 6.9.2.

6.1.7 Decoding of time / frequency grid

The parameter time slots **bsParamSlot**[*ps*] are decoded from the SpatialFrame() data as follows.

If the last signaled parameter time slot is not the last time slot in the frame (*numSlots* - *1*) the last parameter set indices available are copied into a new parameter set and hence the last element in the parameter time slot vector **t** should always be (*numSlots* - *1*). The number of parameter sets is defined according to:

$$L = \begin{cases} numParamSets & ,if\ bsFramingType = 0 \\ numParamSets & ,if\ bsFramingType = 1, \mathbf{bsParamSlot}\left(numParamSets\text{-}1\right) = numSlots - 1 \\ numParamSets + 1 & ,otherwise \end{cases}$$

where $numParamSets = bsNumParamSets + 1$, $numSlots = bsFrameLength + 1$ and furthermore if bsParamSlot[numParamSets-1] != numSlots-1 the last available parameter set is copied according to:

$$\mathbf{idxXXX}\left(pi, numParamSets, pb\right) = \mathbf{idxXXX}\left(pi, numParamSets - 1, pb\right)$$

$$\mathbf{smgData}\left(numParamSets, pb\right) = \mathbf{smgData}\left(numParamSets - 1, pb\right)$$

$$\mathbf{idxICCUpd}\left(pi, numParamSets, pb\right) = \mathbf{idxICCUpd}\left(pi, numParamSets - 1, pb\right)$$ (if applicable)

for all parameter types *XXX*, instances *pi*, and all parameter bands *pb*.

The parameter time slot vector **t** is then finally defined according to:

$$\mathbf{t}\left(l\right) = \begin{cases} ceil\left(\dfrac{numSlots \cdot \left(l+1\right)}{L}\right) - 1 & ,\mathbf{bsFramingType} = 0,\quad l < L - 1 \\ \mathbf{bsParamSlot}\left[l\right] & ,\mathbf{bsFramingType} = 1,\quad l < L - 1 \\ numSlots - 1 & ,l = L - 1 \end{cases},$$

for $0 \le l < L$.

6.1.8 Dequantization of CLD, ATD, ICC or ICCUpd, CPC, and arbitrary downmix gain parameters

All parameter types are dequantized according to the definitions in this subclause for all parameter bands $0 \le m < M_{par}$ and all parameter sets $0 \le l < L$. Throughout this subclause the dequantization function $deq\left(index, parameterType\right)$ is used. It is defined by selecting an appropriate table (Table 82, Table 83 or Table 84) depending on $parameterType$. It returns a dequantized value according to chosen table and $index$.

When parameter interpolation is used as signaled by $\mathbf{bsXXXdataMode}(pi,l,m)=2$ for the corresponding indices $\mathbf{idxXXX}(pi,l,m)$, the dequantization function $deq(index, parameterType)$ will also use the parameter time slot vector $\mathbf{t}$ and the previous and next parameter indices $\mathbf{idxXXX}(pi,l_{bef\S ore},m)$ and $\mathbf{idxXXX}(pi,l_{\text{after}},m)$, respectively, to calculate the interpolated parameter indices according to:

$$\mathbf{idxXXX}(pi,l,m)=\mathbf{idxXXX}(pi,l_{\text{before}},m)+$$
$$INT\left(\frac{\mathbf{idxXXX}(pi,l_{\text{after}},m)-\mathbf{idxXXX}(pi,l_{\text{before}},m)}{\mathbf{t}(l_{\text{after}})-\mathbf{t}(l_{\text{before}})}\left(\mathbf{t}(l)-\mathbf{t}(l_{\text{before}})\right)\right) \quad l_{\text{before}}<l<l_{\text{after}}$$

where

l_{before} is the parameter set with the largest value smaller than l for which $\mathbf{bsXXXdataMode}(pi,l_{\text{before}},m)\neq 2$ and where

l_{after} is the parameter set with the smallest value larger than l for which $\mathbf{bsXXXdataMode}(pi,l_{\text{after}},m)\neq 2$

and where $\mathbf{idxXXX}(pi,-1,m)$ refers to the last parameter set in the previous frame and $\mathbf{t}(-1)$ is set to the first parameter time slot in the current frame, hence equals zero.

In the case $\mathbf{bsIccDiffPresent}(pi,l)$ equals 1, no interpolation is applied to the ICC parameter indices, $\mathbf{idxICC}(pi,l,m)$ in parameter band $0\le m<\mathbf{m}_{\text{resPar}}(pi)$.

Table 82 — CLD and ATD parameter quantization table

Idx	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5
CLD[idx]	-150	-45	-40	-35	-30	-25	-22	-19	-16	-13	-10
Idx	-4	-3	-2	-1	0	1	2	3	4	5	6
CLD[idx]	-8	-6	-4	-2	0	2	4	6	8	10	13
Idx	7	8	9	10	11	12	13	14	15		
CLD[idx]	16	19	22	25	30	35	40	45	150		

Table 83 — ICC parameter quantization table

idx	0	1	2	3	4	5	6	7
ICC[idx]	1	0.937	0.84118	0.60092	0.36764	0	-0.589	-0.99

Table 84 — CPC parameter quantization table

idx	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10
CPC[idx]	-2.0	-1.9	-1.8	-1.7	-1.6	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0
idx	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1
CPC[idx]	-0.9	-0.8	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0.0	0.1
idx	2	3	4	5	6	7	8	9	10	11	12
CPC[idx]	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
idx	13	14	15	16	17	18	19	20	21	22	23
CPC[idx]	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0	2.1	2.2	2.3
idx	24	25	26	27	28	29	30				
CPC[idx]	2.4	2.5	2.6	2.7	2.8	2.9	3.0				

For the CLD parameter type the Energy dependent Quantization (EdQ) tool is available. The use of EdQ is controlled by *bsQuantMode*. In case of a 5-2-5 configuration the EdQ tool is not used. The dequantization of the CLD parameter type is defined as follows.

If *bsQuantMode* = 0 or *bsTreeConfig* > 1 the CLD parameter type is dequantized according to:

$$\mathbf{D}^{Q}_{\mathrm{CLD}}(ott,l,m) = deq\left(\mathbf{idxCLD}(ott,l,m), CLD\right) \quad 0 \le ott < numOttBox$$

and the CLD index shall not be updated by EdQ, hence:

$$\mathbf{idxCLDEdQ}(ott,l,m) = \mathbf{idxCLD}(ott,l,m)$$

otherwise if *bsQuantMode* > 0 the EdQ tool is used and hence the configuration signaled in *bsTreeConfig* is needed for the dequantization of the CLD parameters according to the following, where the function *facFunc*() is defined below.

If *bsTreeConfig* = 0 the CLD dequantization is defined according to:

$$\mathbf{D}^{Q}_{\mathrm{CLD}}(0,l,m) = deq\left(\mathbf{idxCLD}(0,l,m), CLD\right)$$

$$RelativeLocalEnergyFC_{5151}(l,m) = 10\log_{10}\left(\frac{10^{\left(\frac{\mathbf{D}^{Q}_{\mathrm{CLD}}(0,l,m)}{10}\right)}}{1+10^{\left(\frac{\mathbf{D}^{Q}_{\mathrm{CLD}}(0,l,m)}{10}\right)}}\right)$$

$$RelativeLocalEnergyS_{5151}(l,m) = 10\log_{10}\left(\frac{1}{1+10^{\left(\frac{\mathbf{D}^{Q}_{\mathrm{CLD}}(0,l,m)}{10}\right)}}\right)$$

$$\mathbf{idxCLDEdQ}(1,l,m) = \max\left(-15, \min\left(15, round\left(\mathbf{idxCLD}(1,l,m) \cdot facFunc\left(RelativeLocalEnergyFC_{5151}(l,m)\right)\right)\right)\right)$$

$$\mathbf{D}^{Q}_{CLD}(1,l,m) = deq\left(\mathbf{idxCLDEdQ}(1,l,m), CLD\right)$$

$$RelativeLocalEnergyF_{5151}(l,m) = 10\log_{10}\left(\frac{10^{\left(\frac{\mathbf{D}^{Q}_{CLD}(1,l,m)}{10}\right)}}{1+10^{\left(\frac{\mathbf{D}^{Q}_{CLD}(1,l,m)}{10}\right)}} \cdot 10^{\frac{RelativeLocalEnergyFC_{5151}(l,m)}{10}}\right)$$

$$\mathbf{idxCLDEdQ}(2,l,m) = \max\left(-15, \min\left(15, round\left(\mathbf{idxCLD}(2,l,m) \cdot facFunc\left(RelativeLocalEnergyS_{5151}(l,m)\right)\right)\right)\right)$$

$$\mathbf{D}^{Q}_{CLD}(2,l,m) = deq\left(\mathbf{idxCLDEdQ}(2,l,m), CLD\right)$$

$$\mathbf{idxCLDEdQ}(3,l,m) = \max\left(-15, \min\left(15, round\left(\mathbf{idxCLD}(3,l,m) \cdot facFunc\left(RelativeLocalEnergyF_{5151}(l,m)\right)\right)\right)\right)$$

$$\mathbf{D}^{Q}_{CLD}(3,l,m) = deq\left(\mathbf{idxCLDEdQ}(3,l,m), CLD\right)$$

$$\mathbf{D}^{Q}_{CLD}(4,l,m) = deq\left(\mathbf{idxCLD}(4,l,m), CLD\right)$$

If *bsTreeConfig* = 1 the CLD dequantization is defined according to:

$$\mathbf{D}^{Q}_{CLD}(0,l,m) = deq\left(\mathbf{idxCLD}(0,l,m), CLD\right)$$

$$RelativeLocalEnergyLR_{5152}(l,m) = 10\log_{10}\left(\frac{10^{\left(\frac{\mathbf{D}^{Q}_{CLD}(0,l,m)}{10}\right)}}{1+10^{\left(\frac{\mathbf{D}^{Q}_{CLD}(0,l,m)}{10}\right)}}\right)$$

$$\mathbf{idxCLDEdQ}(1,l,m) = \max\left(-15, \min\left(15, round\left(\mathbf{idxCLD}(1,l,m) \cdot facFunc\left(\mathrm{Re}lativeLocalEnergyLR_{5152}(l,m)\right)\right)\right)\right)$$

$$\mathbf{D}^{Q}_{CLD}(1,l,m) = deq\left(\mathbf{idxCLDEdQ}(1,l,m), CLD\right)$$

$$RelativeLocalEnergyL_{5152}(l,m) = 10\log_{10}\left(\frac{10^{\left(\frac{\mathbf{D}^{Q}_{CLD}(1,l,m)}{10}\right)}}{1+10^{\left(\frac{\mathbf{D}^{Q}_{CLD}(1,l,m)}{10}\right)}} \cdot 10^{\frac{RelativeLocalEnergyLR_{5152}(l,m)}{10}}\right)$$

$$RelativeLocalEnergyR_{5152}(l,m) = 10\log_{10}\left(\frac{1}{1+10^{\left(\frac{\mathbf{D}^{Q}_{CLD}(1,l,m)}{10}\right)}} \cdot 10^{\frac{RelativeLocalEnergyLR_{5152}(l,m)}{10}}\right)$$

$$\mathbf{D}^{Q}_{\mathrm{CLD}}(2,l,m) = deq\left(\mathbf{idxCLD}(2,l,m), CLD\right)$$

$$\mathbf{idxCLDEdQ}(3,l,m) =$$
$$\max\left(-15, \min\left(15, round\left(\mathbf{idxCLD}(3,l,m) \cdot facFunc\left(\mathrm{Re}lativeLocalEnergyL_{5152}(l,m)\right)\right)\right)\right)$$
$$\mathbf{D}^{Q}_{\mathrm{CLD}}(3,l,m) = deq\left(\mathbf{idxCLDEdQ}(3,l,m), CLD\right)$$

$$\mathbf{idxCLDEdQ}(4,l,m) =$$
$$\max\left(-15, \min\left(15, round\left(\mathbf{idxCLD}(4,l,m) \cdot facFunc\left(\mathrm{Re}lativeLocalEnergyR_{5152}(l,m)\right)\right)\right)\right)$$
$$\mathbf{D}^{Q}_{\mathrm{CLD}}(4,l,m) = deq\left(\mathbf{idxCLDEdQ}(4,l,m), CLD\right)$$

The function *facFunc*() is defined according to below:

$$facFunc(-x) = \begin{cases} y_1 & , x \le x_1 \\ \dfrac{(x - x_1)(y_2 - y_1)}{(x_2 - x_1)} + y_1 & , x_1 < x \le x_2 \\ y_2 & x_2 < x \end{cases}$$

Where x_1, x_2, y_1 and y_2 are defined according to the Table 85 which depends on the bitstream element *bsQuantMode*. Furthermore Figure 14 illustrates the function *facFunc*().

Table 85 — facFunc parameters table

bsQuantMode	x_1	x_2	y_1	y_2
0	1	1	1	1
1	1	21	1	5
2	1	25	1	8
3	reserved	reserved	reserved	reserved

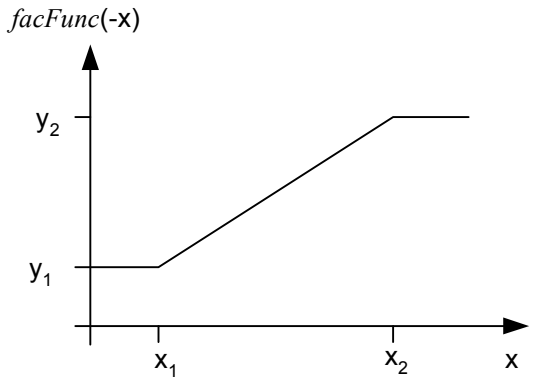

Figure 14 — Visualization of *facFunc*()

The ATD parameter is transmitted for each OTT box in the Arbitrary Trees and hence is only available if arbitrary tree data is present. Dequantization of the ATD parameter type is defined below:

$$\mathbf{D}^{Q}_{\mathrm{ATD}}(atd,l,m) = deq\left(\mathbf{idxATD}(atd,l,m), CLD\right) \quad ,0 \le atd < numOttBoxesAT$$

The ICC parameter can be transmitted either for each OTT box separately, or one combined left/right ICC parameter can be transmitted for all OTT boxes. Additionally a ICC parameter is transmitted for the TTT box if $\mathbf{bsTttModeLow}(ti) < 2$ or $\mathbf{bsTttModeHigh}(ti) < 2$. Dequantization of the ICC parameter type is defined below:

$$\mathbf{D}^{Q}_{\mathrm{ICC}}(pi,l,m) = \begin{cases} deq\left(\mathbf{idxICCUpd}(0,l,m), ICC\right) & ,bsOneICC = 1, 0 \le pi < off \\ deq\left(\mathbf{idxICCUpd}(pi,l,m), ICC\right) & ,bsOneICC = 0, 0 \le pi < off \\ deq\left(\mathbf{idxICCUpd}(off + (ti)\cdot 4,l,m), ICC\right) & ,\begin{cases} off \le pi, \mathbf{bsTttModeLow}(ti) < 2 \\ i(ti,m) = low \end{cases} \\ deq\left(\mathbf{idxICCUpd}(off + (ti)\cdot 4,l,m), ICC\right) & ,\begin{cases} off \le pi, \mathbf{bsTttModeHigh}(ti) < 2 \\ i(ti,m) = high \end{cases} \end{cases}$$

where $ti = pi - numOttBoxes$, $off = numOttBoxes$ and

$$i(ti,m) = \begin{cases} high & ,\mathbf{bsTttBandsLow}(ti) \le m < \mathbf{bsTttBandsHigh}(ti), \mathbf{bsTttDualMode}(ti) = 1 \\ low & ,otherwise \end{cases}$$

For each TTT box the parameter bands can be divided into two regions and for each region either CPC parameters or CLD parameters are transmitted. The dequantization of these parameters are defined according to:

$$\mathbf{D}^{Q}_{\mathrm{CPC\_1}}(ti,l,m) = \begin{cases} deq\left(\mathbf{idxCPC}(off + 4\cdot ti,l,m), CPC\right) & ,\mathbf{bsTttModeLow}(ti) < 2, i(ti,m) = low \\ deq\left(\mathbf{idxCPC}(off + 4\cdot ti + 2,l,m), CPC\right) & ,\mathbf{bsTttModeHigh}(ti) < 2, i(ti,m) = high \end{cases}$$

$$\mathbf{D}^{Q}_{\mathrm{CPC\_2}}(ti,l,m) = \begin{cases} deq\left(\mathbf{idxCPC}(off + 4\cdot ti + 1,l,m), CPC\right) & ,\mathbf{bsTttModeLow}(ti) < 2, i(ti,m) = low \\ deq\left(\mathbf{idxCPC}(off + 4\cdot ti + 3,l,m), CPC\right) & ,\mathbf{bsTttModeHigh}(ti) < 2, i(ti,m) = high \end{cases}$$

$$\mathbf{D}^{Q}_{\mathrm{CLD\_1}}(ti,l,m) = \begin{cases} deq\left(\mathbf{idxCLD}(off + 4\cdot ti,l,m), CLD\right) & ,2 \le \mathbf{bsTttModeLow}(ti), i(ti,m) = low \\ deq\left(\mathbf{idxCLD}(off + 4\cdot ti + 2,l,m), CLD\right) & ,2 \le \mathbf{bsTttModeHigh}(ti), i(ti,m) = high \end{cases}$$

$$\mathbf{D}^{Q}_{\mathrm{CLD\_2}}(ti,l,m) = \begin{cases} deq\left(\mathbf{idxCLD}(off + 4\cdot ti + 1,l,m), CLD\right) & ,2 \le \mathbf{bsTttModeLow}(ti), i(ti,m) = low \\ deq\left(\mathbf{idxCLD}(off + 4\cdot ti + 3,l,m), CLD\right) & ,2 \le \mathbf{bsTttModeHigh}(ti), i(ti,m) = high \end{cases}$$

for $0 \le ti < numTttBoxes$ where $off = numOttBoxes$ and

$$i(ti,m) = \begin{cases} high & ,\mathbf{bsTttBandsLow}(ti) \le m < \mathbf{bsTttBandsHigh}(ti), \mathbf{bsTttDualMode}(ti) = 1 \\ low & ,otherwise \end{cases}$$

The arbitrary downmix gain parameters are dequantized using the CLD parameter dequantization table and are defined according to:

$$\mathbf{G}^{\mathrm{Q}}(ic,l,m) = deq\left(\mathbf{idxCLD}(off + ic,l,m), CLD\right)$$

for $0 \le ic < numInChan$, where $off = numOttBoxes + 4 \cdot numTttBoxes$

6.1.9 Decoding of tree description

In case Arbitrary Trees are used (i.e., arbitrary tree data is present) the configuration of the trees are described by $bsOttBoxPresent[ch][]$, where ch ($0 \le ch < numOutChan$) is the channel index of the output vector x for the predefined trees. Figure 15 graphically shows how to derive a arbitrary sub-tree using the bit-string example $bsOttBoxPresent[0][]$ ="11100010100".

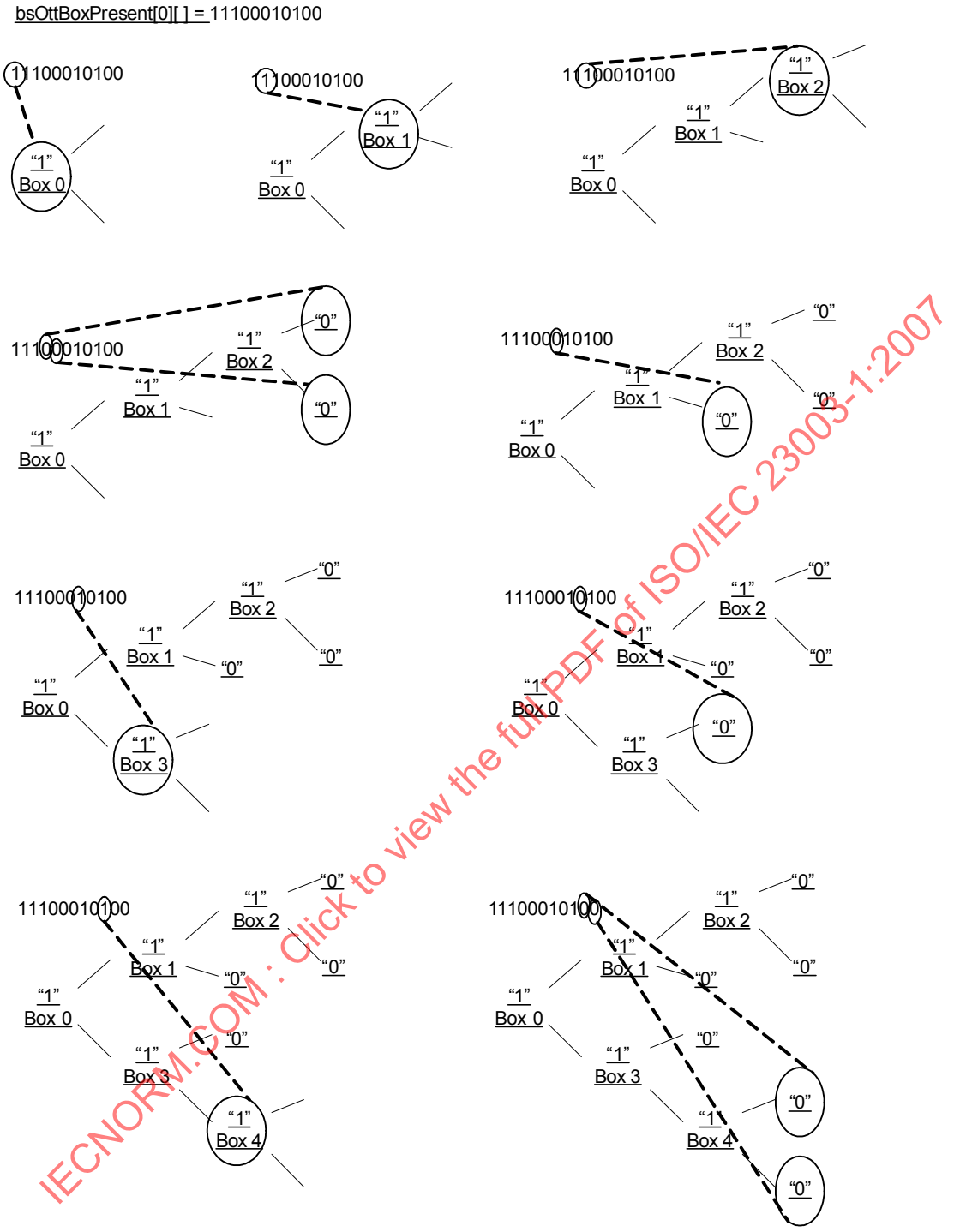

Figure 15 — Graphical representation of decoding an Arbitrary Tree description

The Arbitrary Trees which are fully described by *bsOttBoxPresent*[][] is decoded to the intermediate helper variables $\mathbf{Tree}_{sign}$, $\mathbf{Tree}_{depth}$, $\mathbf{Tree}_{outChan}$ and $\mathbf{Tree}$. by following the flowchart shown in Figure 17. But to further exemplify the definition of the helper variables the *bsOttBoxPresent*[0][ ] = 11100010100 example is used once more as shown below in Figure 16.

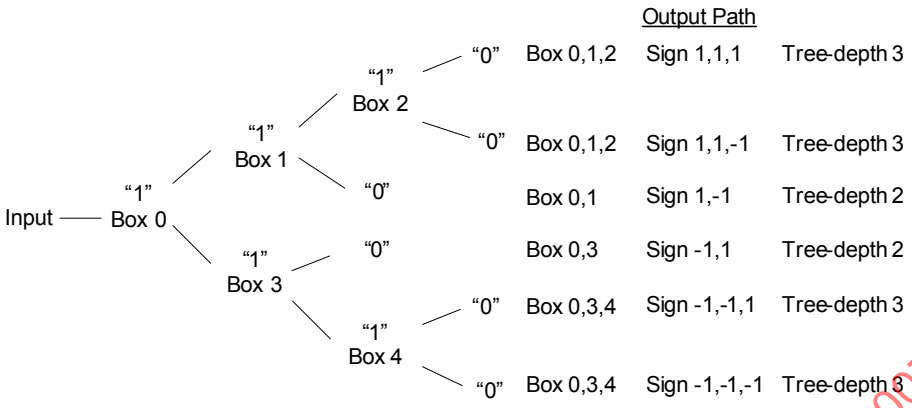

Figure 16 — Graphical representation of the relation between the tree descriptive variables, and the signaled tree.

Which results in the helper variables $\mathbf{Tree}_{\text{sign}}$, $\mathbf{Tree}_{\text{depth}}$, $\mathbf{Tree}_{\text{outChan}}$ and $\mathbf{Tree}$ as below.

As given by Figure 16, the first (0^{th}) output signal (top of the tree) is based on a signal that has been passed through OTT modules 0, 1, and 3, as given by column 0 in the matrix.

$$\mathbf{Tree}_{\text{sign}}(0,,) = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & n/a & n/a & 1 & -1 \end{bmatrix}$$

The matrix holding a column for each output signal of the sub-tree indicating whether the upper (1) or the lower (-1) output of an OTT module should be followed to reach the output signal. Hence, as given by Figure 16, the first (0^{th}) output signal (top of the tree) is based on a signal that has been passed through the upper path of OTT modules 0, 1, and 3, as given by column 0 in the matrix.

$$\mathbf{Tree}_{\text{depth}}(0,) = [3 \quad 3 \quad 2 \quad 2 \quad 3 \quad 3]$$

The number of OTT modules that are passed for every output channel as given by Figure 16. The output channels calculated from top to bottom are given from left to right in the vector.

$$\mathbf{Tree}_{\text{outChan}}(0) = [6]$$

the total number of output channels for the single sub-tree is 6.

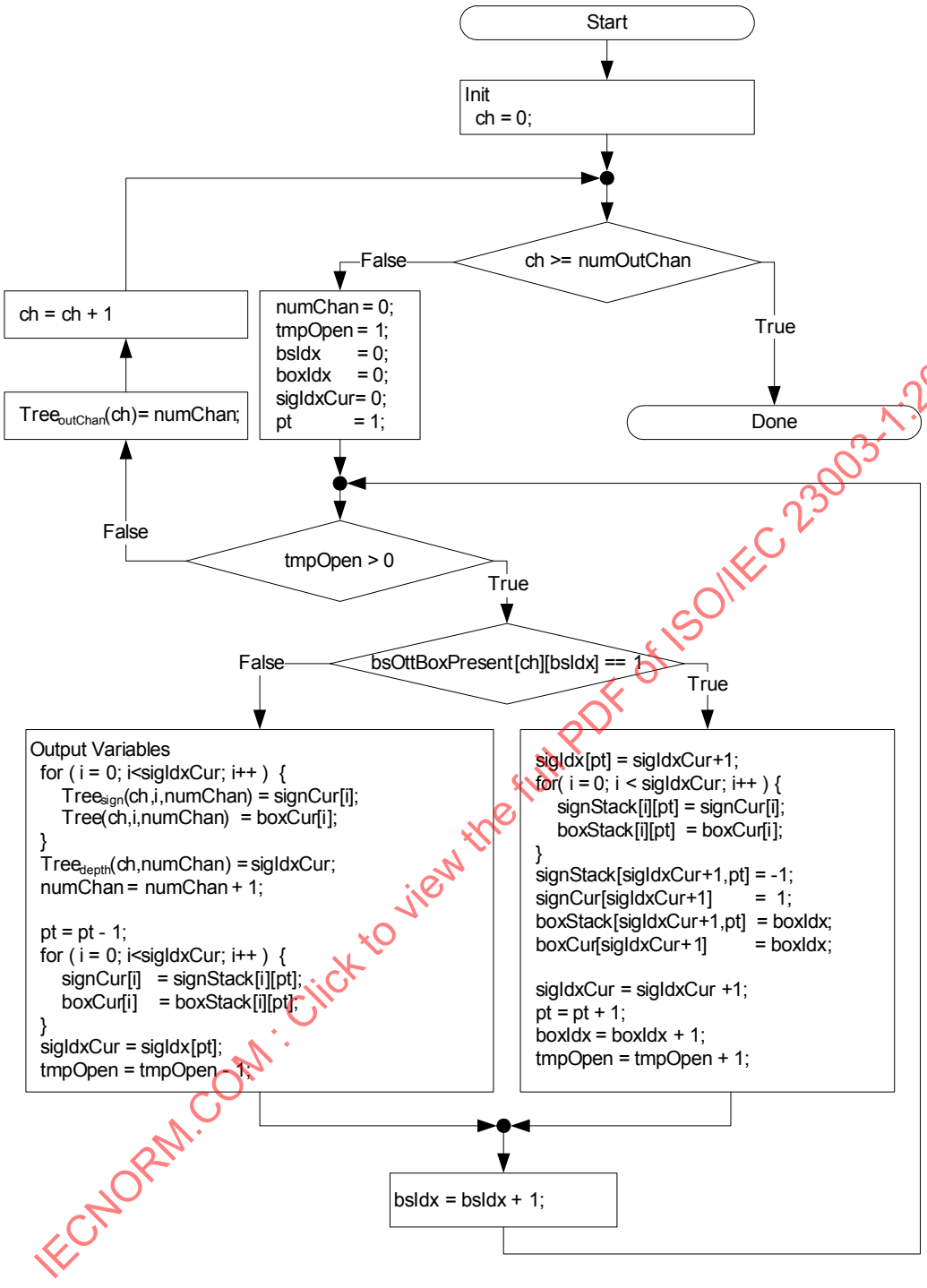

Figure 17 — Flowchart for decoding help variables from bsOttBoxPresent

6.1.10 Decoding and dequantization of guided envelope shaping (GES) data

The envelope ratios used for the guided envelope shaping are obtained from the transmitted reshaping data (signed quantization indices) according to

$$\mathbf{envRatio}_{ch}(n) = 2^{\frac{\mathbf{bsEnvShapeData}[ch][n]}{2}},$$

with $0 \le n < numSlots\text{-}1$ and ch denoting the output channel according to Table 67.

6.1.11 Parameter smoothing

6.1.11.1 Introduction

In order to account for coarse quantization and low up-date rate of spatial parameters (CLD, ICC and CPC), smoothing can be applied. The MPEG Surround decoder performs the smoothing on the matrices resulting from the received parameters rather than directly on the parameters. The resulting effect is the same. The smoothing is performed on the matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ by first order IIR filtering of the parameter bands, for which smoothing is active. The actual smoothing process as well as those matrices are defined in subclauses 6.5.2 and 6.5.3 for matrix $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively.

6.1.11.2 Smoothing flags

Depending on the *bsSmoothControl* smoothing is disabled or enabled. For each parameter band and parameter set a smoothing on/off flag is determined as follows:

$$\mathbf{S}(l,m) = \mathbf{smgData}[l][m],$$

for all parameter sets $0 \le l < L$ and all parameter bands $0 \le m < M_{\text{par}}$.

6.1.11.3 Time constant and filter coefficient

For encoder controlled smoothing one of four time constants can be signaled in the bitstream, while the automatic mode uses a fixed value of 256 time slots:

$$\tau(l) = \mathbf{smoothTime}(l),$$

where $\mathbf{smoothTime}(l)$ is defined by $\mathbf{bsSmoothTime}(l)$ according to Table 64.

As the number of time slots between two subsequent parameter sets is variable, the filter coefficient is determined by

$$\mathbf{s}_{\text{delta}}(l) = \frac{d(l)}{\tau(l)},$$

where

$$d(l) = \begin{cases} t(l)+1 & ,l=0 \\ t(l)-t(l-1) & ,l>0 \end{cases}.$$

6.1.12 Mapping of parameters to processing frequency resolution

This subclause describes the mapping of the smoothed and non-smoothed parameters from the frequency resolution of the bitstream data M_{par} to the frequency resolution of the signal processing M_{proc} being 28.

The parameters are mapped to $M_{\text{proc}} = 28$ bands using to the function $mapfunc\left(m, M_{\text{par}}\right)$, which is defined in Table 86. The mapping function is used as follows:

$$\mathbf{D}_{YYY}\left(i,l,m\right) = \mathbf{D}^{\text{Q}}_{YYY}\left(i,l,mapfunc\left(m,M_{\text{par}}\right)\right)$$

$$\mathbf{G}\left(i,l,m\right) = \mathbf{G}^{\text{Q}}\left(i,l,mapfunc\left(m,M_{\text{par}}\right)\right)$$

for $0 \le m < M_{\text{proc}}$ and all YYY, i and l.

The same mapping procedure should be applied to the smoothing flag matrix $\mathbf{S}\left(l,m\right)$, resulting in the new matrix $\mathbf{S}_{\text{proc}}\left(l,m\right)$, as well as the vectors $\mathbf{m}_{\text{resPar}}$, ***bsTttBandsLow*** and ***bsTttBandsHigh***, resulting in the new vectors $\mathbf{m}_{\text{resProc}}$, $\mathbf{m}_{\text{tttLowProc}}$ and $\mathbf{m}_{\text{tttHighProc}}$, respectively.

Table 86 — Mapping function $mapfunc\left(m, M_{\text{par}}\right)$

m	$mapfunc\left(m, M_{\text{par}}\right)$						
	M_{par} =28	M_{par} =20	M_{par} =14	M_{par} =10	M_{par} =7	M_{par} =5	M_{par} =4
0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0
2	2	2	1	1	0	0	0
3	3	3	1	1	0	0	0
4	4	4	2	2	1	1	0
5	5	5	3	2	1	1	0
6	6	6	4	3	2	1	1
7	7	7	4	3	2	1	1
8	8	8	5	4	2	2	1
9	9	9	6	4	3	2	1
10	10	10	6	5	3	2	1
11	11	11	7	5	3	2	2
12	12	12	7	6	3	2	2
13	13	13	8	6	4	2	2
14	14	14	8	7	4	3	2
15	15	14	8	7	4	3	2
16	16	15	9	7	4	3	2
17	17	15	9	7	4	3	2
18	18	16	10	8	5	3	2
19	19	16	10	8	5	3	2
20	20	17	11	8	5	3	2
21	21	17	11	8	5	3	2
22	22	18	12	9	6	4	3
23	23	18	12	9	6	4	3
24	24	18	12	9	6	4	3
25	25	19	13	9	6	4	3
26	26	19	13	9	6	4	3
27	27	19	13	9	6	4	3

6.1.13 Requirements

The following requirements apply to the spatial audio parameters. **bsXXXdataMode**[*numParamSets*-1] shall not have the value 2.

The allowed values for *bsResidualFramesPerSpatialFrame* or *bsArbitraryDownmixResidualFramesPerSpatialFrame* depend on *numSlots*. Table 87 shows the allowed values for *bsResidualFramesPerSpatialFrame* or *bsArbitraryDownmixResidualFramesPerSpatialFrame* as a function of *numSlots*.

Table 87 — Allowed combinations of numSlots and *bsResidualFramesPerSpatialFrame* or *bsArbitraryDownmixResidualFramesPerSpatialFrame*

numSlots	*bsResidualFramesPerSpatialFrame* **or** *bsArbitraryDownmixResidualFramesPerSpatialFrame*
15	0
16	0
18	0
24	0
30	0
32	0, 1
36	1
48	1
60	1
64	1, 3
72	2

The allowed values for *bsResidualSamplingFrequencyIndex* or *bsArbitraryDownmixResidualSamplingFrequencyIndex* depend on *bsFrameLength*, *bsSamplingFrequencyIndex* and *bsResidualFramesPerSpatialFrame* or *bsArbitraryDownmixResidualFramesPerSpatialFrame*, respectively, as shown in Table 88.

Table 88 — Allowed combinations of *bsSamplingFrequencyIndex* and *bsResidualSamplingFrequencyIndex* or *bsArbitraryDownmixResidualSamplingFrequencyIndex*

(*bsFrameLength*+1)/ (*bsResidualFramesPerSpatialFrame*+1) or (*bsFrameLength*+1)/ (*bsArbitraryDownmixResidualFramesPerSpatialFrame*+1)	Allowed combinations of {*bsSamplingFrequencyIndex*, *bsResidualSamplingFrequencyIndex*} or {*bsSamplingFrequencyIndex*, *bsArbitraryDownmixResidualSamplingFrequencyIndex*}
15	{0x0, 0x0}, {0x1, 0x1}, {0x2, 0x2}, {0x3, 0x3}, {0x4, 0x4}, {0x5, 0x5}, {0x6, 0x6}, {0x7, 0x7}, {0x8, 0x8}, {0x9, 0x9}, {0xa, 0xa} and {0xb, 0xb}
16	{0x0, 0x0}, {0x1, 0x1}, {0x2, 0x2}, {0x3, 0x3}, {0x4, 0x4}, {0x5, 0x5}, {0x6, 0x6}, {0x7, 0x7}, {0x8, 0x8}, {0x9, 0x9}, {0xa, 0xa} and {0xb, 0xb}
18	{0x0, 0x2}, {0x1, 0x2}, {0x2, 0x3}, {0x3, 0x5}, {0x4, 0x5}, {0x5, 0x6}, {0x6, 0x8}, {0x7, 0x8}, {0x8, 0x9}, {0x9, 0xb} and {0xa, 0xb}
24	{0x0, 0x3}, {0x1, 0x3}, {0x2, 0x5}, {0x3, 0x5}, {0x4, 0x5}, {0x5, 0x8}, {0x6, 0x9}, {0x7, 0x9} and {0x8, 0xb}
30	{0x0, 0x3}, {0x1, 0x3}, {0x2, 0x5}, {0x3, 0x7}, {0x4, 0x7}, {0x5, 0x8}, {0x6, 0x9}, {0x7, 0x9} and {0x8, 0xb}
32	{0x0, 0x3}, {0x1, 0x4}, {0x2, 0x5}, {0x3, 0x6}, {0x4, 0x7}, {0x5, 0x8}, {0x6, 0x9}, {0x7, 0xa}, {0x8, 0xb}, {0x9, 0xb}, {0xa, 0xb} and {0xb, 0xb}

6.2 Enhanced Matrix Mode of MPEG Surround

6.2.1 Introduction

In the Enhanced Matrix Mode configuration, the spatial parameters for the spatial synthesis stage are derived from an analysis of the received stereo down mix. The process of decoder-side parameter generation is depicted in Figure 18.

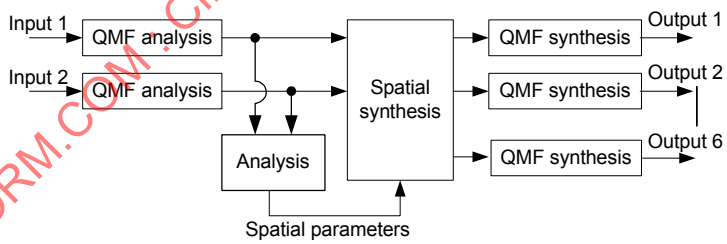

Figure 18 — Overview of parameter-less SAC extension

The analysis stage generates spatial parameters based on stereo down-mix parameters. These down-mix parameters comprise a Channel Level Difference $\left(CLD_{\mathrm{dm}}\right)$ and an Interchannel Cross Correlation $\left(CLD_{\mathrm{dm}}\right)$ parameter per processing band. STP and GES are not supported by Enhanced Matrix Mode of MPEG Surround.

The conversion from stereo input signal to spatial parameters and temporal processing flags comprises 4 steps that are executed in the following order:

1. Conversion of the current analysis states to temporal processing flags; this process is performed once per frame (at the start of a frame);

2. Conversion of the current analysis states to stereo down-mix parameters; this process is performed every 4 slots;

3. Conversion of stereo down-mix parameters to spatial parameters; this process is also performed every 4 slots (whenever step 1 is performed);

4. Update of the analysis states for a every newly received down-mix signal slot.

6.2.2 Down mix analysis states

The hybrid QMF-domain representations of the stereo input signals are given by $x_{L_0}^{n,k}$, $x_{R_0}^{n,k}$, with *n* the time slot index and *k* the hybrid frequency-band index. For each processing band *m* ($0 \le m < M_{\text{proc}}$ $M_{\text{proc}} = 28$) and each newly received signal slot *n*, three states ($P_1^{n,m}, P_2^{n,m}, P_3^{n,m}$) are updated according to:

$$P_1^{n,m} = cP_1^{n-1,m} + (1-c)\sum_{k \in K(m)} x_{L_0}^{n,k}\left(x_{L_0}^{n,k}\right)^*,$$

$$P_2^{n,m} = cP_2^{n-1,m} + (1-c)\sum_{k \in K(m)} x_{R_0}^{n,k}\left(x_{R_0}^{n,k}\right)^*$$

$$P_3^{n,m} = cP_3^{n-1,m} + (1-c)\Re\left\{\sum_{k \in K(m)} x_{L_0}^{n,k}\left(x_{R_0}^{n,k}\right)^*\right\}$$

with, $K(m)$ representing all hybrid frequency bands *k* corresponding to processing band *m*, $\Re$ the real-value operator, and *c* a low-pass constant (representing a time constant of 60 ms), given by

$$c = \exp\left\{\frac{-64}{0.06F_s}\right\},$$

with F_s the (time-domain) input sampling rate.

Furthermore, a slot-based energy Q_X^n for each slot *n* is computed from both stereo hybrid QMF-domain input signals ($x_X^{n,k}$ $x_i^{n,k}$) following

$$Q_X^n = \sum_{k=16}^{42} x_X^{n,k}\left(x_X^{n,k}\right)^*, \quad X \in \{L_0, R_0\}$$

From Q_X^n, two analysis state parameters $r_{X,1}^n$ and $r_{X,2}^n$ for $X \in \{L_0, R_0\}$ are subsequently updated according to:

$$r_{X,1}^n = c_r r_{X,1}^{n-1} + (1-c_r)\left(Q_X^n\right)^2,$$

$$r_{X,2}^n = c_r r_{X,2}^{n-1} + (1-c_r)\left(Q_X^n - Q_X^{n-1}\right)^2,$$

with

$$c_r = \exp\left\{\frac{-64}{0.3F_s}\right\}.$$

6.2.3 Stereo parameter generation

A CLD parameter $\mathbf{dm}_{\mathrm{CLD}}$ and a ICC parameter $\mathbf{dm}_{\mathrm{ICC}}$ of the stereo down mix is generated for each processing band m according to:

$$\mathbf{dm}_{\mathrm{CLD}}(m) = 10\log_{10}\left(\frac{\varepsilon + P_1^{n,m}}{\varepsilon + P_2^{n,m}}\right),$$

$$\mathbf{dm}_{\mathrm{ICC}}(m) = \frac{\varepsilon + P_3^{n,m}}{\sqrt{\left(\varepsilon + P_1^{n,m}\right)\left(\varepsilon + P_2^{n,m}\right)}},$$

with ε a small number to prevent division by zero (ε=1e-9).

6.2.4 Spatial parameter generation

The stereo parameters $\mathbf{dm}_{\mathrm{CLD}}$ and $\mathbf{dm}_{\mathrm{ICC}}$ are converted to (unrounded) table lookup indices *CLI* (Channel Level Index) $\mathbf{dm}_{\mathrm{CLI}}$ and *ICI* (Interchannel Correlation Index) $\mathbf{dm}_{\mathrm{ICI}}$ for each processing band m:

$$\mathbf{dm}_{\mathrm{CLI}}(m) = \begin{cases} 30 & ,if\ \left|\mathbf{dm}_{\mathrm{CLD}}(m)\right| > 30 \\ \left|\mathbf{dm}_{\mathrm{CLD}}(m)\right| & ,otherwise \end{cases},$$

$$\mathbf{dm}_{\mathrm{ICI}}(m) = 10ICC_{dm}(m) + 10$$

The quantizer indices of the spatial parameters required to compute the mixing matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ are obtained by table lookup T using the table lookup indices $\mathbf{dm}_{\mathrm{CLI}}$ and $\mathbf{dm}_{\mathrm{ICI}}$. These quantizer indexes for the required CPC, CLD and ICC spatial parameters are given in Table A.31. The actual parameter values are obtained using bi-linear interpolation of the dequantized parameter indexes based on the $\mathbf{dm}_{\mathrm{CLI}}$ and $\mathbf{dm}_{\mathrm{ICI}}$ table indexes:

$$\begin{aligned}\mathbf{CPC}(m) = {} & (1-w_1(m))(1-w_2(m))\mathbf{CLD}\left[T_{CPC}\left(\left\lfloor \mathbf{dm}_{\mathrm{CLI}}(m)\right\rfloor, \left\lfloor \mathbf{dm}_{\mathrm{ICI}}(m)\right\rfloor\right)\right] + \\ & w_1(m)(1-w_2(m))\mathbf{CLD}\left[T_{CPC}\left(\left\lceil \mathbf{dm}_{\mathrm{CLI}}(m)\right\rceil, \left\lfloor \mathbf{dm}_{\mathrm{ICI}}(m)\right\rfloor\right)\right] + \\ & (1-w_1(m))w_2(m)\mathbf{CLD}\left[T_{CPC}\left(\left\lfloor \mathbf{dm}_{\mathrm{CLI}}(m)\right\rfloor, \left\lceil \mathbf{dm}_{\mathrm{ICI}}(m)\right\rceil\right)\right] + \\ & w_1(m)w_2(m)\mathbf{CLD}\left[T_{CPC}\left(\left\lceil \mathbf{dm}_{\mathrm{CLI}}(m)\right\rceil, \left\lceil \mathbf{dm}_{\mathrm{ICI}}(m)\right\rceil\right)\right]\end{aligned}$$

where $\lfloor . \rfloor$ denotes the 'floor' function, $\lceil . \rceil$ the 'ceil' function, and w_1 and w_2 the interpolation weights:

$$w_1 = \mathbf{dm}_{\mathrm{CLI}}(m) - \left\lfloor \mathbf{dm}_{\mathrm{CLI}}(m)\right\rfloor,$$

$$w_2 = \mathbf{dm}_{\mathrm{ICI}}(m) - \lfloor \mathbf{dm}_{\mathrm{ICI}}(m) \rfloor .$$

This process is performed for two CPC parameters, one CLD and one ICC parameter, according to Table. A.31. These parameters are used to generate the spatial synthesis matrices M_1 and M_2, under the following constraints:

1. Generation of M_1 and M_2 is based on the '525' mode, using matrix compatibility (bsMatrixMode=1, see subclause 6.5.2.4), and a prediction based up-mix (bsTttModeLow =1, see subclause 6.5), and where $\gamma^{l,m}$ is assumed to be $\gamma^{l,m} = 1$;

2. The same CLD and ICC value is used for CLD_1 / CLD_2 and ICC_1 / ICC_2 , for both OTT boxes respectively;

3. The CPC parameters for processing band *m* have to be interchanged if $CLD_{dm}(m) < 0$.

6.2.5 Parameter processing and interpolation

The process of state updates, stereo parameter estimation, spatial parameter lookup and parameter positioning is depicted in Figure 19. A new down-mix frame is received starting at hybrid QMF down mix slot '0' up to N-1 (N=32). The down-mix states (S) are updated for every down-mix slot. Stereo parameters are estimated after a state-update of the last slot of the previous frame (slot '-1'). The stereo parameters are converted to spatial parameters using a table lookup *T*. Finally, the generated spatial parameters are applied 4 slots ahead. The process of spatial parameter generation is performed once per 4 slots. The M_1 and M_2 matrices are interpolated linearly for hybrid QMF slots in between parameter positions. The temporal processing flags for the whole frame are based on the analysis states of slot '-1'.

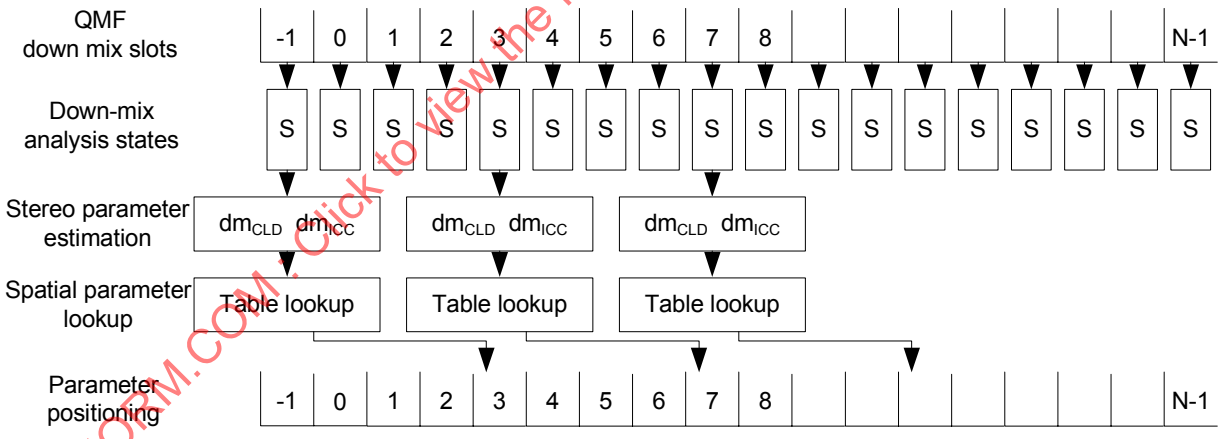

Figure 19 — Stereo down mix analysis procedure

6.3 Time / frequency transforms

6.3.1 Introduction

The MPEG Surround decoder employs hybrid QMF banks to convert time domain signals into non-uniform (hybrid) subband domain signals and vice versa. All processing of the MPEG Surround decoder is conducted in the hybrid subband domain. The hybrid QMF banks consist of two parts, the first being a uniform complex-modulated QMF bank and secondly additional oddly-modulated Nyquist filter banks. The analysis filterbank is outlined in Figure 20.

Apart from the regular mode of operation where the Spatial Audio decoder is fed with time-domain samples $\tilde{\mathbf{x}}^{v}$, also intermediate (QMF) subband domain samples $\hat{\mathbf{x}}^{n,k}$ from an HE-AAC decoder can be taken. In that case the subband domain samples prior to HE-AAC QMF synthesis are taken, as defined in ISO/IEC 14496-3 subclause 4.6.18. Hence, the vector $\hat{\mathbf{x}}^{n,k}$ holds the subsample n for the QMF subband k for all input channels. Furthermore, if enabled, the residual decoding module provides subband domain samples $\hat{\mathbf{x}}_{res}^{n,k}$ that also need to be transformed to the hybrid domain ($\mathbf{x}^{n,k}$).

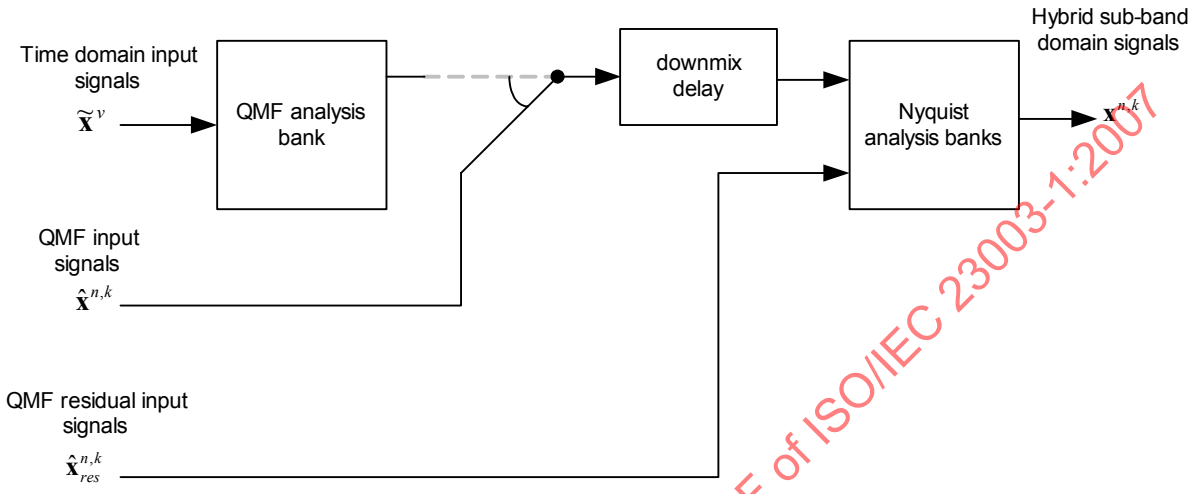

Figure 20 — Time/frequency transforms in MPEG Surround, hybrid QMF analysis bank

A delay of 5 QMF samples is inserted in the downmix signal path prior to the Nyquist analysis filterbanks, as shown in Figure 20. It is applied in order to account for the delay introduced by the real-to-complex transform utilized in the Low Power decoder. In case of enhanced matrix mode operation, no such delay is inserted.

At the synthesis side the hybrid subband domain samples $\mathbf{y}^{n,k}$ are transformed back to the time domain samples $\tilde{\mathbf{y}}^{v}$. The synthesis filterbank is outlined in Figure 21.

Figure 21 — Time/frequency transforms in MPEG Surround, hybrid QMF synthesis bank

6.3.2 Filterbanks

6.3.2.1 QMF filterbanks

If the Spatial Audio decoder is fed with time-domain samples a QMF analysis bank as defined in ISO/IEC 14496-3, subclause 4.B.18.2 is employed. However, in the equation for matrix M(k,n) and in Figure 4.B.20, the term „(2*n+1)“ has to be substituted by „(2*n-1)“. The input of the filter bank are blocks of 64 samples of each

column vector of the input signal $\tilde{\mathbf{x}}^{v} = \begin{bmatrix} \tilde{x}_1^v & \tilde{x}_2^v & \ldots & \tilde{x}_K^v \end{bmatrix}$ with *K* the number of input signals. The output of the filter bank are slots of 64 subband domain samples which are stored in $\hat{\mathbf{x}}^{n,k}$.

The QMF synthesis filter banks are identical to the 64 bands complex QMF synthesis filter banks as defined in ISO/IEC 14496-3 subclause 4.6.18.4.2. The input to the filter banks are slots consisting of 64 subband domain samples of $\hat{\mathbf{y}}^{n,k}$. For each slot the filter bank outputs one block of 64 time domain samples of $\tilde{\mathbf{y}}^{v}$.

6.3.2.2 Nyquist filterbanks

The additional filter banks are applied on the lower sub bands, and gives in total a number of 71 hybrid sub bands. This 71 bands configuration is identical to that used for the 20 stereo-bands configuration in Parametric Stereo as defined in ISO/IEC 14496-3. Hence, the output of the QMF filter bank, for all input channels $\hat{\mathbf{x}}^{n,k}$ are mapped to, either by hybrid filtering or by delay, to $\mathbf{x}^{n,k}$ as outlined in subclause 8.6.4.3 of ISO/IEC 14496-3.

The Nyquist synthesis filter banks is identical to that used for the 20 stereo-bands configuration in Parametric Stereo as defined in ISO/IEC 14496-3. The vector $\mathbf{y}^{n,k}$ (or $\mathbf{z}^{n,k}$ in the case of Arbitrary Trees) holding the output channels are synthesized with the hybrid filterbank according to subclause 8.6.4.3 in ISO/IEC 14496-3 forming the input $\hat{\mathbf{y}}^{n,k}$ to the QMF synthesis filterbanks.

6.3.3 Support for lower and higher sampling frequencies

Support for low sampling frequencies, that is if *bsSamplingFrequency* < 27713, and for high sampling frequencies, that is if *bsSamplingFrequency* >= 55426, is provided by using downsampled 32 band QMF banks or upsampled 128 band QMF banks, respectively, instead of the 64 band QMF banks that are used for normal sampling frequencies like 32, 44.1, or 48 kHz.

For downsampled operation, the 32 band QMF analysis bank is defined in ISO/IEC 14496-3 subclause 4.6.18.4.1. The 32 band QMF synthesis bank is defined in ISO/IEC 14496-3 subclause 4.6.18.4.3. The upper 32 QMF bands are set to zero prior to MPEG Surround processing and only the lower 32 QMF bands are processed by the QMF synthesis bank.

For upsampled operation, the 128 band QMF analysis bank is defined by replacing the modulation of the 64 band QMF analysis bank with

$$\exp\left(i\frac{pi}{256}(k+0.5)(2n-2) \right), \quad 0 \le k < 128, 0 \le n < 256$$

using a 1280 sample version of the window function c[i] where the additional intermediate samples are obtained by linear interpolation of neighboring samples of the original 640 sample window function specified in ISO/IEC 14496-3 subclause 4.A.6.2 Table 4.A.87. The 128 band QMF synthesis bank is defined by replacing the modulation of the 64 band QMF synthesis bank with

$$\exp\left(i\frac{pi}{256}(k+0.5)(2n-510) \right), \quad 0 \le k < 128, 0 \le n < 256$$

using a 1280 sample version of the window function c[i] as defined above. The upper border of the highest processing band is moved from the 64th QMF band to the 128th QMF band. Decorrelation (see subclause 5.5.2) for the upper 64 QMF bands is done by means of a 1 QMF sample delay, and no energy-shaping gain vector is applied to the decorrelated signal in the upper 64 QMF bands.

For downsampled and upsampled operation of MPEG Surround in combination with an HE-AAC coded downmix signal, it is necessary to connect the HE-AAC decoder to the MPEG Surround decoder in the time domain.

6.4 MPEG Surround processing overview

6.4.1 Introduction

The following subclauses give a general overview of the processing of the input signals to form the output signals based on the spatial parameters. The whole processing is defined by two matrix multiplications, the first forming the input signals to the decorrelation units and also performing the two-to-three channel upmix for stereo-downmix signals, and the second forming the output signals based on the downmix input and the output from the decorrelators.

The general concept is outlined in e.g. Figure 22 for the 5-1-5 configuration and Figure 25 for the 5-2-5 configuration. Alternative visualizations are given in e.g. Figure 23 and Figure 26, where the multi-channel reconstruction is outlined as a tree structure built up from OTT (One-To-Two) and TTT (Two-To-Three) boxes. The OTT boxes are conceptual boxes re-creating two channels from one input channel and a decorrelator providing a decorrelated version of the signal being input to the OTT box. Every decorrelator can, for certain frequency regions as indicated by the compressed data stream, be replace by a residual signal. The residual signal is coded in the MDCT domain, and transformed to the QMF domain by the transformation outlined in subclause 6.9. The TTT boxes are conceptual boxes re-creating three channels from two input channels. The two-to-three upmix is done differently (prediction based upmix or energy based upmix) depending on the data stream since different coding or processing of the downmix signal being input to the TTT box requires different upmix methods. The TTT box also includes a decorrelator that can by means of signalling in the data stream by activated. When a prediction based 2 to 3 channel upmix is used, this decorrelator compensates for the prediction error, by replacing the missing signal energy by decorrelated signal. Alternatively, the prediction error is compensated for by means of a residual signal.

Returning to the matrix visualization, it is clear that the first matrix strives to create input signals to the decorrelators that have the same level as the input signals to the OTT boxes of which the decorrelators conceptually are part of, and performs the two-to-three upmix for the 5-2-5 configuration. This allows for a tree structured parameterization while having a "flat" matrix based processing.

Given this, the processing of the input channels to form the output channels can be described according to:

$$\mathbf{v}^{n,k} = \mathbf{M}_1^{n,k}\mathbf{x}^{n,k}$$

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k}$$

$\mathbf{M}_1^{n,k}$ is a matrix (or for the 5-1-5 configurations a vector) mapping a certain number of input channels to a certain number of channels going into the decorrelators, and is defined for every time-slot n, and every hybrid subband k. $\mathbf{M}_2^{n,k}$ is a matrix mapping a certain number of pre-processed channels to a certain number of output channels, and is defined for every time-slot n, and every hybrid subband k.

If temporal shaping (STP or GES) of the upmixed signal is used, the $\mathbf{w}^{n,k}$ vector comes in two versions, $\mathbf{w}_{direct}^{n,k}$ and $\mathbf{w}_{diffuse}^{n,k}$. Based on these, two temporary output vectors are derived, one holding the direct signal and one holding the diffuse signal, according to:

$$\begin{cases} \mathbf{y}_{\text{direct}}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}_{\text{direct}}^{n,k} \\ \mathbf{y}_{\text{diffuse}}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}_{\text{diffuse}}^{n,k} \end{cases}$$

The output is derived from these as outlined in subclause 6.8, if the GES tool is used, and subclause 6.7 if the STP tool is used, as indicated by data stream element bsTempShapeConfig.

In the following subclauses the vectors $\mathbf{x}^{n,k}$, $\mathbf{v}^{n,k}$, $\mathbf{w}^{n,k}$, and $\mathbf{y}^{n,k}$ are defined for the different configurations. The $\mathbf{M}_1^{n,k}$ and $\mathbf{M}_2^{n,k}$ matrices are defined for the different configurations in subclause 6.5.

6.4.2 The 5-1-5 configuration

6.4.2.1 Introduction

There are two different 5-1-5 configurations given, 5-1-5_1 and 5-1-5_2. They provide different advantages for different signal-types or operation conditions. For both configurations, the input vector to be multiplied by $\mathbf{M}_1^{n,k}$ is a vector containing the downmix mono channel. Four decorrelators are used, and the outputs of the decorrelators are depending on the bitstream replaced by residual signals for certain frequency regions. No decorrelation is used for the separation of the centre channel and the LFE channel, and no residual signal can be inserted for this OTT box.

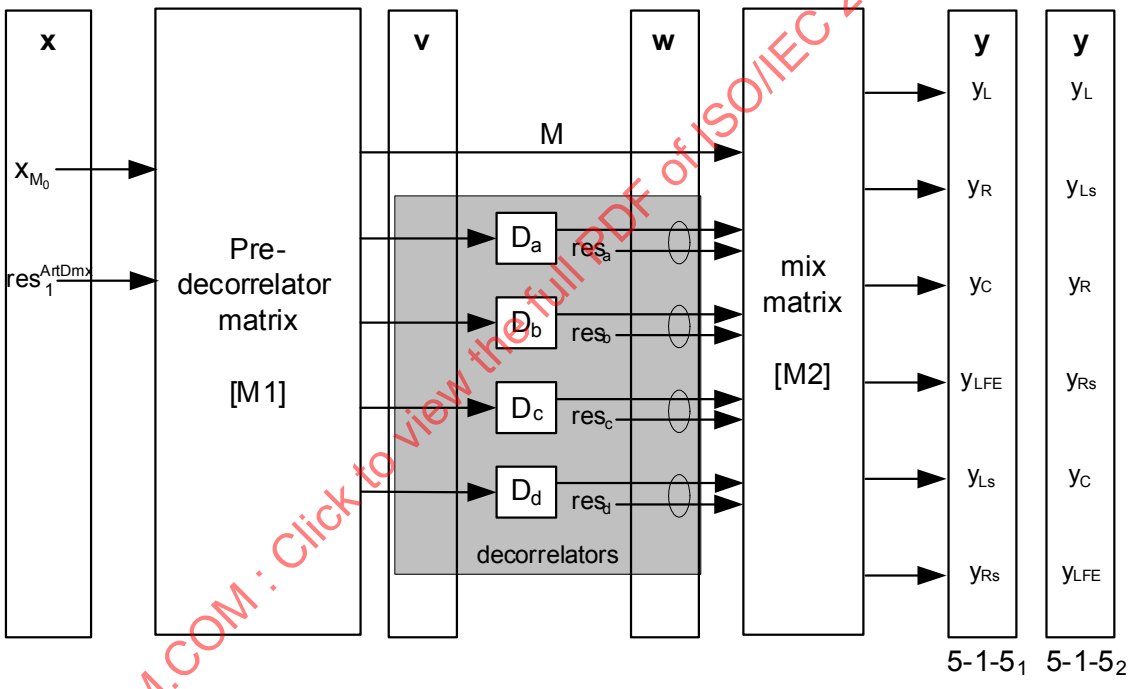

Figure 22 — Matrix view of the spatial audio processing for the 5-1-5 configuration

The decorrelators and residual signals in Figure 22 (labelled “a” to “d”) correspond to different OTT boxes depending on configuration.

The multi-channel reconstruction for the 5-1-5 configuration can also be visualized by means of a tree-structure. This is outlined in Figure 23 and Figure 24. In Figure 23, every OTT box re-creates two channels based on one input channel, the corresponding CLD and ICC parameters, and residual signal. The OTT boxes and the corresponding data are numbered corresponding to the order they appear in the bitstream.

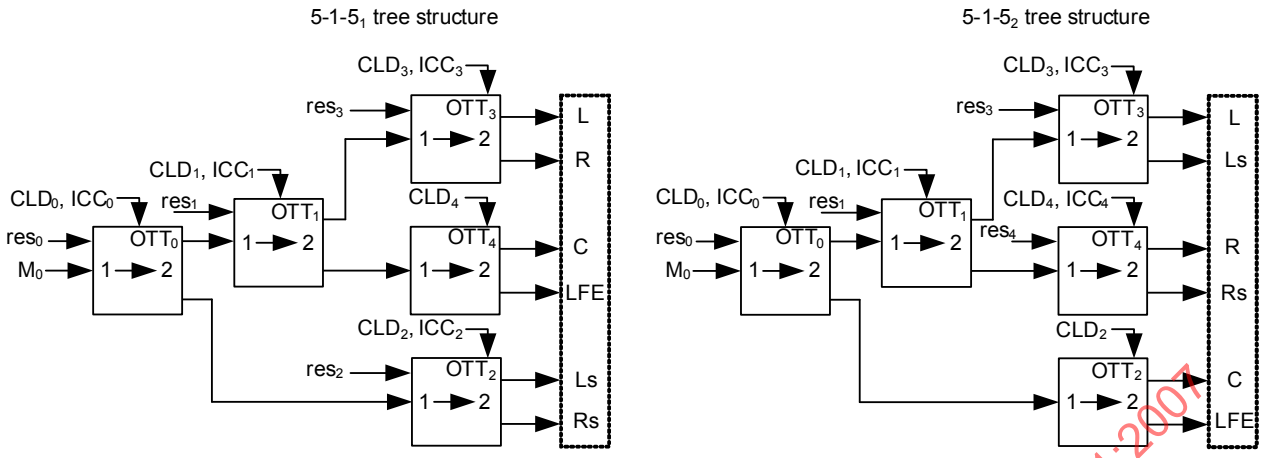

Figure 23 — Tree structure view of the spatial audio processing for the 5-1-5 configurations

The parameterization used for the different 5-1-5 configurations are outlined in Figure 24. Again, the CLD and ICC parameters are labelled corresponding to the order they appear in the bitstream.

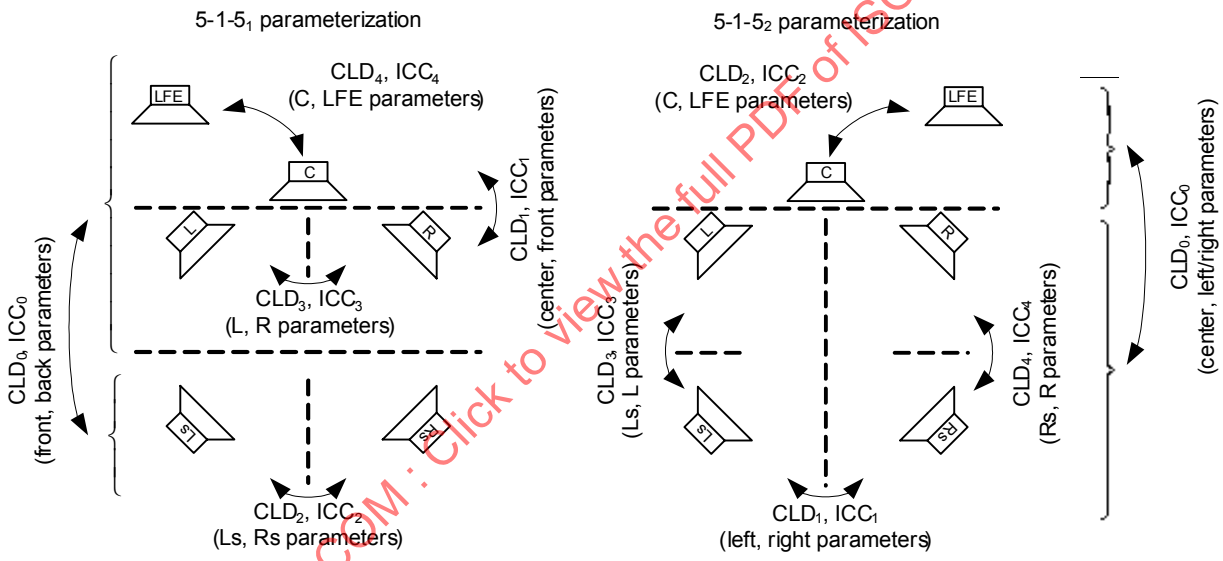

Figure 24 — Parameterization view of the spatial audio processing for the 5-1-5 configurations

The definitions of the vectors and matrices depend on which 5-1-5 configuration is used. The matrixes $\mathbf{M}_1^{n,k}$ and $\mathbf{M}_2^{n,k}$ are defined accordingly in subclause 6.5, while the vectors to be multiplied with the matrices in order to form the output are defined in the following subclauses.

6.4.2.2 Vector definitions for the 5-1-5_1 configuration

6.4.2.2.1 Operation without temporal shaping tools

For the 5-1-5_1 configuration, the input signals to the decorrelators are defined by $\mathbf{v}^{n,k}$, which is derived from the input vector $\mathbf{x}^{n,k}$ and the matrix $\mathbf{M}_1^{n,k}$ having 5 rows and 1 column, according to:

$$\mathbf{v}^{n,k} = \mathbf{M}_1^{n,k}\mathbf{x}^{n,k} = \mathbf{M}_1^{n,k}\begin{bmatrix} x_{\mathrm{M}_0}^{n,k} \\ x_{\mathrm{res}_1}^{n,k} \end{bmatrix}^{\mathrm{ArtDmx}} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ v_0^{n,k} \\ v_1^{n,k} \\ v_3^{n,k} \\ v_2^{n,k} \end{bmatrix}$$

The subscripts for the different elements in the $\mathbf{v}^{n,k}$ vector indicate which OTT box decorrelator the signal is input to, with the exception of $v_{\mathrm{M}_0}^{n,k}$, which is the direct signal.

The vector $\mathbf{w}^{n,k}$ holding the direct signal, decorrelated signals, and the residual signals is defined according to:

$$\mathbf{w}^{n,k} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ \delta_0(k) D_0\left(v_0^{n,k}\right) + \left(1-\delta_0(k)\right) v_{\mathrm{res}0}^{n,k} \\ \delta_1(k) D_1\left(v_1^{n,k}\right) + \left(1-\delta_1(k)\right) v_{\mathrm{res}1}^{n,k} \\ \delta_3(k) D_3\left(v_3^{n,k}\right) + \left(1-\delta_3(k)\right) v_{\mathrm{res}3}^{n,k} \\ \delta_2(k) D_2\left(v_2^{n,k}\right) + \left(1-\delta_2(k)\right) v_{\mathrm{res}2}^{n,k} \end{bmatrix} = \begin{bmatrix} w_{\mathrm{M}_0}^{n,k} \\ w_0^{n,k} \\ w_1^{n,k} \\ w_3^{n,k} \\ w_2^{n,k} \end{bmatrix}$$

where $\delta_x(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{\mathrm{set}}\} \\ 1 & ,otherwise \end{cases}$ and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{\mathrm{resProc}}(X)$ given by Table A.31, and where $D_X\left(v_X^{n,k}\right)$ is the output from decorrelator D_X given the input signal $v_X^{n,k}$.

The subscripts for the different elements indicate which OTT box the signal corresponds to the numbering of OTT boxes for the 5-1-5_1 configuration as given by Figure 23. Hence, $D_X\left(v_X^{n,k}\right)$ is the decorrelator output from box OTT$_X$ and $v_{\mathrm{res}_X}^{n,k}$ is the corresponding residual signal.

The subband output signals are subsequently defined for every time-slot n, and every hybrid subband k, by $\mathbf{y}^{n,k}$, which is derived from the vector $\mathbf{w}^{n,k}$ and the matrix $\mathbf{M}_2^{n,k}$ having 6 rows and 5 columns, according to

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k}\begin{bmatrix} w_{\mathrm{M}_0}^{n,k} \\ w_0^{n,k} \\ w_1^{n,k} \\ w_3^{n,k} \\ w_2^{n,k} \end{bmatrix} = \begin{bmatrix} y_{\mathrm{L}}^{n,k} \\ y_{\mathrm{R}}^{n,k} \\ y_{\mathrm{C}}^{n,k} \\ y_{\mathrm{LFE}}^{n,k} \\ y_{\mathrm{Ls}}^{n,k} \\ y_{\mathrm{Rs}}^{n,k} \end{bmatrix}.$$

The elements of $\mathbf{M}_2^{n,k}$ are defined in subclause 6.5.3, and the hybrid subband signals defined in $\mathbf{y}^{n,k}$ are synthesized to the time-domain by the hybrid synthesis filterbank as defined in subclause 6.3.

6.4.2.2.2 Operation with temporal shaping tools

If temporal shaping is used, the vector $\mathbf{v}^{n,k}$ is defined identically to the previous subclause, however two $\mathbf{w}^{n,k}$ vectors are defined. The first, $\mathbf{w}_{\mathrm{direct}}^{n,k}$ holds the direct signal and the residual signals, while the second $\mathbf{w}_{\mathrm{diffuse}}^{n,k}$ holds the decorrelator output signals, according to:

$$\mathbf{w}_{\mathrm{direct}}^{n,k} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ \left(1-\delta_0(k)\right)v_{\mathrm{res0}}^{n,k} \\ \left(1-\delta_1(k)\right)v_{\mathrm{res1}}^{n,k} \\ \left(1-\delta_3(k)\right)v_{\mathrm{res3}}^{n,k} \\ \left(1-\delta_2(k)\right)v_{\mathrm{res2}}^{n,k} \end{bmatrix} = \begin{bmatrix} w_{\mathrm{M}_0}^{n,k} \\ w_0^{n,k} \\ w_1^{n,k} \\ w_3^{n,k} \\ w_2^{n,k} \end{bmatrix}$$

$$\mathbf{w}_{\mathrm{diffuse}}^{n,k} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ \delta_0(k)D_0\left(v_0^{n,k}\right) \\ \delta_1(k)D_1\left(v_1^{n,k}\right) \\ \delta_3(k)D_3\left(v_3^{n,k}\right) \\ \delta_2(k)D_2\left(v_2^{n,k}\right) \end{bmatrix} = \begin{bmatrix} w_{\mathrm{M}_0}^{n,k} \\ w_0^{n,k} \\ w_1^{n,k} \\ w_3^{n,k} \\ w_2^{n,k} \end{bmatrix}$$

where $\delta_X(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{\mathrm{set}}\} \\ 1 & ,otherwise \end{cases}$ and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{\mathrm{resProc}}(X)$ given by Table A.31, and where $D_X\left(v_X^{n,k}\right)$ is the output from decorrelator D_X given the input signal $v_X^{n,k}$. The subscripts are used as outlined in the previous subclause.

Two temporary output vectors are derived, $\mathbf{y}_{\mathrm{direct}}^{n,k}$ holding the direct signal, and $\mathbf{y}_{\mathrm{diffuse}}^{n,k}$ holding the diffuse signal. They are calculated from $\mathbf{w}_{\mathrm{direct}}^{n,k}$ and $\mathbf{w}_{\mathrm{diffuse}}^{n,k}$, using $\mathbf{M}_2^{n,k}$ which is identical to that used if no temporal shaping is applied. The output is derived from these as outlined in subclause 6.7, if the STP tool is used, and subclause 6.8 if the GES tool is used, as indicated by data stream element *bsTempShapeConfig*.

6.4.2.3 Vector definitions for the 5-1-5_2 configuration

6.4.2.3.1 Operation without temporal shaping tools

For the 5-1-5_2 configuration, the input signal to the decorrelators is defined by $\mathbf{v}^{n,k}$, which is derived from the input vector $\mathbf{x}^{n,k}$ and the matrix $\mathbf{M}_1^{n,k}$ having 5 rows and 1 column, according to:

$$\mathbf{v}^{n,k} = \mathbf{M}_1^{n,k}\mathbf{x}^{n,k} = \mathbf{M}_1^{n,k}\begin{bmatrix} x_{\mathrm{M}_0}^{n,k} \\ x_{\mathrm{res}_1}^{n,k} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ v_0^{n,k} \\ v_1^{n,k} \\ v_3^{n,k} \\ v_4^{n,k} \end{bmatrix}$$

The subscripts for the different elements in the $\mathbf{v}^{n,k}$ vector indicate which OTT box decorrelator the signal is input to, with the exception of $v_m^{n,k}$, which is the direct signal.

The vector $\mathbf{w}^{n,k}$ holding the direct signal, decorrelated signals, and the residual signals is defined according to:

$$\mathbf{w}^{n,k} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ \delta_0(k) D_0\left(v_0^{n,k}\right) + \left(1-\delta_0(k)\right) v_{\mathrm{res}0}^{n,k} \\ \delta_1(k) D_1\left(v_1^{n,k}\right) + \left(1-\delta_1(k)\right) v_{\mathrm{res}1}^{n,k} \\ \delta_3(k) D_3\left(v_3^{n,k}\right) + \left(1-\delta_3(k)\right) v_{\mathrm{res}3}^{n,k} \\ \delta_4(k) D_4\left(v_4^{n,k}\right) + \left(1-\delta_4(k)\right) v_{\mathrm{res}4}^{n,k} \end{bmatrix}$$

where $\delta_X(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{\mathrm{set}}\} \\ 1 & ,otherwise \end{cases}$ and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{\mathrm{resProc}}(X)$ given by Table A.31, and where $D_X\left(v_X^{n,k}\right)$ is the output from decorrelator D_X given the input signal $v_X^{n,k}$.

The subscripts for the different elements indicate which OTT box the signal corresponds to according to the numbering of OTT boxes for the 5-1-5_2 configuration as given by Figure 23. Hence, $D_X\left(v_X^{n,k}\right)$ is the decorrelator output from box OTT_X, and $v_{\mathrm{res}_X}^{n,k}$ is the corresponding residual signal.

The subband output signals are subsequently defined for every time-slot n, and every hybrid subband k, by $\mathbf{y}^{n,k}$, which is derived from the vector $\mathbf{w}^{n,k}$ and the matrix $\mathbf{M}_2^{n,k}$ having 6 rows and 5 columns, according to

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k}\begin{bmatrix} w_{\mathrm{M}_0}^{n,k} \\ w_0^{n,k} \\ w_1^{n,k} \\ w_3^{n,k} \\ w_4^{n,k} \end{bmatrix} = \begin{bmatrix} y_{\mathrm{L}}^{n,k} \\ y_{\mathrm{Ls}}^{n,k} \\ y_{\mathrm{R}}^{n,k} \\ y_{\mathrm{Rs}}^{n,k} \\ y_{\mathrm{C}}^{n,k} \\ y_{\mathrm{LFE}}^{n,k} \end{bmatrix}.$$

The elements of $\mathbf{M}_2^{n,k}$ are defined in subclause 6.5.3, and the hybrid subband signals defined in $\mathbf{y}^{n,k}$ are synthesized to the time-domain by the hybrid synthesis filterbank as defined in subclause 6.3.

6.4.2.3.2 Operation with temporal shaping tools

If temporal shaping is used, the vector $\mathbf{v}^{n,k}$ is defined identically to the previous subclause, however two $\mathbf{w}^{n,k}$ vectors are defined. The first, $\mathbf{w}_{\mathrm{direct}}^{n,k}$ holds the direct signal and the residual signals, while the second $\mathbf{w}_{\mathrm{diffuse}}^{n,k}$ holds the decorrelator output signals, according to:

$$\mathbf{w}_{\mathrm{direct}}^{n,k} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ \left(1-\delta_0(k)\right)v_{\mathrm{res0}}^{n,k} \\ \left(1-\delta_1(k)\right)v_{\mathrm{res1}}^{n,k} \\ \left(1-\delta_3(k)\right)v_{\mathrm{res3}}^{n,k} \\ \left(1-\delta_4(k)\right)v_{\mathrm{res4}}^{n,k} \end{bmatrix}$$

$$\mathbf{w}_{\mathrm{diffuse}}^{n,k} = \begin{bmatrix} v_{\mathrm{M}_0}^{n,k} \\ \delta_0(k)D_0\left(v_0^{n,k}\right) \\ \delta_1(k)D_1\left(v_1^{n,k}\right) \\ \delta_3(k)D_3\left(v_3^{n,k}\right) \\ \delta_4(k)D_4\left(v_4^{n,k}\right) \end{bmatrix}$$

where $\delta_X(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{set}\} \\ 1 & ,otherwise \end{cases}$ and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{\mathrm{resProc}}(X)$ given by Table A.31, and where $D_X\left(v_X^{n,k}\right)$ is the output from decorrelator D_X given the input signal $v_X^{n,k}$. The subscripts are used as outlined in the previous subclause.

Two temporary output vectors are derived, $\mathbf{y}_{\mathrm{direct}}^{n,k}$ holding the direct signal, and $\mathbf{y}_{\mathrm{diffuse}}^{n,k}$ holding the diffuse signal. They are calculated from $\mathbf{w}_{\mathrm{direct}}^{n,k}$ and $\mathbf{w}_{\mathrm{diffuse}}^{n,k}$, using $\mathbf{M}_2^{n,k}$ which is identical to that used if no temporal shaping is applied. The output is derived from these as outlined in subclause 6.7, if the STP tool is used, and subclause 6.8 if the GES tool is used, as indicated by data stream element bsTempShapeConfig.

6.4.3 The 5-2-5 configuration

6.4.3.1 Introduction

In the following subclauses the general structure for the 5-2-5 system is outlined. For this configuration, two downmix signals are present denoted x_{L_0} and x_{R_0}, and a number of residual signals are also present (depending on the data stream).

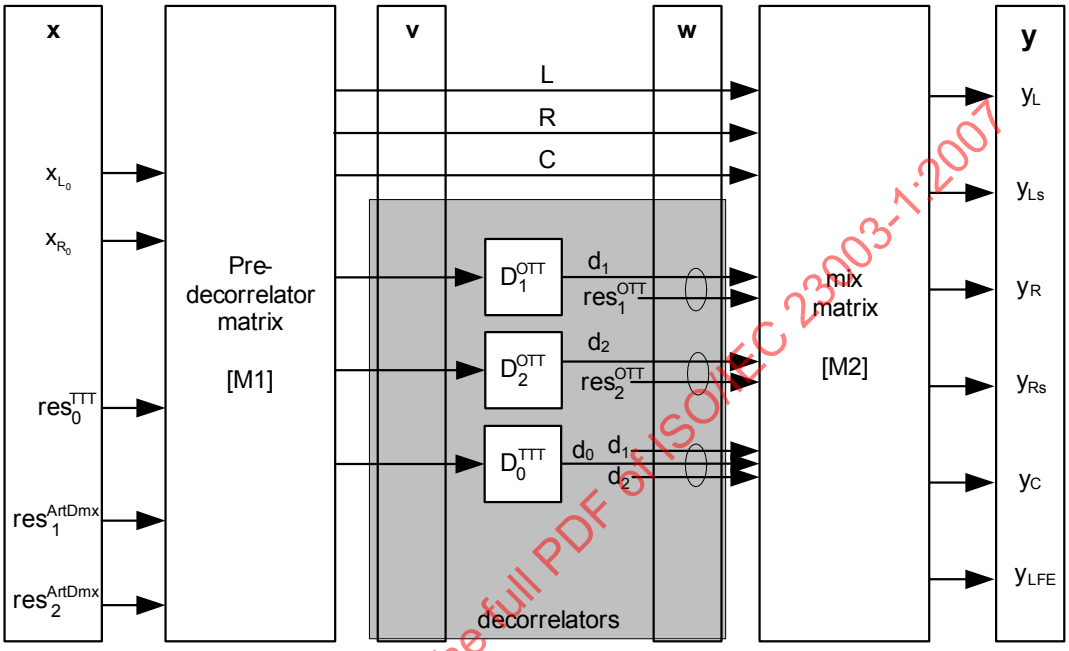

Figure 25 — Matrix view of the spatial audio processing for the 5-2-5 configuration

An alternative visualization of the 5-2-5 system is given in Figure 26, where the system is outlined as a tree-structured system.

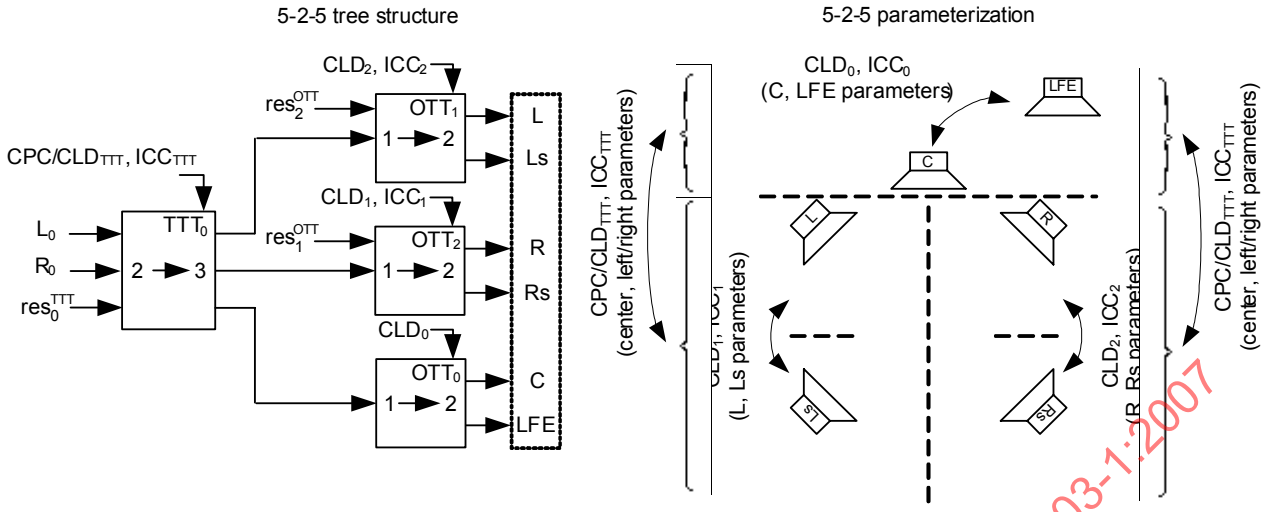

Figure 26 — Tree structure and parameterization view of the spatial audio processing for the 5-2-5 configuration

For the 5-2-5 configuration, the input vector to be multiplied with $\mathbf{M}_1^{n,k}$ consists of the two downmix signals, and an optional residual signal. For this configuration, two decorrelators are used in the same manner with respect to residual coding as for the 5-1-5 configuration. However, if decorrelation in the TTT box is employed, an additional decorrelator is used, the output of which is combined with the output of the other decorrelators.

6.4.3.2 Vector definitions for the 5-2-5 configuration

6.4.3.2.1 Operation without temporal shaping tools

As for the previous configurations, the input signals to the decorrelators are defined by $\mathbf{v}^{n,k}$, which is derived from the input vector $\mathbf{x}^{n,k}$ and the matrix $\mathbf{M}_1^{n,k}$ having for the 5-2-5 configuration 6 rows and 3 columns, according to:

$$\mathbf{v}^{n,k} = \mathbf{M}_1^{n,k}\mathbf{x}^{n,k} = \mathbf{M}_1^{n,k}\begin{bmatrix} x_{\mathrm{L}_0}^{n,k} \\ x_{\mathrm{R}_0}^{n,k} \\ x_{\mathrm{res}_0^{\mathrm{TTT}}}^{n,k} \\ x_{\mathrm{res}_1^{\mathrm{ArtDmx}}}^{n,k} \\ x_{\mathrm{res}_2^{\mathrm{ArtDmx}}}^{n,k} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ v_{\mathrm{OTT}_2}^{n,k} \\ v_{\mathrm{OTT}_1}^{n,k} \\ v_{\mathrm{TTT}_0}^{n,k} \end{bmatrix}$$

The subscripts for the different elements in the $\mathbf{v}^{n,k}$ vector indicate which OTT or TTT box decorrelator the signal is input to, with the exception of $v_{\mathrm{L}}^{n,k}$, $v_{\mathrm{R}}^{n,k}$, and $v_{\mathrm{C}}^{n,k}$, which are the direct signal.

The vector $\mathbf{w}^{n,k}$ holding the direct signal, decorrelated signals, and the residual signals for the OTT boxes is defined according to:

$$\mathbf{w}^{n,k} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ \delta_{\mathrm{OTT}_1}(k) D_1^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_1}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_1}(k)\right) v_{\mathrm{res}_1^{\mathrm{OTT}}}^{n,k} \\ \delta_{\mathrm{OTT}_2}(k) D_2^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_2}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_2}(k)\right) v_{\mathrm{res}_2^{\mathrm{OTT}}}^{n,k} \\ \delta_{\mathrm{TTT}_0}(k)\left(\frac{1}{\sqrt{2}} D_0^{\mathrm{TTT}}\left(v_{\mathrm{TTT}_0}^{n,k}\right) + D_2^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_2}^{n,k}\right) + D_1^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_1}^{n,k}\right)\right) \end{bmatrix} = \begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{OTT}_1}^{n,k} \\ w_{\mathrm{OTT}_2}^{n,k} \\ w_{\mathrm{TTT}_0}^{n,k} \end{bmatrix}$$

where

$$\delta_{\mathrm{OTT}_1}(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{\mathrm{set}}\} \\ 1 & ,otherwise \end{cases}$$

and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{\mathrm{resProc}}(1)$

$$\delta_{OTT_2}(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{set}\} \\ 1 & ,otherwise \end{cases}$$

and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{\mathrm{resProc}}(2)$

$$\delta_{\mathrm{TTT}_0}(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{\mathrm{set}}\} \\ 1 & ,otherwise \end{cases}$$

and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{\mathrm{resProc}}(3)$

The $\kappa(k)$ function is given by Table A.31.

The subscripts for the different elements indicate which OTT or TTT box the signal corresponds to according to the numbering of OTT and TTT boxes for the 5-2-5 configuration as given by Figure 26. Hence, $D_X^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_X}^{n,k}\right)$ is the decorrelator output from box OTT_X, and $v_{\mathrm{res}_X^{\mathrm{OTT}}}^{n,k}$ is the corresponding residual signal.

The subband output signals are subsequently defined for every time-slot n, and every hybrid subband k by $\mathbf{y}^{n,k}$, which is derived from the vector $\mathbf{w}^{n,k}$ and the matrix $\mathbf{M}_2^{n,k}$ having 6 rows and 6 columns, according to

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k}\begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{OTT}_1}^{n,k} \\ w_{\mathrm{OTT}_2}^{n,k} \\ w_{\mathrm{TTT}_0}^{n,k} \end{bmatrix} = \begin{bmatrix} y_{\mathrm{L}}^{n,k} \\ y_{\mathrm{Ls}}^{n,k} \\ y_{\mathrm{R}}^{n,k} \\ y_{\mathrm{Rs}}^{n,k} \\ y_{\mathrm{C}}^{n,k} \\ y_{\mathrm{LFE}}^{n,k} \end{bmatrix}.$$

The elements of $\mathbf{M}_2^{n,k}$ are defined in subclause 6.5.3, and the hybrid subband signals defined in $\mathbf{y}^{n,k}$ are synthesized to the time-domain by the hybrid synthesis filterbank as defined in subclause 6.3.

6.4.3.2.2 Operation with temporal shaping tools

Similarly to the 5-1-5 configurations, if temporal shaping is used, the vector $\mathbf{v}^{n,k}$ is defined identically to the previous subclause, however two $\mathbf{w}^{n,k}$ vectors are defined. The first, $\mathbf{w}_{\mathrm{direct}}^{n,k}$ holds the direct signal and the residual signals, while the second $\mathbf{w}_{\mathrm{diffuse}}^{n,k}$ holds the decorrelator output signals, according to:

$$\mathbf{w}_{\mathrm{direct}}^{n,k} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ \left(1-\delta_{\mathrm{OTT}_1}(k)\right) v_{\mathrm{res}_1}^{n,k\,\mathrm{OTT}} \\ \left(1-\delta_{\mathrm{OTT}_2}(k)\right) v_{\mathrm{res}_2}^{n,k\,\mathrm{OTT}} \\ 0 \end{bmatrix} = \begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{OTT}_1}^{n,k} \\ w_{\mathrm{OTT}_2}^{n,k} \\ w_{\mathrm{TTT}_0}^{n,k} \end{bmatrix}$$

$$\mathbf{w}_{\mathrm{diffuse}}^{n,k} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ \delta_{\mathrm{OTT}_1}(k) D_1^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_1}^{n,k}\right) \\ \delta_{\mathrm{OTT}_2}(k) D_2^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_2}^{n,k}\right) \\ \delta_{\mathrm{TTT}_1}(k)\left(\frac{1}{\sqrt{2}} D_0^{\mathrm{TTT}}\left(v_{\mathrm{TTT}_0}^{n,k}\right) + D_2^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_2}^{n,k}\right) + D_1^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_1}^{n,k}\right)\right) \end{bmatrix} = \begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{OTT}_1}^{n,k} \\ w_{\mathrm{OTT}_2}^{n,k} \\ w_{\mathrm{TTT}_0}^{n,k} \end{bmatrix}$$

where $\delta_{\mathrm{OTT}_1}(k)$, $\delta_{\mathrm{OTT}_2}(k)$ and $\delta_{\mathrm{TTT}_0}(k)$ are defined identically to the previous subclause.

Two temporary output vectors are derived, $\mathbf{y}_{\text{direct}}^{n,k}$ holding the direct signal, and $\mathbf{y}_{\text{diffuse}}^{n,k}$ holding the diffuse signal. They are calculated from $\mathbf{w}_{\text{direct}}^{n,k}$ and $\mathbf{w}_{\text{diffuse}}^{n,k}$, using $\mathbf{M}_2^{n,k}$ which is identical to that used if no temporal shaping is applied. The output is derived from these as outlined in subclause 6.7, if the STP tool is used, and subclause 6.8 if the GES tool is used, as indicated by data stream element bsTempShapeConfig.

6.4.4 The 7-2-7 configurations

6.4.4.1 Introduction

In the following subclauses the general structure for the 7-2-7 configurations is outlined. For these configurations, two downmix signals are present denoted x_{L_0} and x_{R_0}, and a number of residual signals are also present (depending on the data stream).

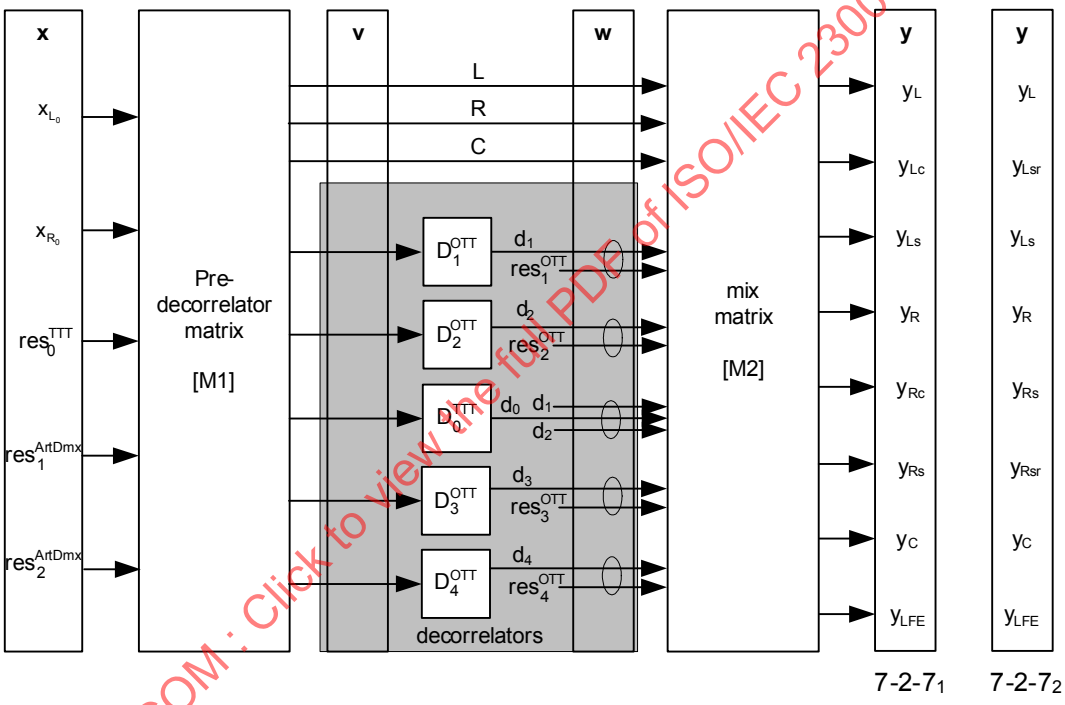

Figure 27 — Matrix view of the spatial audio processing for the 7-2-7 configurations

As for the other configurations the 7-2-7 configuration can also be visualized by means of a tree-structure. This is outlined in Figure 26 and Figure 29. As before, in Figure 26, every OTT box re-creates two channels based on one input channel, the corresponding CLD and ICC parameters, and residual signal. The OTT boxes and the corresponding data are numbered corresponding to the order they appear in the bitstream.

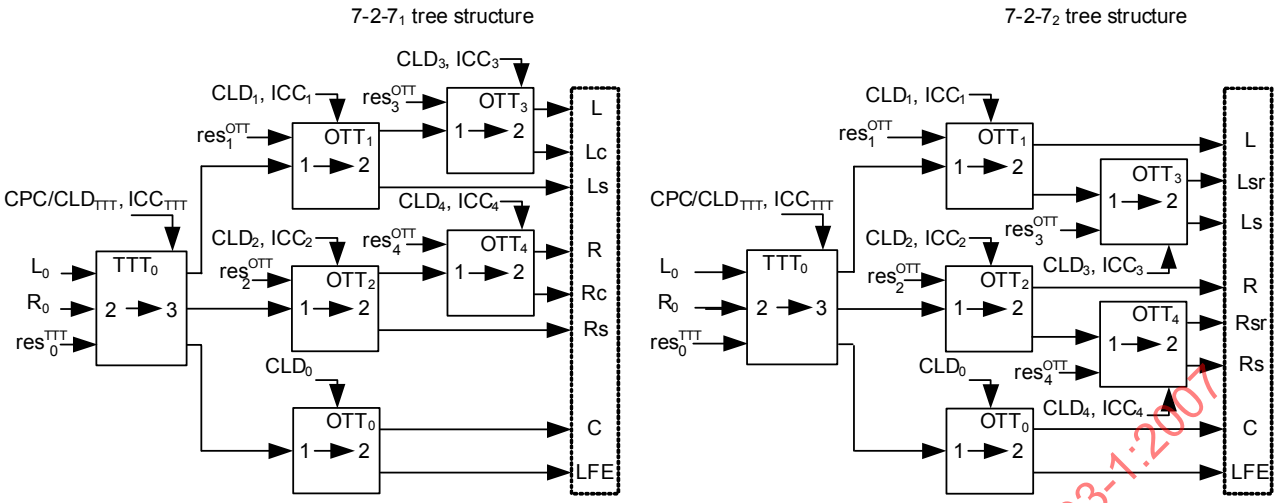

Figure 28 — Tree structure view of the spatial audio processing for the 7-2-7 configurations

The parameterization used for the different 7-2-7 configurations are outlined in Figure 29. Again, the CLD and ICC parameters are labelled corresponding to the order they appear in the bitstream.

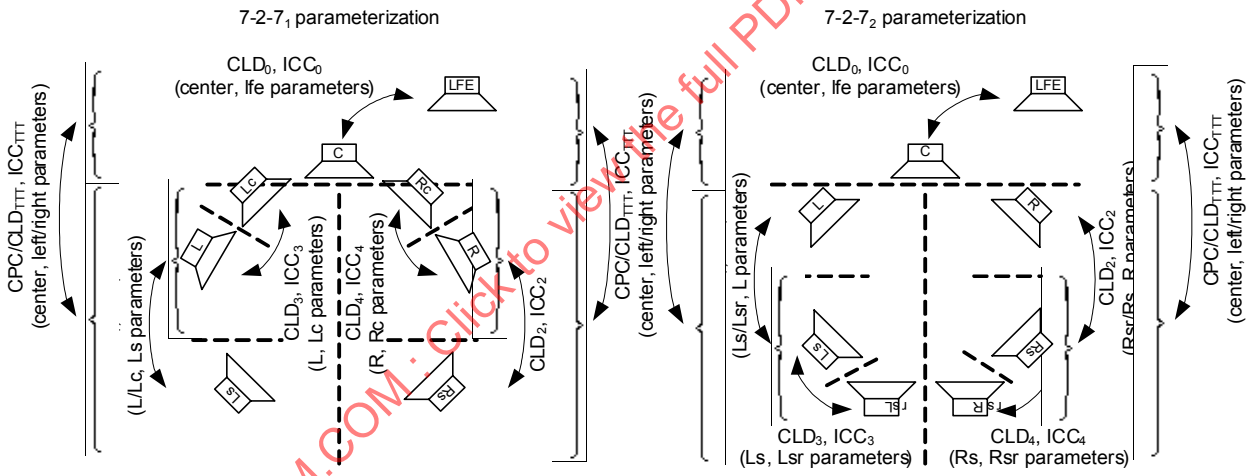

Figure 29 — Parameterization view of the spatial audio processing for the 7-2-7 configurations

For the 7-2-7 configurations, the input vector to be multiplied with $\mathbf{M}_1^{n,k}$ consists of the two downmix signals, and an optional residual signal. For this configuration, four decorrelators are used in the same manner with respect to residual coding as for the 5-2-5 configuration. Similar to the 5-2-5 configurations, if decorrelation in the TTT box is employed, an additional decorrelator is used, the output of which is combined with the output of two of the other decorrelators.

6.4.4.2 Vector definitions for the 7-2-7 configurations

6.4.4.2.1 Operation without temporal shaping tools

As for the previous configurations, the input signals to the decorrelators are defined by $\mathbf{v}^{n,k}$, which is derived from the input vector $\mathbf{x}^{n,k}$ and the matrix $\mathbf{M}_1^{n,k}$ having for the 7-2-7 configuration 8 rows and 3 columns, according to:

$$\mathbf{v}^{n,k} = \mathbf{M}_1^{n,k}\mathbf{x}^{n,k} = \mathbf{M}_1^{n,k}\begin{bmatrix} x_{\mathrm{L}_0}^{n,k} \\ x_{\mathrm{R}_0}^{n,k} \\ x_{\mathrm{res}_0^{\mathrm{TTT}}}^{n,k} \\ x_{\mathrm{res}_1^{\mathrm{ArtDmx}}}^{n,k} \\ x_{\mathrm{res}_2^{\mathrm{ArtDmx}}}^{n,k} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ v_{\mathrm{OTT}_1}^{n,k} \\ v_{\mathrm{OTT}_2}^{n,k} \\ v_{\mathrm{TTT}_0}^{n,k} \\ v_{\mathrm{OTT}_3}^{n,k} \\ v_{\mathrm{OTT}_4}^{n,k} \end{bmatrix}$$

The subscripts for the different elements in the $\mathbf{v}^{n,k}$ vector indicate which OTT or TTT box decorrelator the signal is input to, with the exception of $v_{\mathrm{L}}^{n,k}$, $v_{\mathrm{R}}^{n,k}$, and $v_{\mathrm{C}}^{n,k}$, which are the direct signal.

The vector $\mathbf{w}^{n,k}$ holding the direct signal, decorrelated signals, and the residual signals for the OTT boxes is defined according to:

$$\mathbf{w}^{n,k} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ \delta_{\mathrm{OTT}_1}(k) D_1^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_1}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_1}(k)\right) v_{\mathrm{res}_1^{\mathrm{OTT}}}^{n,k} \\ \delta_{\mathrm{OTT}_2}(k) D_2^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_2}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_2}(k)\right) v_{\mathrm{res}_2^{\mathrm{OTT}}}^{n,k} \\ \delta_{\mathrm{TTT}_0}(k)\left(\frac{1}{\sqrt{2}} D_0^{\mathrm{TTT}}\left(v_{\mathrm{TTT}_0}^{n,k}\right) + D_2^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_2}^{n,k}\right) + D_1^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_1}^{n,k}\right)\right) \\ \delta_{\mathrm{OTT}_3}(k) D_3^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_3}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_3}(k)\right) v_{\mathrm{res}_3^{\mathrm{OTT}}}^{n,k} \\ \delta_{\mathrm{OTT}_4}(k) D_4^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_4}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_4}(k)\right) v_{\mathrm{res}_4^{\mathrm{OTT}}}^{n,k} \end{bmatrix} = \begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{OTT}_1}^{n,k} \\ w_{\mathrm{OTT}_2}^{n,k} \\ w_{\mathrm{TTT}_0}^{n,k} \\ w_{\mathrm{OTT}_3}^{n,k} \\ w_{\mathrm{OTT}_4}^{n,k} \end{bmatrix}$$

where

$$\delta_{\mathrm{OTT_X}}(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{\mathrm{set}}\} \\ 1 & ,otherwise \end{cases}$$

and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{resProc}(X)$ for $1 \le X \le 4$.

$$\delta_{TTT_0}(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{set}\} \\ 1 & ,otherwise \end{cases}$$

and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{resProc}(5)$

The $\kappa(k)$ function is given by Table A.31.

The subscripts for the different elements indicate which OTT or TTT box the signal corresponds to according to the numbering of OTT and TTT boxes for the 7-2-7 configurations as given by Figure 26. Hence, $D_X^{OTT}\left(v_{OTT_X}^{n,k}\right)$ is the decorrelator output from box OTT_X, and $v_{res_X^{OTT}}^{n,k}$ is the corresponding residual signal.

The subband output signals are subsequently defined for every time-slot n, and every hybrid subband k by $\mathbf{y}^{n,k}$, which is derived from the vector $\mathbf{w}^{n,k}$ and the matrix $\mathbf{M}_2^{n,k}$ having 8 rows and 8 columns, according to

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k} \begin{bmatrix} w_L^{n,k} \\ w_R^{n,k} \\ w_C^{n,k} \\ w_{OTT_1}^{n,k} \\ w_{OTT_2}^{n,k} \\ w_{TTT_0}^{n,k} \\ w_{OTT_3}^{n,k} \\ w_{OTT_4}^{n,k} \end{bmatrix} = \begin{bmatrix} y_L^{n,k} \\ y_{Lc}^{n,k} \\ y_{Ls}^{n,k} \\ y_R^{n,k} \\ y_{Rc}^{n,k} \\ y_{Rs}^{n,k} \\ y_C^{n,k} \\ y_{LFE}^{n,k} \end{bmatrix}$$ for the 7-2-7_1 configuration, and

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k} \begin{bmatrix} w_L^{n,k} \\ w_R^{n,k} \\ w_C^{n,k} \\ w_{OTT_1}^{n,k} \\ w_{OTT_2}^{n,k} \\ w_{TTT_0}^{n,k} \\ w_{OTT_3}^{n,k} \\ w_{OTT_4}^{n,k} \end{bmatrix} = \begin{bmatrix} y_L^{n,k} \\ y_{Lsr}^{n,k} \\ y_{Ls}^{n,k} \\ y_R^{n,k} \\ y_{Rs}^{n,k} \\ y_{Rsr}^{n,k} \\ y_C^{n,k} \\ y_{LFE}^{n,k} \end{bmatrix}$$ for the 7-2-7_2 configuration.

The elements of $\mathbf{M}_2^{n,k}$ are defined in subclause 6.4.4, and the hybrid subband signals defined in $\mathbf{y}^{n,k}$ are synthesized to the time-domain by the hybrid synthesis filterbank as defined in subclause 6.3.

6.4.4.2.2 Operation with temporal shaping tools

Similarly to the other configurations, if temporal shaping is used, the vector $\mathbf{v}^{n,k}$ is defined identically to the previous subclause, however two $\mathbf{w}^{n,k}$ vectors are defined. The first, $\mathbf{w}^{n,k}_{direct}$ holds the direct signal and the residual signals, while the second $\mathbf{w}^{n,k}_{diffuse}$ holds the decorrelator output signals, according to:

$$\mathbf{w}^{n,k}_{\text{direct}} = \begin{bmatrix} v^{n,k}_{\text{L}} \\ v^{n,k}_{\text{R}} \\ v^{n,k}_{\text{C}} \\ \left(1-\delta_{\text{OTT}_1}(k)\right) v^{n,k}_{\text{res}_1^{\text{OTT}}} \\ \left(1-\delta_{\text{OTT}_2}(k)\right) v^{n,k}_{\text{res}_2^{\text{OTT}}} \\ 0 \\ \left(1-\delta_{\text{OTT}_3}(k)\right) v^{n,k}_{\text{res}_3^{\text{OTT}}} \\ \left(1-\delta_{\text{OTT}_4}(k)\right) v^{n,k}_{\text{res}_4^{\text{OTT}}} \end{bmatrix} = \begin{bmatrix} w^{n,k}_{\text{L}} \\ w^{n,k}_{\text{R}} \\ w^{n,k}_{\text{C}} \\ w^{n,k}_{\text{OTT}_1} \\ w^{n,k}_{\text{OTT}_2} \\ w^{n,k}_{\text{TTT}_0} \\ w^{n,k}_{\text{OTT}_3} \\ w^{n,k}_{\text{OTT}_4} \end{bmatrix}$$

$$\mathbf{w}^{n,k}_{\text{diffuse}} = \begin{bmatrix} v^{n,k}_{\text{L}} \\ v^{n,k}_{\text{R}} \\ v^{n,k}_{\text{C}} \\ \delta_{\text{OTT}_1}(k) D_1^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_1}\right) \\ \delta_{\text{OTT}_2}(k) D_2^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_2}\right) \\ \delta_{\text{TTT}_0}(k)\left(\frac{1}{\sqrt{2}} D_0^{\text{TTT}}\left(v^{n,k}_{\text{TTT}_0}\right) + D_2^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_2}\right) + D_1^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_1}\right)\right) \\ \delta_{\text{OTT}_3}(k) D_3^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_3}\right) \\ \delta_{\text{OTT}_4}(k) D_4^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_4}\right) \end{bmatrix} = \begin{bmatrix} w^{n,k}_{\text{L}} \\ w^{n,k}_{\text{R}} \\ w^{n,k}_{\text{C}} \\ w^{n,k}_{\text{OTT}_1} \\ w^{n,k}_{\text{OTT}_2} \\ w^{n,k}_{\text{TTT}_0} \\ w^{n,k}_{\text{OTT}_3} \\ w^{n,k}_{\text{OTT}_4} \end{bmatrix}$$

where $\delta_{\text{OTT}_1}(k)$, $\delta_{\text{OTT}_2}(k)$, $\delta_{\text{OTT}_3}(k)$, $\delta_{\text{OTT}_4}(k)$ and $\delta_{\text{TTT}_0}(k)$ are defined identically to the previous subclause.

Two temporary output vectors are derived, $\mathbf{y}^{n,k}_{\text{direct}}$ holding the direct signal, and $\mathbf{y}^{n,k}_{\text{diffuse}}$ holding the diffuse signal. They are calculated from $\mathbf{w}^{n,k}_{\text{direct}}$ and $\mathbf{w}^{n,k}_{\text{diffuse}}$, using $\mathbf{M}_2^{n,k}$ which is identical to that used if no temporal shaping is applied. The output is derived from these as outlined in subclause 6.7, if the STP tool is used, and subclause 6.8 if the GES tool is used, as indicated by data stream element bsTempShapeConfig.

6.4.5 The 7-5-7 configurations

6.4.5.1 Introduction

In the following subclauses the general structure for the 7-5-7 configurations is outlined. For these configurations, 5.1 downmix signals are present denoted x_L, x_{Ls}, x_R, x_{Rs}, x_C and x_{LFE}, and a number of residual signals are also present (depending on the data stream).

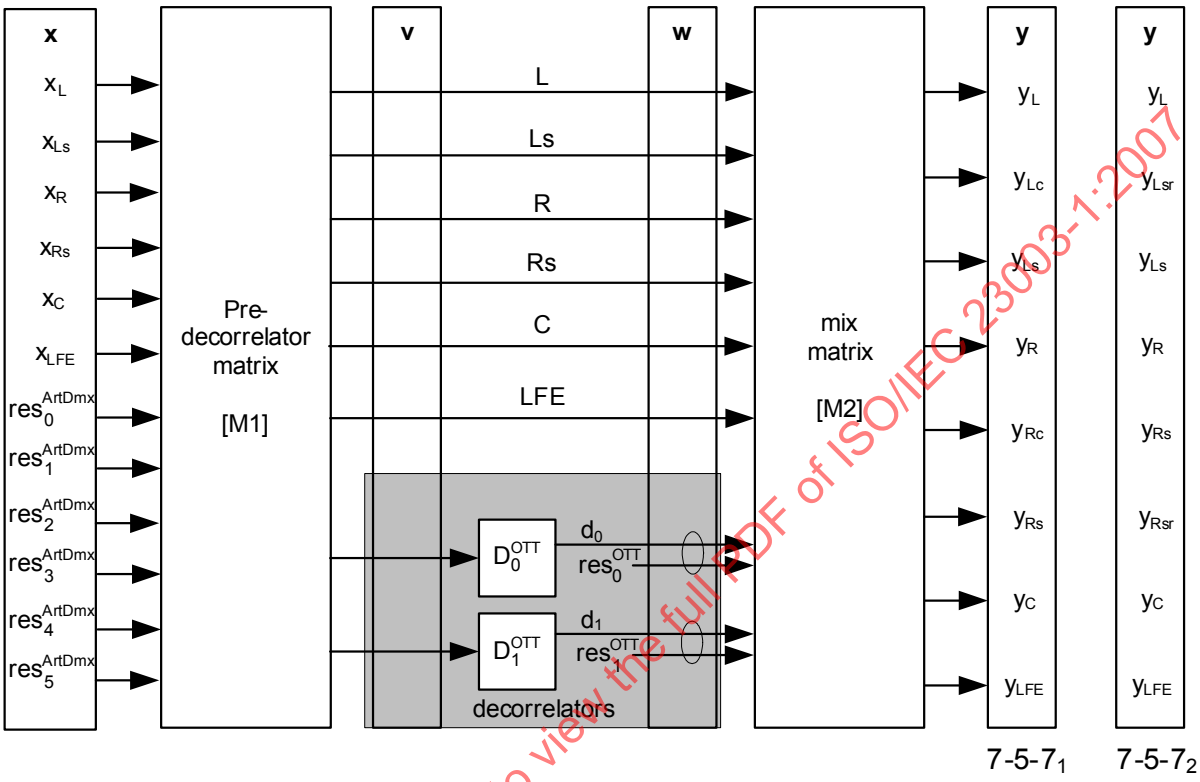

Figure 30 — Matrix view of the spatial audio processing for the 7-5-7 configurations

As for the other configurations the 7-5-7 configuration can also be visualized by means of a tree-structure. This is outlined in Figure 31 and Figure 32. As before, in Figure 31, every OTT box re-creates two channels based on one input channel, the corresponding CLD and ICC parameters, and residual signal. The OTT boxes and the corresponding data are numbered corresponding to the order they appear in the bitstream.

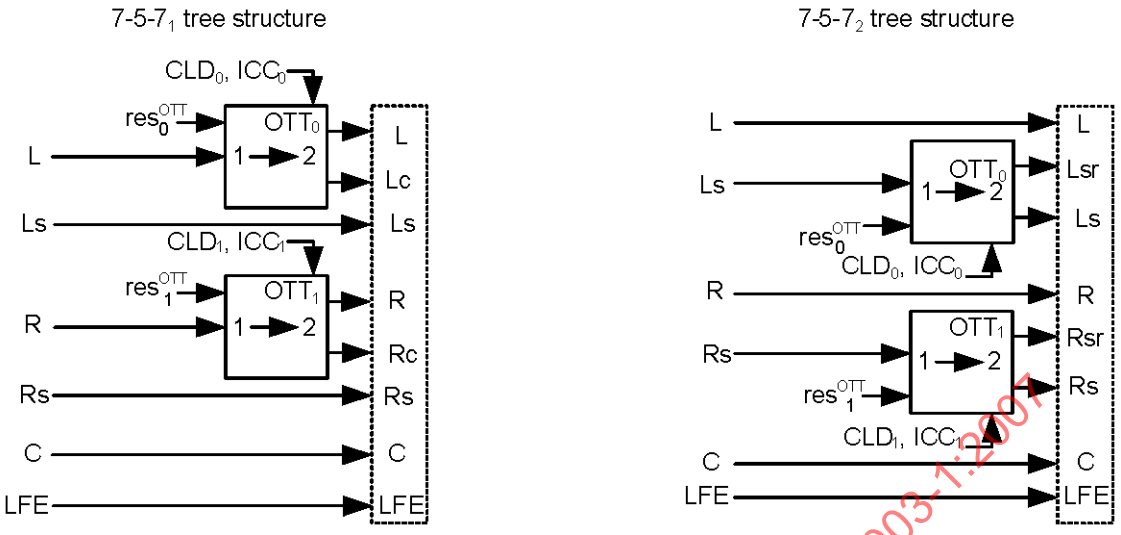

Figure 31 — Tree structure view of the spatial audio processing for the 7-5-7 configurations

The parameterization used for the different 5-1-5 configurations are outlined in Figure 32. Again, the CLD and ICC parameters are labelled corresponding to the order they appear in the bitstream.

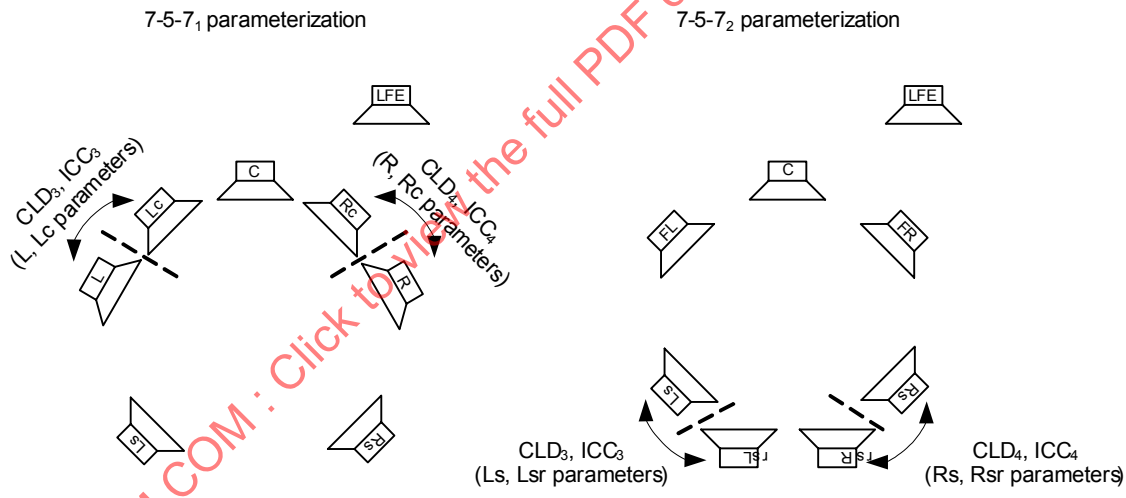

Figure 32 — Parameterization view of the spatial audio processing for the 7-5-7 configurations

For the 7-5-7 configurations, the input vector to be multiplied with $\mathbf{M}_1^{n,k}$ consists of the six downmix signals. For this configuration, four decorrelators are used in the same manner with respect to residual coding as for all the other configurations.

6.4.5.2 Vector definitions for the 7-5-7 configurations

6.4.5.2.1 Operation without temporal shaping tools

As for the previous configurations, the input signals to the decorrelators are defined by $\mathbf{v}^{n,k}$, which is derived from the input vector $\mathbf{x}^{n,k}$ and the matrix $\mathbf{M}_1^{n,k}$ having for the 7-5-7 configuration 8 rows and 6 columns, according to:

$$\mathbf{v}^{n,k} = \mathbf{M}_1^{n,k}\mathbf{x}^{n,k} = \mathbf{M}_1^{n,k}\begin{bmatrix} x_{\mathrm{L}}^{n,k} \\ x_{\mathrm{Ls}}^{n,k} \\ x_{\mathrm{R}}^{n,k} \\ x_{\mathrm{Rs}}^{n,k} \\ x_{\mathrm{C}}^{n,k} \\ x_{\mathrm{LFE}}^{n,k} \\ x_{\mathrm{res}_0}^{n,k\,\mathrm{ArtDmx}} \\ x_{\mathrm{res}_1}^{n,k\,\mathrm{ArtDmx}} \\ x_{\mathrm{res}_2}^{n,k\,\mathrm{ArtDmx}} \\ x_{\mathrm{res}_3}^{n,k\,\mathrm{ArtDmx}} \\ x_{\mathrm{res}_4}^{n,k\,\mathrm{ArtDmx}} \\ x_{\mathrm{res}_5}^{n,k\,\mathrm{ArtDmx}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{Ls}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{Rs}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ v_{\mathrm{LFE}}^{n,k} \\ v_{\mathrm{OTT}_0}^{n,k} \\ v_{\mathrm{OTT}_1}^{n,k} \end{bmatrix}$$

The subscripts for the different elements in the $\mathbf{v}^{n,k}$ vector indicate which OTT or TTT box decorrelator the signal is input to, with the exception of $v_{\mathrm{L}}^{n,k}$, $v_{\mathrm{Ls}}^{n,k}$, $v_{\mathrm{R}}^{n,k}$, $v_{\mathrm{Rs}}^{n,k}$, $v_{\mathrm{C}}^{n,k}$, and $v_{\mathrm{LFE}}^{n,k}$, which are the direct signals.

The vector $\mathbf{w}^{n,k}$ holding the direct signals, decorrelated signals, and the residual signals for the OTT boxes is defined according to:

$$\mathbf{w}^{n,k} = \begin{bmatrix} v_{\mathrm{L}}^{n,k} \\ v_{\mathrm{Ls}}^{n,k} \\ v_{\mathrm{R}}^{n,k} \\ v_{\mathrm{Rs}}^{n,k} \\ v_{\mathrm{C}}^{n,k} \\ v_{\mathrm{LFE}}^{n,k} \\ \delta_{\mathrm{OTT}_0}(k) D_0^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_0}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_0}(k)\right) v_{\mathrm{res}_0}^{n,k\,\mathrm{OTT}} \\ \delta_{\mathrm{OTT}_1}(k) D_1^{\mathrm{OTT}}\left(v_{\mathrm{OTT}_1}^{n,k}\right) + \left(1-\delta_{\mathrm{OTT}_1}(k)\right) v_{\mathrm{res}_1}^{n,k\,\mathrm{OTT}} \end{bmatrix} = \begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{Ls}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{Rs}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{LFE}}^{n,k} \\ w_{\mathrm{OTT}_0}^{n,k} \\ w_{\mathrm{OTT}_1}^{n,k} \end{bmatrix}$$

where

$$\delta_{\mathrm{OTT}_X}(k) = \begin{cases} 0 & ,0 \le k \le \max\{k_{\mathrm{set}}\} \\ 1 & ,otherwise \end{cases}$$

and where k_{set} is the set for which all values of k fulfil $\kappa(k) < \mathbf{m}_{resProc}(X)$ for $0 \le X \le 1$.

The $\kappa(k)$ function is given by Table A.31.

As for the other configurations, the subscripts for the different elements indicate which OTT box the signal corresponds to according to the numbering of OTT boxes for the 7-5-7 configurations as given by Figure 26. Hence, $D_X^{OTT}\left(v_{OTT_X}^{n,k}\right)$ is the decorrelator output from box OTT_X, and $v_{res_X^{OTT}}^{n,k}$ is the corresponding residual signal.

The subband output signals are subsequently defined for every time-slot n, and every hybrid subband k by $\mathbf{y}^{n,k}$, which is derived from the vector $\mathbf{w}^{n,k}$ and the matrix $\mathbf{M}_2^{n,k}$ having 8 rows and 8 columns, according to

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k}\begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{Ls}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{Rs}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{LFE}}^{n,k} \\ w_{\mathrm{OTT_0}}^{n,k} \\ w_{\mathrm{OTT_1}}^{n,k} \end{bmatrix} = \begin{bmatrix} y_{\mathrm{L}}^{n,k} \\ y_{\mathrm{Lc}}^{n,k} \\ y_{\mathrm{Ls}}^{n,k} \\ y_{\mathrm{R}}^{n,k} \\ y_{\mathrm{Rc}}^{n,k} \\ y_{\mathrm{Rs}}^{n,k} \\ y_{\mathrm{C}}^{n,k} \\ y_{\mathrm{LFE}}^{n,k} \end{bmatrix}$$ for the 7-5-7_1 configuration, and

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k}\begin{bmatrix} w_{\mathrm{L}}^{n,k} \\ w_{\mathrm{Ls}}^{n,k} \\ w_{\mathrm{R}}^{n,k} \\ w_{\mathrm{Rs}}^{n,k} \\ w_{\mathrm{C}}^{n,k} \\ w_{\mathrm{LFE}}^{n,k} \\ w_{\mathrm{OTT_0}}^{n,k} \\ w_{\mathrm{OTT_1}}^{n,k} \end{bmatrix} = \begin{bmatrix} y_{\mathrm{L}}^{n,k} \\ y_{\mathrm{Lsr}}^{n,k} \\ y_{\mathrm{Ls}}^{n,k} \\ y_{\mathrm{R}}^{n,k} \\ y_{\mathrm{Rsr}}^{n,k} \\ y_{\mathrm{Rs}}^{n,k} \\ y_{\mathrm{C}}^{n,k} \\ y_{\mathrm{LFE}}^{n,k} \end{bmatrix}$$ for the 7-5-7_2 configuration.

The elements of $\mathbf{M}_2^{n,k}$ are defined in subclause 6.4.5, and the hybrid subband signals defined in $\mathbf{y}^{n,k}$ are synthesized to the time-domain by the hybrid synthesis filterbank as defined in subclause 6.3.

6.4.5.2.2 Operation with temporal shaping tools

Similarly to the other configurations, if temporal shaping is used, the vector $\mathbf{v}^{n,k}$ is defined identically to the previous subclause, however two $\mathbf{w}^{n,k}$ vectors are defined. The first, $w^{n,k}_{\text{direct}}$ holds the direct signal and the residual signals, while the second $w^{n,k}_{\text{diffuse}}$ holds the decorrelator output signals, according to:

$$\mathbf{w}^{n,k}_{\text{direct}} = \begin{bmatrix} v^{n,k}_{\text{L}} \\ v^{n,k}_{\text{Ls}} \\ v^{n,k}_{\text{R}} \\ v^{n,k}_{\text{Rs}} \\ v^{n,k}_{\text{C}} \\ v^{n,k}_{\text{LFE}} \\ \left(1-\delta_{\text{OTT}_0}(k)\right) v^{n,k}_{\text{res}_0^{\text{OTT}}} \\ \left(1-\delta_{\text{OTT}_1}(k)\right) v^{n,k}_{\text{res}_1^{\text{OTT}}} \end{bmatrix} = \begin{bmatrix} w^{n,k}_{\text{L}} \\ w^{n,k}_{\text{Ls}} \\ w^{n,k}_{\text{R}} \\ w^{n,k}_{\text{Rs}} \\ w^{n,k}_{\text{C}} \\ w^{n,k}_{\text{LFE}} \\ w^{n,k}_{\text{OTT}_0} \\ w^{n,k}_{\text{OTT}_1} \end{bmatrix}$$

$$\mathbf{w}^{n,k}_{\text{diffuse}} = \begin{bmatrix} v^{n,k}_{\text{L}} \\ v^{n,k}_{\text{Ls}} \\ v^{n,k}_{\text{R}} \\ v^{n,k}_{\text{Rs}} \\ v^{n,k}_{\text{C}} \\ v^{n,k}_{\text{LFE}} \\ \delta_{\text{OTT}_0}(k) D_0^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_0}\right) \\ \delta_{\text{OTT}_1}(k) D_1^{\text{OTT}}\left(v^{n,k}_{\text{OTT}_1}\right) \end{bmatrix} = \begin{bmatrix} w^{n,k}_{\text{L}} \\ w^{n,k}_{\text{Ls}} \\ w^{n,k}_{\text{R}} \\ w^{n,k}_{\text{Rs}} \\ w^{n,k}_{\text{C}} \\ w^{n,k}_{\text{LFE}} \\ w^{n,k}_{\text{OTT}_0} \\ w^{n,k}_{\text{OTT}_1} \end{bmatrix}$$

where $\delta_{\text{OTT}_0}(k)$, $\delta_{\text{OTT}_1}(k)$ are defined identically to the previous subclause.

Two temporary output vectors are derived, $\mathbf{y}^{n,k}_{\text{direct}}$ holding the direct signal, and $\mathbf{y}^{n,k}_{\text{diffuse}}$ holding the diffuse signal. They are calculated from $\mathbf{w}^{n,k}_{\text{direct}}$ and $\mathbf{w}^{n,k}_{\text{diffuse}}$, using $\mathbf{M}^{n,k}_2$ which is identical to that used if no temporal shaping is applied. The output is derived from these as outlined in subclause 6.7, if the STP tool is used, and subclause 6.8 if the GES tool is used, as indicated by data stream element bsTempShapeConfig.

6.4.6 The Arbitrary Trees configurations

6.4.6.1 Introduction

In the following subclause the general structure for the Arbitrary Trees is outlined. For this configuration any of the predefined trees e.g. 5-2-5 or 7-5-7 can be extended to output even more channels. This is achieved by

applying a post-matrix $\mathbf{M}_3^{n,k}$ to the output from any of the predefined trees. Figure 33 below shows a general overview of how the post-matrix is applied to the output of the predefined trees.

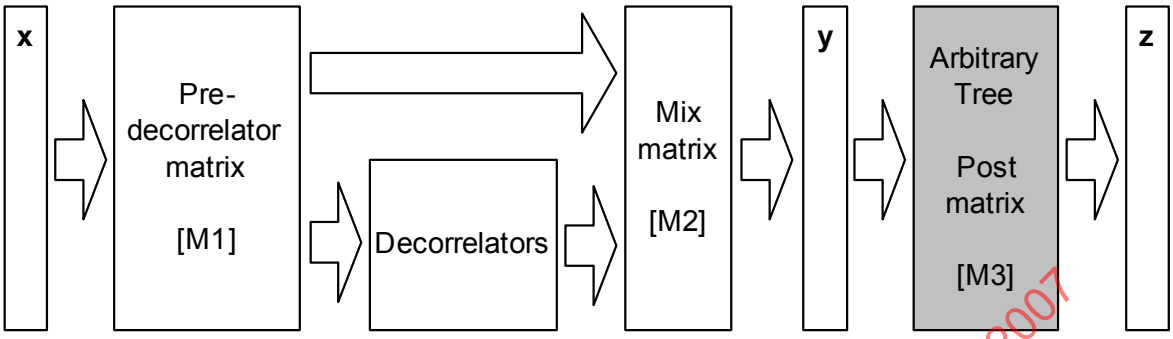

Figure 33 — Matrix view of the spatial audio processing when adding the arbitrary trees as a post process

6.4.6.2 Vector definitions for Arbitrary Trees

Since Arbitrary Trees can be combined with any of the predefined trees the output vector from the predefined tree $\mathbf{y}^{n,k}$ can have several different number of channels and channel order, which in combination with the fact that Arbitrary Trees can produce many output channels result in the fact that the size of the post-matrix $\mathbf{M}_3^{n,k}$ can vary a lot. The subband output signals from the Arbitrary Trees is defined for every time-slot n, and every hybrid subband k by $\mathbf{z}^{n,k}$, which is derived from the vector $\mathbf{y}^{n,k}$ and the post-matrix $\mathbf{M}_3^{n,k}$ according to:

$$\mathbf{z}^{n,k} = \mathbf{M}_3^{n,k}\mathbf{y}^{n,k}$$

The elements of $\mathbf{M}_3^{n,k}$ are defined in subclause 6.5.4, and the hybrid subband signals defined in $\mathbf{z}^{n,k}$ are synthesized to the time-domain by the hybrid synthesis filterbank as defined in subclause 6.3.2.2.

6.4.7 Stereo output from mono downmix signals

6.4.7.1 Introduction

In order to output stereo signal from a mono downmix with the corresponding MPEG Surround parameterization (5-1-5_1 or 5-1-5_2 tree structure) the parameters needs to be recalculated to a single set of CLD and ICC parameters.

6.4.7.2 Parameter recalculation

6.4.7.2.1 5-1-5_1 configuration

For $0 \le m < M_{\text{proc}}, 0 \le l < L$, the four CLD parameters $CLD_X^{l,m}$, $X = 0,1,2,3$, are used to obtain normalized channel powers

$$E_{\mathrm{L}}^{l,m} = \left(c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} c_{1,\mathrm{OTT}_3}^{l,m} \right)^2 ,$$

$$E_{\mathrm{R}}^{l,m} = \left(c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} c_{2,\mathrm{OTT}_3}^{l,m} \right)^2$$

$$E_{\mathrm{C}}^{l,m} = \left(c_{1,\mathrm{OTT}_0}^{l,m} c_{2,\mathrm{OTT}_1}^{l,m} \right)^2$$

$$E_{\mathrm{Ls}}^{l,m} = \left(c_{2,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_2}^{l,m} \right)^2$$

$$E_{\mathrm{Rs}}^{l,m} = \left(c_{2,\mathrm{OTT}_0}^{l,m} c_{2,\mathrm{OTT}_2}^{l,m} \right)^2$$

where

$$c_{1,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{10^{\frac{CLD_X^{l,m}}{10}}}{1+10^{\frac{CLD_X^{l,m}}{10}}}} \text{ and } c_{2,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{1}{1+10^{\frac{CLD_X^{l,m}}{10}}}}$$

where

$$CLD_X^{l,m} = \mathbf{D}_{\mathrm{CLD}}\left(X,l,m\right)$$

The normalized powers of the stereo output channels are then estimated by

$$E_{\mathrm{L}_0}^{l,m} = E_{\mathrm{L}}^{l,m} + E_{\mathrm{Ls}}^{l,m} + \frac{E_{\mathrm{C}}^{l,m}}{2},$$

$$E_{\mathrm{R}_0}^{l,m} = E_{\mathrm{R}}^{l,m} + E_{\mathrm{Rs}}^{l,m} + \frac{E_{\mathrm{C}}^{l,m}}{2}.$$

Two of the ICC parameters $ICC_X^{l,m}$, $X = 2,3$ where

$$ICC_X^{l,m} = \mathbf{D}_{\mathrm{ICC}}\left(X,l,m\right)$$

are incorporated into an estimate of the correlation of the stereo output channels,

$$p_{\mathrm{L}_0\mathrm{R}_0} = \frac{E_{\mathrm{C}}^{l,m}}{2} + ICC_2^{l,m}\left(\left(c_{2,\mathrm{OTT}_0}^{l,m}\right)^2 c_{1,\mathrm{OTT}_2}^{l,m} c_{2,\mathrm{OTT}_2}^{l,m}\right) + ICC_3^{l,m}\left(\left(c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m}\right)^2 c_{1,\mathrm{OTT}_3}^{l,m} c_{2,\mathrm{OTT}_3}^{l,m}\right)$$

The desired unquantized CLD and ICC values are then computed according to

$$CLD_{\mathrm{f}}^{l,m} = 10\log_{10}\left(\frac{E_{\mathrm{L}_0}^{l,m} + \varepsilon}{E_{\mathrm{R}_0}^{l,m} + \varepsilon}\right),$$

$$ICC_{\mathrm{f}}^{l,m} = \max\left\{-.99, \min\left\{1, \frac{p_{\mathrm{L}_0\mathrm{R}_0} + \varepsilon}{\sqrt{E_{\mathrm{L}_0}^{l,m} E_{\mathrm{R}_0}^{l,m}} + \varepsilon}\right\}\right\},$$

which are subsequently quantized into $CLD_{\mathrm{q}}^{l,m}$ and $ICC_{\mathrm{q}}^{l,m}$ by rounding to the nearest CLD and ICC values in Table 82 and Table 83.

A gain adjustment factor is also defined by

$$g^{l,m} = 1.$$

6.4.7.2.2 5-1-5$_2$ configuration

For $0 \le m < M_{Proc}, 0 \le l < L$, the two CLD parameters $CLD_X^{l,m}$ for $X = 0,1$, are first used to obtain normalized channel powers

$$E_{\mathrm{L}}^{l,m} = \left(c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} \right)^2,$$

$$E_{\mathrm{R}}^{l,m} = \left(c_{1,\mathrm{OTT}_0}^{l,m} c_{2,\mathrm{OTT}_1}^{l,m} \right)^2,$$

$$E_{\mathrm{C}}^{l,m} = \left(c_{2,\mathrm{OTT}_0}^{l,m} \right)^2,$$

Two of the ICC parameters $ICC_X^{l,m}$, $X = 0,1$, where

$$ICC_X^{l,m} = \mathbf{D}_{\mathrm{ICC}}(X,l,m)$$

are incorporated into estimates of normalized powers and correlation of the stereo output channels,

$$E_{\mathrm{L}_0}^{l,m} = E_{\mathrm{L}}^{l,m} + \frac{E_{\mathrm{C}}^{l,m}}{2} + \sqrt{2} ICC_0^{l,m} c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} c_{2,\mathrm{OTT}_0}^{l,m},$$

$$E_{\mathrm{R}_0}^{l,m} = E_{\mathrm{R}}^{l,m} + \frac{E_{\mathrm{C}}^{l,m}}{2} + \sqrt{2} ICC_0^{l,m} c_{1,\mathrm{OTT}_0}^{l,m} c_{2,\mathrm{OTT}_1}^{l,m} c_{2,\mathrm{OTT}_0}^{l,m},$$

$$p_{\mathrm{L}_0\mathrm{R}_0} = \frac{E_{\mathrm{C}}^{l,m}}{2} + c_{1,\mathrm{OTT}_0}^{l,m} \left(ICC_1^{l,m} c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} c_{2,\mathrm{OTT}_1}^{l,m} + \frac{1}{\sqrt{2}} ICC_0^{l,m} c_{2,\mathrm{OTT}_0}^{l,m} \sqrt{1 + ICC_1^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} c_{2,\mathrm{OTT}_1}^{l,m}} \right).$$

where

$$c_{1,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{10^{\frac{CLD_X^{l,m}}{10}}}{1 + 10^{\frac{CLD_X^{l,m}}{10}}}} \quad \text{and} \quad c_{2,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{1}{1 + 10^{\frac{CLD_X^{l,m}}{10}}}}$$

where

$$CLD_X^{l,m} = \mathbf{D}_{\mathrm{CLD}}(X,l,m)$$

The desired unquantized CLD and ICC values are then computed according to

$$CLD_{\mathrm{f}}^{l,m} = 10\log_{10}\left(\frac{E_{\mathrm{L}_0}^{l,m}+\varepsilon}{E_{\mathrm{R}_0}^{l,m}+\varepsilon}\right),$$

$$ICC_{\mathrm{f}}^{l,m} = \max\left\{-.99, \min\left\{1, \frac{p_{\mathrm{L}_0\mathrm{R}_0}+\varepsilon}{\sqrt{L_0 R_0}+\varepsilon}\right\}\right\},$$

which are subsequently quantized into $CLD_{\mathrm{q}}^{l,m}$ and $ICC_{\mathrm{q}}^{l,m}$ by rounding to the nearest CLD and ICC values in Table 82 and Table 83.

A gain adjustment factor is also defined by

$$g^{l,m} = \sqrt{E_{\mathrm{L}_0}^{l,m} + E_{\mathrm{R}_0}^{l,m}}$$

6.4.7.3 Stereo output processing

Given the parameters, the output is derived similar as for the normal operation.

One decorrelator is used, and the input signal to the decorrelator is defined by $\mathbf{v}^{n,k}$, which is derived from the input vector $\mathbf{x}^{n,k}$ and the matrix $\mathbf{M}_1^{n,k}$ has 2 rows and 1 column, according to:

$$\mathbf{v}^{n,k} = \mathbf{M}_1^{n,k}\mathbf{x}^{n,k} = \begin{bmatrix}1\\1\end{bmatrix}\left[x_1^{n,k}\right] = \begin{bmatrix}v_{\mathrm{M}_0}^{n,k}\\ v_0^{n,k}\end{bmatrix}$$

The subscripts for the different elements in the $\mathbf{v}^{n,k}$ vector indicate which OTT box decorrelator the signal is input to, with the exception of $v_m^{n,k}$, which is the direct signal.

The vector $\mathbf{w}^{n,k}$ holding the direct signal, decorrelated signals, and the residual signals is defined according to:

$$\mathbf{w}^{n,k} = \begin{bmatrix}v_{\mathrm{M}_0}^{n,k}\\ D_0\left(v_0^{n,k}\right)\end{bmatrix} = \begin{bmatrix}w_{\mathrm{M}_0}^{n,k}\\ w_0^{n,k}\end{bmatrix}$$

where $D_0\left(v_0^{n,k}\right)$ is the output from decorrelator D_0 given the input signal $v_0^{n,k}$.

The subband output signals are subsequently defined for every time-slot n, and every hybrid subband k, by $\mathbf{y}^{n,k}$, which is derived from the vector $\mathbf{w}^{n,k}$ and the matrix $\mathbf{M}_2^{n,k}$ having 2 rows and 2 columns, according to

$$\mathbf{y}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}^{n,k} = \mathbf{M}_2^{n,k}\begin{bmatrix}w_{\mathrm{M}_0}^{n,k}\\ w_0^{n,k}\end{bmatrix} = \begin{bmatrix}y_{\mathrm{L}}^{n,k}\\ y_{\mathrm{R}}^{n,k}\end{bmatrix},$$

where $\mathbf{M}_2^{n,k}$ is defined according to subclause 6.5.3, albeit given $\mathbf{R}_2$ as defined below.

The $\mathbf{R}_2^{l,m}$ matrix is defined, given the recalculated CLD and ICC parameters, according to:

$$\mathbf{R}_2^{l,m} = \begin{bmatrix} H11^{l,m} & H12^{l,m} \\ H21^{l,m} & H22^{l,m} \end{bmatrix} = g^{l,m} \cdot \begin{bmatrix} c_1^{l,m}\cos\left(\alpha^{l,m}+\beta^{l,m}\right) & c_1^{l,m}\sin\left(\alpha^{l,m}+\beta^{l,m}\right) \\ c_2^{l,m}\cos\left(-\alpha^{l,m}+\beta^{l,m}\right) & c_2^{l,m}\sin\left(-\alpha^{l,m}+\beta^{l,m}\right) \end{bmatrix}$$

where

$$c_1^{l,m} = \sqrt{\frac{10^{\frac{CLD_q^{l,m}}{10}}}{1+10^{\frac{CLD_q^{l,m}}{10}}}}, \text{ and } c_2^{l,m} = \sqrt{\frac{1}{1+10^{\frac{CLD_q^{l,m}}{10}}}},$$

and where:

$$\beta^{l,m} = \arctan\left(\tan\left(\alpha^{l,m}\right)\frac{c_2^{l,m}-c_1^{l,m}}{c_2^{l,m}+c_1^{l,m}}\right), \text{ and } \alpha^{l,m} = \frac{1}{2}\arccos\left(ICC_q^{l,m}\right)$$

for $0 \le m < M_{\text{proc}}, 0 \le l < L$, given the $CLD_q^{l,m}$ and $ICC_q^{l,m}$ parameters calculated in the previous subclauses

6.5 Calculation of pre-matrix M1, mix-matrix M2 and post-matrix M3

6.5.1 Introduction

In the following subclauses, the matrices $\mathbf{M}_1^{n,k}$ and $\mathbf{M}_2^{n,k}$ are defined for every time-slot n, and every hybrid subband k. They are interpolated versions of $\mathbf{R}_1^{l,m}\mathbf{G}_1^{l,m}\mathbf{H}^{l,m}$ and $\mathbf{R}_2^{l,m}$ which are defined for a given parameter time-slot l, and a given processing band m, based on the CLD and ICC and CPC parameters valid for that parameter time-slot and processing band. Hence, the matrices are frequency dependent and $\mathbf{R}_1^{l,m}\mathbf{G}_1^{l,m}\mathbf{H}^{l,m}$ and $\mathbf{R}_2^{l,m}$ are calculated for every processing band, and subsequently mapped to every hybrid band.

6.5.2 Definition of pre-matrix M1

6.5.2.1 Introduction

The pre-matrix $\mathbf{M}_1^{n,k}$ defines how the available downmix signals are input to the decorrelators used in the decoder. Therefore, the matrix size depends on the number of downmix input signals available and the number of decorrelators used, while the elements of the matrix are derived from the CLD and/or CPC parameters.

$$\mathbf{M}_1^{n,k} = \begin{cases} \mathbf{W}_1^{l,k}\alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{W}_1^{-1,k} & , 0 \le n \le \mathbf{t}(l), l = 0 \\ \mathbf{W}_1^{l,k}\alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{W}_1^{l-1,k} & , \mathbf{t}(l-1) < n \le \mathbf{t}(l), 1 \le l < L \end{cases}$$

for $0 \le l < L$, $0 \le k < K$ where

$$\alpha(n,l) = \begin{cases} \dfrac{n+1}{\mathbf{t}(l)+1} & ,l = 0 \\ \dfrac{n-\mathbf{t}(l-1)}{\mathbf{t}(l)-\mathbf{t}(l-1)} & ,otherwise \end{cases}$$

and $\mathbf{W}_1^{l,k}$ can be processed by smoothing according to:

$$\mathbf{W}_1^{l,k} = \begin{cases} \mathbf{s}_{\text{delta}}(l) \cdot \mathbf{W}_{\text{konj}}^{l,k} + (1 - \mathbf{s}_{\text{delta}}(l)) \cdot \mathbf{W}_1^{l-1,k} & ,\mathbf{S}_{\text{proc}}(l, \kappa(k)) = 1 \\ \mathbf{W}_{\text{konj}}^{l,k} & ,\mathbf{S}_{\text{proc}}(l, \kappa(k)) = 0 \end{cases}$$

where

$$\mathbf{W}_{\text{temp}}^{l,k} = \mathbf{R}_1^{l,\kappa(k)} \mathbf{G}_1^{l,\kappa(k)} \mathbf{H}^{l,\kappa(k)}$$

$$\mathbf{W}_{\text{konj}}^{l,k} = \kappa_{\text{konj}}(k, \mathbf{W}_{\text{temp}}^{l,k})$$

for $0 \le k < K, 0 \le l < L$ and where $\kappa(k)$ and $\kappa_{\text{konj}}(k,x)$ are given in Table A.31, where the first row is the hybrid subband k, the second row is the corresponding processing band, and the third row is the complex conjugation $x*$ of x for certain hybrid subbands k.

$\mathbf{W}_1^{-1,k}$ corresponds to the last parameter set of the previous frame (zero for the first frame).

The matrices $\mathbf{R}_1^{l,m}$, $\mathbf{G}_1^{l,m}$, and $\mathbf{H}^{l,m}$ are defined in the following subclauses.

6.5.2.2 Calculation of $\mathbf{R}_1$

6.5.2.2.1 Introduction

The $\mathbf{R}_1^{l,m}$ matrix controls the amount of input to the decorrelators, and for the 5-2-5 configuration does the upmix from two downmix channels to a left, centre, right channel. It does not add decorrelated signal, and therefore, it is only a function of the CLD and/or CPC parameters. The $\mathbf{R}_1^{l,m}$ matrix is defined differently depending on the configuration used. For the 5-1-5 configurations the elements of the $\mathbf{R}_1^{l,m}$ matrix is defined by the CLD parameters corresponding to the OTT modules as indicated in the bitstream. For the 5-2-5 configuration the functionality of the TTT box is implemented as well as controlling the input to the decorrelators. The Two-To-Three channel upmix as done by the TTT box, can be done differently for different frequency ranges, this is signaled by means of **bsTttModeLow** and **bsTttModeHigh**.

6.5.2.2.2 5-1-5 configuration

6.5.2.2.2.1 5-1-5$_1$ configuration

For the 5-1-5$_1$ configuration $\mathbf{R}_1^{l,m}$ is defined according to:

$$\mathbf{R}_1^{l,m} = \begin{bmatrix} 1 \\ 1 \\ c_{1,\mathrm{OTT}_0}^{l,m} \\ c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} \\ c_{2,\mathrm{OTT}_0}^{l,m} \end{bmatrix}, \text{ where } c_{1,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{10^{\frac{CLD_X^{l,m}}{10}}}{1+10^{\frac{CLD_X^{l,m}}{10}}}} \text{ and } c_{2,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{1}{1+10^{\frac{CLD_X^{l,m}}{10}}}},$$

and where:

$$CLD_X^{l,m} = \mathbf{D}_{\mathrm{CLD}}\left(X,l,m\right), \quad 0 \le X < 2, 0 \le m < M_{\mathrm{proc}}, 0 \le l < L$$

The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 23, and the subscript $_{1,OTT_X}$ indicates the upper output of box OTT_X, and consequently $_{2,OTT_X}$ indicates the lower output.

6.5.2.2.2.2 5-1-5$_2$ configuration

For the 5-1-5$_2$ configuration $\mathbf{R}_1^{l,m}$ is defined according to:

$$\mathbf{R}_1^{l,m} = \begin{bmatrix} 1 \\ 1 \\ c_{1,\mathrm{OTT}_0}^{l,m} \\ c_{1,\mathrm{OTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} \\ c_{1,\mathrm{OTT}_0}^{l,m} c_{2,\mathrm{OTT}_1}^{l,m} \end{bmatrix}, \text{ where } c_{1,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{10^{\frac{CLD_X^{l,m}}{10}}}{1+10^{\frac{CLD_X^{l,m}}{10}}}} \text{ and } c_{2,\mathrm{OTT}_X}^{l,m} = \sqrt{\frac{1}{1+10^{\frac{CLD_X^{l,m}}{10}}}},$$

and where:

$$CLD_X^{l,m} = \mathbf{D}_{\mathrm{CLD}}\left(X,l,m\right), \quad 0 \le X < 2, 0 \le m < M_{\mathrm{proc}}, 0 \le l < L$$

The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 23, and the subscript $_{1,OTT_X}$ indicates the upper output of box OTT_X, and consequently $_{2,OTT_X}$ indicates the lower output.

6.5.2.2.3 5-2-5 configuration

6.5.2.2.3.1 Introduction

For the 5-2-5 configuration the pre-matrix $\mathbf{R}_1^{l,m}$ is defined differently depending on **bsTttModeLow** and **bsTttModeHigh**. This enables the use of different up-mix strategies for different frequency ranges, which is particularly useful for downmix signals where parts of the frequency range is coded by a non-waveform

preserving coding algorithm e.g. SBR in High Efficiency AAC. The subsequent definitions are given based on **bsTttModeLow** for the frequency range covered by $0 \le m < \mathbf{m}_{\text{tttLowProc}}(0)$. The exact same definitions apply for the parameter bands covered by $\mathbf{m}_{\text{tttLowProc}}(0) \le m < \mathbf{m}_{\text{tttHighProc}}(0)$, albeit dependent on **bsTttModeHigh**.

6.5.2.2.3.2 Prediction based up-mix

The prediction based up-mix comes in two flavours, with decorrelation ($\mathbf{bsTttModeLow}(0)=0$) or without decorrelation ($\mathbf{bsTttModeLow}(0)=1$). The prediction based upmix introduces a prediction error that is equivalent to an energy loss in the three up-mixed signals. This energy loss can be compensated for by:

- adding a residual signal corresponding to the energy loss;
- applying a gain to the up-mixed signals compensating for the energy loss;
- applying a decorrelated signal corresponding to the energy loss.

Neither of the above can be applied simultaneously for the same frequency range. For a data stream where e.g. $\mathbf{bsTttModeLow}(0)=0$ is indicated, i.e. decorrelation is to be used, and a residual signal is present, the decorrelator signal is replaced by the residual signal for the frequency range where the residual signal is present. For $\mathbf{bsTttModeLow}(0)=1$ no decorrelation is used and therefore, in the matrix definitions given below, the last row of the matrix is not applicable and thus set to zero. A residual signal can be added nevertheless.

Furthermore, the decorrelated signal added for the frequency range where $\mathbf{bsTttModeLow}(0)=0$, and where no residual is present, is added to the left front, right front and centre output channels, by mixing in decorrelated signal as outlined in subclause 6.5.3.5. Hence, the energy compensating signal for the left and right side signals (subsequently to become left front/surround pair and right front/surround pair) is only added to the left front and right front part of the left front/surround pair and the right front/surround pair.

For $\mathbf{bsTttModeLow}(0)=0$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m} = \frac{1}{3}\begin{bmatrix} (\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1 \\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1 \\ (1-\alpha^{l,m})\gamma^{l,m}\sqrt{2} & (1-\beta^{l,m})\gamma^{l,m}\sqrt{2} & -\sqrt{2} \\ (\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1 \\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1 \\ (1-\alpha^{l,m})\gamma^{l,m}\sqrt{2} & (1-\beta^{l,m})\gamma^{l,m}\sqrt{2} & 0 \end{bmatrix}, \quad 0 \le m < \mathbf{m}_{\text{tttLowProc}}(0), 0 \le l < L$$

For $\mathbf{bsTttModeLow}(0)=1$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m}=\frac{1}{3}\begin{bmatrix}(\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1\\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1\\ (1-\alpha^{l,m})\gamma^{l,m}\sqrt{2} & (1-\beta^{l,m})\gamma^{l,m}\sqrt{2} & -\sqrt{2}\\ (\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1\\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1\\ 0 & 0 & 0\end{bmatrix},\quad 0\le m<\mathbf{m}_{\text{tttLowProc}}(0),0\le l<L$$

where $\alpha^{l,m}$, $\beta^{l,m}$ and $\gamma^{l,m}$ are given by:

$$\gamma^{l,m}=\begin{cases}1 & m<\mathbf{m}_{resProc}(3),\mathbf{bsResidualPresent}(3)=1,bsResidualCoding=1\\ \dfrac{1}{\mathbf{D}_{ICC}(3,l,m)} & otherwise\end{cases}$$

$$\alpha^{l,m}=\mathbf{D}_{\text{CPC\_1}}(0,l,m),\quad 0\le m<\mathbf{m}_{\text{tttLowProc}}(0),0\le l<L$$

$$\beta^{lm}=\mathbf{D}_{\text{CPC\_2}}(0,l,m),\quad 0\le m<\mathbf{m}_{\text{tttLowProc}}(0),0\le l<L$$

6.5.2.2.3.3 Energy based up-mix

The energy based upmix is designed for non-waveform preserving coding of the downmix signal, such as the SBR frequency range as coded by the High Efficiency AAC codec. The energy based upmix comes in two flavours, $\mathbf{bsTttModeLow}(0)=3$ (with subtraction) and $\mathbf{bsTttModeLow}(0)=5$ (without subtraction). For energy based up-mix neither a decorrelator signal nor a residual signal can be added to the output signals of the TTT module, since there is no prediction error to compensate for. Hence, the third column and the sixth row of the matrices are not applicable for this operation mode. They are set to zero in order to keep the matrix size constant throughout the description.

For $\mathbf{bsTttModeLow}(0)\ge 3$ the pre-gain matrix depends on parameters:

$$q_1^{l,m}=10^{\frac{CLD_1^{l,m}}{10}},q_2^{l,m}=10^{\frac{CLD_2^{l,m}}{10}},$$

where

$$CLD_1^{l,m}=\mathbf{D}_{\text{CLD\_1}}(0,l,m)$$
$$CLD_2^{l,m}=\mathbf{D}_{\text{CLD\_2}}(0,l,m)$$

for $0\le m<\mathbf{m}_{\text{tttLowProc}}(0),0\le l<L$.

For $\mathbf{bsTttModeLow}(0)=3$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m}=\begin{bmatrix} w_{11}^{l,m} & w_{12}^{l,m} & 0 \\ w_{21}^{l,m} & w_{22}^{l,m} & 0 \\ w_{31}^{l,m}\sqrt{2} & w_{32}^{l,m}\sqrt{2} & 0 \\ w_{11}^{l,m} & w_{12}^{l,m} & 0 \\ w_{21}^{l,m} & w_{22}^{l,m} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad 0\le m<\mathbf{m}_{\text{tttLowProc}}(0), 0\le l<L$$

where

$$w_{11}^{l,m}=\sqrt{\frac{q_1^{l,m}q_2^{l,m}+q_2^{l,m}\left(1+q_2^{l,m}\right)}{q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2}} \qquad w_{12}^{l,m}=-\sqrt{\frac{q_2^{l,m}\left(1+q_2^{l,m}\right)^2}{\left(q_1^{l,m}+q_2^{l,m}+1\right)\left(q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2\right)}}$$

$$w_{21}^{l,m}=-\sqrt{\frac{\left(1+q_2^{l,m}\right)^2}{\left(q_1^{l,m}q_2^{l,m}+q_2^{l,m}+1\right)\left(q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2\right)}} \qquad w_{22}^{l,m}=\sqrt{\frac{q_1^{l,m}q_2^{l,m}+q_2^{l,m}+1}{q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2}}$$

$$w_{31}^{l,m}=\tfrac{1}{2}\sqrt{\frac{q_2^{l,m}+1}{q_2^{l,m}+1+q_1^{l,m}q_2^{l,m}}} \qquad w_{32}^{l,m}=\tfrac{1}{2}\sqrt{\frac{q_2^{l,m}+1}{q_1^{l,m}+q_2^{l,m}+1}}$$

For $\mathbf{bsTttModeLow}(0)=5$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m}=\begin{bmatrix} w_{11}^{l,m} & 0 & 0 \\ 0 & w_{22}^{l,m} & 0 \\ w_{31}^{l,m}\sqrt{2} & w_{32}^{l,m}\sqrt{2} & 0 \\ w_{11}^{l,m} & 0 & 0 \\ 0 & w_{22}^{l,m} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad 0\le m<\mathbf{m}_{\text{tttLowProc}}(0), 0\le l<L$$

where

$$w_{11}^{l,m}=\sqrt{\frac{q_1^{l,m}q_2^{l,m}}{q_2^{l,m}+1+q_1^{l,m}q_2^{l,m}}} \qquad w_{22}^{l,m}=\sqrt{\frac{q_1^{l,m}}{q_1^{l,m}+q_2^{l,m}+1}}$$

$$w_{31}^{l,m}=\tfrac{1}{2}\sqrt{\frac{q_2^{l,m}+1}{q_2^{l,m}+1+q_1^{l,m}q_2^{l,m}}} \qquad w_{32}^{l,m}=\tfrac{1}{2}\sqrt{\frac{q_2^{l,m}+1}{q_1^{l,m}+q_2^{l,m}+1}}$$

6.5.2.2.4 7-2-7 configuration

6.5.2.2.4.1 Introduction

For the 7-2-7 configuration the pre-matrix $\mathbf{R}_1^{l,m}$ is defined similar to the pre-matrix for the 5-2-5 configuration, i.e. differently depending on **bsTttModeLow** and **bsTttModeHigh** . The matrix is identical to that of the 5-2-5 configuration, albeit with two additional rows.

6.5.2.2.4.2 Prediction based up-mix

For $\mathbf{bsTttModeLow}(0)=0$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m}=\frac{1}{3}\begin{bmatrix}(\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1\\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1\\ (1-\alpha^{l,m})\gamma^{l,m}\sqrt{2} & (1-\beta^{l,m})\gamma^{l,m}\sqrt{2} & -\sqrt{2}\\ (\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1\\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1\\ (1-\alpha^{l,m})\gamma^{l,m}\sqrt{2} & (1-\beta^{l,m})\gamma^{l,m}\sqrt{2} & 0\\ (\alpha^{l,m}+2)\gamma^{l,m}c_{\delta,\mathrm{OTT}_1}^{l,m} & (\beta^{l,m}-1)\gamma^{l,m}c_{\delta,\mathrm{OTT}_1}^{l,m} & c_{\delta,\mathrm{OTT}_1}^{l,m}\\ (\alpha^{l,m}-1)\gamma^{l,m}c_{\delta,\mathrm{OTT}_2}^{l,m} & (\beta^{l,m}+2)\gamma^{l,m}c_{\delta,\mathrm{OTT}_2}^{l,m} & c_{\delta,\mathrm{OTT}_2}^{l,m}\end{bmatrix},\quad 0\le m<\mathbf{m}_{\mathrm{tttLowProc}}(0),0\le l<L$$

For $\mathbf{bsTttModeLow}(0)=1$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m}=\frac{1}{3}\begin{bmatrix}(\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1\\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1\\ (1-\alpha^{l,m})\gamma^{l,m}\sqrt{2} & (1-\beta^{l,m})\gamma^{l,m}\sqrt{2} & -\sqrt{2}\\ (\alpha^{l,m}+2)\gamma^{l,m} & (\beta^{l,m}-1)\gamma^{l,m} & 1\\ (\alpha^{l,m}-1)\gamma^{l,m} & (\beta^{l,m}+2)\gamma^{l,m} & 1\\ 0 & 0 & 0\\ (\alpha^{l,m}+2)\gamma^{l,m}c_{\delta,\mathrm{OTT}_1}^{l,m} & (\beta^{l,m}-1)\gamma^{l,m}c_{\delta,\mathrm{OTT}_1}^{l,m} & c_{\delta,\mathrm{OTT}_1}^{l,m}\\ (\alpha^{l,m}-1)\gamma^{l,m}c_{\delta,\mathrm{OTT}_2}^{l,m} & (\beta^{l,m}+2)\gamma^{l,m}c_{\delta,\mathrm{OTT}_2}^{l,m} & c_{\delta,\mathrm{OTT}_2}^{l,m}\end{bmatrix},\quad 0\le m<\mathbf{m}_{\mathrm{tttLowProc}}(0),0\le l<L$$

where $\alpha^{l,m}$, $\beta^{l,m}$ and $\gamma^{l,m}$ are given by:

$$\gamma^{l,m} = \begin{cases} 1 & m < \mathbf{m}_{\text{resProc}}(3), \mathbf{bsResidualPresent}(3) = 1, bsResidualCoding = 1 \\ \dfrac{1}{\mathbf{D}_{\text{ICC}}(3,l,m)} & otherwise \end{cases}$$

$$\alpha^{l,m} = \mathbf{D}_{\text{CPC\_1}}(0,l,m), \quad 0 \le m < \mathbf{m}_{\text{tttLowProc}}(0), 0 \le l < L$$

$$\beta^{l,m} = \mathbf{D}_{\text{CPC\_2}}(0,l,m), \quad 0 \le m < \mathbf{m}_{\text{tttLowProc}}(0), 0 \le l < L$$

$\delta = 1$ for the 7-2-7_1 configuration, and $\delta = 2$ for the 7-2-7_2 configuration,

and where

$$c_{1,\text{OTT}_X}^{l,m} = \sqrt{\frac{10^{\frac{CLD_X^{l,m}}{10}}}{1+10^{\frac{CLD_X^{l,m}}{10}}}} \text{ and } c_{2,\text{OTT}_X}^{l,m} = \sqrt{\frac{1}{1+10^{\frac{CLD_X^{l,m}}{10}}}},$$

for

$$CLD_X^{l,m} = \mathbf{D}_{\text{CLD}}(X,l,m), \quad 1 \le X \le 2, 0 \le m < M_{\text{proc}}, 0 \le l < L$$

The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 26, and the subscript $_{1,OTT_X}$ indicates the upper output of box OTT_X, and consequently $_{2,OTT_X}$ indicates the lower output.

6.5.2.2.4.3 Energy based up-mix

For $\mathbf{bsTttModeLow}(0) \ge 3$ the pre-gain matrix depends on parameters:

$$q_1^{l,m} = 10^{\frac{CLD_1^{l,m}}{10}}, q_2^{l,m} = 10^{\frac{CLD_2^{l,m}}{10}},$$

where

$$CLD_1^{l,m} = \mathbf{D}_{\text{CLD\_1}}(0,l,m)$$
$$CLD_2^{l,m} = \mathbf{D}_{\text{CLD\_2}}(0,l,m)$$

for $0 \le m < \mathbf{m}_{\text{tttLowProc}}(0), 0 \le l < L$ and where

$\delta = 1$ for the 7-2-7_1 configuration, and $\delta = 2$ for the 7-2-7_2 configuration,

For $\mathbf{bsTttModeLow}(0)=3$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m}=\begin{bmatrix} w_{11}^{l,m} & w_{12}^{l,m} & 0 \\ w_{21}^{l,m} & w_{22}^{l,m} & 0 \\ w_{31}^{l,m}\sqrt{2} & w_{32}^{l,m}\sqrt{2} & 0 \\ w_{11}^{l,m} & w_{12}^{l,m} & 0 \\ w_{21}^{l,m} & w_{22}^{l,m} & 0 \\ 0 & 0 & 0 \\ w_{11}^{l,m}c_{\delta,\mathrm{OTT}_1}^{l,m} & w_{12}^{l,m}c_{\delta,\mathrm{OTT}_1}^{l,m} & c_{\delta,\mathrm{OTT}_1}^{l,m} \\ w_{21}^{l,m}c_{\delta,\mathrm{OTT}_2}^{l,m} & w_{12}^{l,m}c_{\delta,\mathrm{OTT}_2}^{l,m} & c_{\delta,\mathrm{OTT}_2}^{l,m} \end{bmatrix}, \quad 0\le m<\mathbf{m}_{\mathrm{tttLowProc}}(0), 0\le l<L$$

where

$$w_{11}^{l,m}=\sqrt{\frac{q_1^{l,m}q_2^{l,m}+q_2^{l,m}\left(1+q_2^{l,m}\right)}{q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2}} \qquad w_{12}^{l,m}=\sqrt{\frac{q_2^{l,m}\left(1+q_2^{l,m}\right)^2}{\left(q_1^{l,m}+q_2^{l,m}+1\right)\left(q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2\right)}}$$

$$w_{21}^{l,m}=-\sqrt{\frac{\left(1+q_2^{l,m}\right)^2}{\left(q_1^{l,m}q_2^{l,m}+q_2^{l,m}+1\right)\left(q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2\right)}} \qquad w_{22}^{l,m}=\sqrt{\frac{q_1^{l,m}q_2^{l,m}+q_2^{l,m}+1}{q_1^{l,m}q_2^{l,m}+\left(1+q_2^{l,m}\right)^2}}$$

$$w_{31}^{l,m}=\tfrac{1}{2}\sqrt{\frac{q_2^{l,m}+1}{q_2^{l,m}+1+q_1^{l,m}q_2^{l,m}}} \qquad w_{32}^{l,m}=\tfrac{1}{2}\sqrt{\frac{q_2^{l,m}+1}{q_1^{l,m}+q_2^{l,m}+1}}$$

and where $c_{\delta,\mathrm{OTT}_1}^{l,m}$ and $c_{\delta,\mathrm{OTT}_2}^{l,m}$ are the same as for the prediction based up-mix, and $\delta=1$ for the 7-2-7_1 configuration, and $\delta=2$ for the 7-2-7_2 configuration.

For $\mathbf{bsTttModeLow}(0)=5$ the pre-gain matrix is defined according to:

$$\mathbf{R}_1^{l,m}=\begin{bmatrix} w_{11}^{l,m} & 0 & 0 \\ 0 & w_{22}^{l,m} & 0 \\ w_{31}^{l,m}\sqrt{2} & w_{32}^{l,m}\sqrt{2} & 0 \\ w_{11}^{l,m} & 0 & 0 \\ 0 & w_{22}^{l,m} & 0 \\ 0 & 0 & 0 \\ w_{11}^{l,m}c_{\delta,\mathrm{OTT}_1}^{l,m} & 0 & c_{\delta,\mathrm{OTT}_1}^{l,m} \\ 0 & w_{22}^{l,m}c_{\delta,\mathrm{OTT}_2}^{l,m} & c_{\delta,\mathrm{OTT}_2}^{l,m} \end{bmatrix}, \quad 0\le m<\mathbf{m}_{\mathrm{tttLowProc}}(0), 0\le l<L$$

where

$$w_{11}^{l,m} = \sqrt{\frac{q_1^{l,m} q_2^{l,m}}{q_2^{l,m} + 1 + q_1^{l,m} q_2^{l,m}}} \qquad w_{22}^{l,m} = \sqrt{\frac{q_1^{l,m}}{q_1^{l,m} + q_2^{l,m} + 1}}$$

$$w_{31}^{l,m} = \tfrac{1}{2}\sqrt{\frac{q_2^{l,m} + 1}{q_2^{l,m} + 1 + q_1^{l,m} q_2^{l,m}}} \qquad w_{32}^{l,m} = \tfrac{1}{2}\sqrt{\frac{q_2^{l,m} + 1}{q_1^{l,m} + q_2^{l,m} + 1}}$$

and where $c_{\delta,\mathrm{OTT}_1}^{l,m}$ and $c_{\delta,\mathrm{OTT}_2}^{l,m}$ are the same as for the prediction based up-mix, and $\delta = 1$ for the 7-2-7_1 configuration, and $\delta = 2$ for the 7-2-7_2 configuration.

6.5.2.2.5 7-5-7 configuration

For the 7-5-7 configurations the pre-matrix $\mathbf{R}_1^{l,m}$ is defined according to:

$$\mathbf{R}_1^{l,m} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad 0 \le m < M_{\mathrm{proc}}, 0 \le l < L \text{ for the 7-5-}7_1 \text{ configuration,}$$

and

$$\mathbf{R}_1^{l,m} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad 0 \le m < M_{\mathrm{proc}}, 0 \le l < L \text{ for the 7-5-}7_2 \text{ configuration,}$$

6.5.2.3 External downmix compensation

6.5.2.3.1 Introduction

In order to handle modifications of the down-mix signal prior to MPEG Surround decoding, or externally supplied downmix signals, data stream controlled correction factors can be applied. The correction factor is applied to the downmix by means of the $\mathbf{G}_1^{l,m}$ matrix, and ensures that the level of the downmix, for the specific time frequency tile the parameter represents, is the same as the level of the downmix signal obtained

when the spatial parameters were estimated on the encoder side. This is accomplished by the $\mathbf{G}_1^{l,m}$ matrix. Three cases are distinguished, no external downmix compensation ($bsArbitraryDownmix = 0$), parameterized external downmix compensation ($bsArbitraryDownmix = 1$), and residual coding based external downmix compensation ($bsArbitraryDownmix = 2$).

A decoder that does not support residual based external downmix compensation shall operate as if $bsArbitraryDownmix = 1$.

6.5.2.3.2 No external downmix compensation $bsArbitraryDownmix = 0$

If no external downmix compensation is applied, the $\mathbf{G}_1^{l,m}$ matrix is defined according to the following.

For the 5-1-5 configurations, the matrix is defined according to:

$$G_1^{l,m} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

For the 5-2-5 configuration it is defined according to:

$$G_1^{l,m} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

For the 7-2-7 configuration it is defined according to:

$$G_1^{l,m} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

For the 7-5-7 configuration it is defined according to:

$$G_1^{l,m} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

6.5.2.3.3 Parameterized external downmix compensation $bsArbitraryDownmix = 1$

For the 5-1-5 configurations, the matrix is defined according to:

$$G_1^{l,m} = \begin{bmatrix} g_1^{l,m} & 0 \end{bmatrix}$$

where

$$g_1^{l,m} = \mathbf{G}(0,l,m), \quad 0 \le m < M_{\text{proc}}, 0 \le l < L,$$

For the 5-2-5 configuration it is defined according to:

$$G_1^{l,m} = \begin{bmatrix} g_0^{l,m} & 0 & 0 & 0 & 0 \\ 0 & g_1^{l,m} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

where

$$g_X^{l,m} = \mathbf{G}(X,l,m), \quad 0 \le X < 2, 0 \le m < M_{\text{proc}}, 0 \le l < L.$$

For the 7-2-7 configuration it is defined according to:

$$G_1^{l,m} = \begin{bmatrix} g_0^{l,m} & 0 & 0 & 0 & 0 \\ 0 & g_1^{l,m} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

where

$$g_X^{l,m} = \mathbf{G}(X,l,m), \quad 0 \le X < 2, 0 \le m < M_{\text{proc}}, 0 \le l < L.$$

For the 7-5-7 configuration it is defined according to:

$$G_1^{l,m} = \begin{bmatrix} g_0^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & g_1^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_2^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_3^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_4^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & g_5^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where

$$g_X^{l,m} = \mathbf{G}(X,l,m), \quad 0 \le X < 6, 0 \le m < M_{\text{proc}}, 0 \le l < L.$$

6.5.2.3.4 Residual coding based external downmix compensation $bsArbitraryDownmix = 2$

For the 5-1-5 configurations, the matrix is defined according to:

$$G_1^{l,m} = \begin{cases} \begin{bmatrix} \alpha \cdot g_0^{l,m} & 1 \end{bmatrix} & , \quad \mathrm{m} \le \mathrm{m}_{\mathrm{ArtDmxRes}}(i) \\ \begin{bmatrix} g_0^{l,m} & 0 \end{bmatrix} & , otherwise \end{cases}$$

where

$$g_0^{l,m} = \mathbf{G}(0,l,m), \quad 0 \le m < M_{proc}, 0 \le l < L,$$

and the value of α is updated as follows

$$\alpha_k = \begin{cases} \max(0, \alpha_{k-1} - \delta) & , bsArbitraryDownmixResidualAbs = 0, \\ \min(1, \alpha_{k-1} + \delta) & , otherwise, \end{cases}$$

where α_k denotes the value of α for the k-th data frame and the adaptation speed δ amounts to 0.33.

For the 5-2-5 configuration it is defined according to:

$$G_1^{l,m} = \begin{cases} \begin{bmatrix} \alpha \cdot g_0^{l,m} & 0 & 0 & 1 & 0 \\ 0 & \alpha \cdot g_1^{l,m} & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} & , \mathrm{m} \le \mathrm{m}_{\mathrm{ArtDmxRes}}(i) \\ \begin{bmatrix} g_0^{l,m} & 0 & 0 & 0 & 0 \\ 0 & g_1^{l,m} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} & , otherwise \end{cases}$$

where

$$g_X^{l,m} = \mathbf{G}(X,l,m), \quad 0 \le X < 2, 0 \le m < M_{\mathrm{proc}}, 0 \le l < L$$

and the value of α is updated as in the case of the 5-1-5 coder, hence

$$\alpha_k = \begin{cases} \max(0, \alpha_{k-1} - \delta) & , bsArbitraryDownmixResidualAbs = 0, \\ \min(1, \alpha_{k-1} + \delta) & , otherwise, \end{cases}$$

For the 7-2-7 configuration it is defined according to:

$$G_1^{l,m} = \begin{cases} \begin{bmatrix} \alpha \cdot g_0^{l,m} & 0 & 0 & 1 & 0 \\ 0 & \alpha \cdot g_1^{l,m} & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} & , \mathrm{m} \le \mathrm{m}_{\mathrm{ArtDmxRes}}(i) \\ \begin{bmatrix} g_0^{l,m} & 0 & 0 & 0 & 0 \\ 0 & g_1^{l,m} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} & , otherwise \end{cases}$$

where

$$g_X^{l,m} = \mathbf{G}(X,l,m), \quad 0 \le X < 2, 0 \le m < M_{\text{proc}}, 0 \le l < L$$

and the value of α is updated as for the previous configurations.

For the 7-5-7 configuration it is defined according to:

$$G_1^{l,m} = \begin{cases} \begin{bmatrix} \alpha \cdot g_0^{l,m} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha \cdot g_1^{l,m} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha \cdot g_2^{l,m} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha \cdot g_3^{l,m} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha \cdot g_4^{l,m} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha \cdot g_5^{l,m} & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & ,\mathrm{m} \le \mathrm{m}_{\text{ArtDmxRes}}(i) \\ \begin{bmatrix} g_0^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & g_1^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_2^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_3^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_4^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & g_5^{l,m} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & ,otherwise \end{cases}$$

where

$$g_X^{l,m} = \mathbf{G}(X,l,m), \quad 0 \le X < 6, 0 \le m < M_{\text{proc}}, 0 \le l < L$$

and the value of α is updated as for the previous configurations.

6.5.2.4 Matrix compatibility

In order to handle matrix surround compatible stereo down-mixes, the matrix surround operation on the stereo down-mix is reverted if this is indicated in the SpatialSpecificConfig(), i.e., if *bsMatrixMode* is 1.

The inversion of the matrix surround encoded downmix is visualized in Figure 34, where scaled versions of the two downmix signals are used to derive the matrix surround inverted downmix signals.

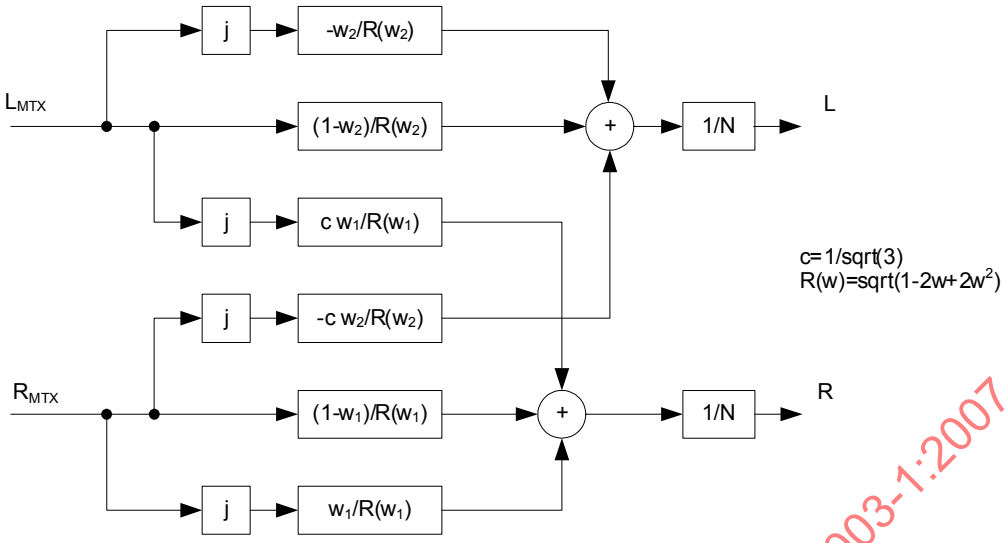

Figure 34 — Matrix compatibility, inversion matrix

The matrix surround inversion matrix only processes the stereo down-mix. Other input signals like TTT residuals and artistic down-mix enhancement residuals are passed unchanged. The size of the matrix surround inversion matrix is therefore defined by the size of the input signal vector $\mathbf{x}^{n,k}$. The inversion matrix for configurations with a stereo down-mix is defined according to:

$$\mathbf{H}^{l,m} = \begin{bmatrix} h_{11}^{l,m} & h_{12}^{l,m} & 0 & 0 & \cdots & 0 \\ h_{21}^{l,m} & h_{22}^{l,m} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

where the entries $h_{xy}^{l,m}$ are complex functions of the CLD and CPC parameters:

$$h_{11}^{l,m} = \frac{g_{22}^{l,m}}{g_{11}^{l,m} g_{22}^{l,m} - g_{12}^{l,m} g_{21}^{l,m}} \qquad h_{12}^{l,m} = \frac{-g_{12}^{l,m}}{g_{11}^{l,m} g_{22}^{l,m} - g_{12}^{l,m} g_{21}^{l,m}}$$

$$h_{21}^{l,m} = \frac{-g_{21}^{l,m}}{g_{11}^{l,m} g_{22}^{l,m} - g_{12}^{l,m} g_{21}^{l,m}} \qquad h_{22}^{l,m} = \frac{g_{11}^{l,m}}{g_{11}^{l,m} g_{22}^{l,m} - g_{12}^{l,m} g_{21}^{l,m}}$$

with

$$g_{11}^{l,m} = \frac{1 - w_1^{l,m} + jw_1^{l,m}}{\left|1 - w_1^{l,m} + jw_1^{l,m}\right|} \qquad g_{12}^{l,m} = \frac{jw_2^{l,m}}{\sqrt{3}\left|1 - w_2^{l,m} + jw_2^{l,m}\right|}$$

$$g_{21}^{l,m} = \frac{-jw_1^{l,m}}{\sqrt{3}\left|1 - w_1^{l,m} + jw_1^{l,m}\right|} \qquad g_{22}^{l,m} = \frac{1 - w_2^{l,m} - jw_2^{l,m}}{\left|1 - w_2^{l,m} + jw_2^{l,m}\right|}$$

For parameter bands where a prediction-based up-mix is used, the variables $w_1^{l,m}$ and $w_2^{l,m}$ are given by:

$$w_1^{l,m} = f_1\left(\mathbf{D}_{\mathrm{CLD}}(1,l,m)\right) f_2\left(\mathbf{D}_{\mathrm{CPC\_1}}(0,l,m)\right) \qquad w_2^{l,m} = f_1\left(\mathbf{D}_{\mathrm{CLD}}(2,l,m)\right) f_2\left(\mathbf{D}_{\mathrm{CPC\_2}}(0,l,m)\right)$$

For parameter bands where an energy-based up-mix is used, the variables $w_1^{l,m}$ and $w_2^{l,m}$ are given by:

$$w_1^{l,m} = f_1\left(\mathbf{D}_{\mathrm{CLD}}(1,l,m)\right) f_3\left(\mathbf{D}_{\mathrm{CLD\_1}}(0,l,m), \mathbf{D}_{\mathrm{CLD\_2}}(0,l,m)\right)$$

$$w_2^{l,m} = f_1\left(\mathbf{D}_{\mathrm{CLD}}(2,l,m)\right) f_4\left(\mathbf{D}_{\mathrm{CLD\_1}}(0,l,m), \mathbf{D}_{\mathrm{CLD\_2}}(0,l,m)\right)$$

where $f_1 \ldots\ldots f_4$ are defined as:

$$f_1(CLD) = \frac{10^{\frac{-CLD}{20}}}{1+10^{\frac{-CLD}{20}}}$$

$$f_2(CPC) = \begin{cases} -1-2CPC & , \quad -1 < CPC < -\frac{1}{2} \\ 1 & , \quad |CPC| \geq 1 \\ \frac{1+2CPC}{3} & , \quad -\frac{1}{2} \leq CPC < 1 \end{cases}$$

$$f_3(CLD_1, CLD_2) = \sqrt{\frac{CLD_1 CLD_2}{CLD_1 CLD_2 + 2(CLD_2 + 1)}}$$

$$f_4(CLD_1, CLD_2) = \sqrt{\frac{CLD_1}{CLD_1 + 2(CLD_2 + 1)}}$$

This definition is valid for all processing bands where prediction based, or energy base up-mix is used, and matrix inversion is enabled (i.e. $\mathbf{bsTttModeLow}(0) < 3$, or $\mathbf{bsTttModeLow}(0) = 4$ if $0 \leq m < \mathbf{m}_{\mathrm{tttLowProc}}(0)$, or $\mathbf{bsTttModeHigh}(0) < 3$, or $\mathbf{bsTttModeHigh}(0) = 4$ if $\mathbf{m}_{\mathrm{tttLowProc}}(0) \leq m < \mathbf{m}_{\mathrm{tttHighProc}}(0)$). For other processing bands, the matrix $\mathbf{H}^{l,m}$ is the unity matrix.

For configurations with a down-mix that contains more (e.g. 757) or less (e.g. 515) down-mix channels the matrix surround inversion is not applicable. The inversion matrix for these configurations therefore defaults to the unity matrix with a size that is equal to the number of columns in input signal vector $\mathbf{x}^{n,k}$ for all parameter sets and processing bands.

6.5.3 Definition of mix-matrix M2

6.5.3.1 Introduction

The post-matrix $\mathbf{M}_2^{n,k}$ defines how the direct signals and the decorrelated signals shall be combined in order to form the re-created multi-channel output signal. It is defined according to:

$$\mathbf{M}_2^{n,k} = \begin{cases} \mathbf{W}_2^{l,k}\alpha(n,l) + (1-\alpha(n,l))\mathbf{W}_2^{-1,k}, & ,0 \le n \le \mathbf{t}(l), l = 0 \\ \mathbf{W}_2^{l,k}\alpha(n,l) + (1-\alpha(n,l))\mathbf{W}_2^{l-1,k}, & ,\mathbf{t}(l-1) < n \le \mathbf{t}(l), 1 \le l < L \end{cases}$$

for $0 \le l < L$, $0 \le k < K$ where

$$\alpha(n,l) = \begin{cases} \dfrac{n+1}{\mathbf{t}(l)+1} & ,l = 0 \\ \dfrac{n-\mathbf{t}(l-1)}{\mathbf{t}(l)-\mathbf{t}(l-1)} & ,otherwise \end{cases}$$

and $\mathbf{W}_2^{l,k}$ can be processed by smoothing according to:

$$\mathbf{W}_2^{l,k} = \begin{cases} \mathbf{s}_{\text{delta}}(l) \cdot \mathbf{R}_2^{l,\kappa(k)} + (1-\mathbf{s}_{\text{delta}}(l)) \cdot \mathbf{W}_2^{l-1,k} & ,\mathbf{S}_{\text{proc}}(l,\kappa(k)) = 1 \\ \mathbf{R}_2^{l,\kappa(k)} & ,\mathbf{S}_{\text{proc}}(l,\kappa(k)) = 0 \end{cases}$$

and where $\kappa(k)$ is given in Table A.31, where the first row is the hybrid band k, and the second row is the corresponding processing band. $\mathbf{W}_2^{-1,k}$ corresponds to the last parameter set of the previous frame (zero for the first frame).

6.5.3.2 Derivation of arbitrary matrix element

For the $\mathbf{R}_2^{n,k}$ matrix the elements are calculated from an equivalent model of an OTT box. The OTT boxes depicted in Figure 23 and Figure 26 can be visualized as shown in Figure 35. The OTT box is essentially a parametric stereo decoder as outlined in ISO/IEC 14496-3. The mono input signal is input to the decorrelator and the mixing unit that creates a stereo output, based on the mono signal, the decorrelated counterpart, and the CLD and ICC parameters, where the CLD controls the localization in the stereo field, and the ICC parameter controls the wideness of the stereo output signal.

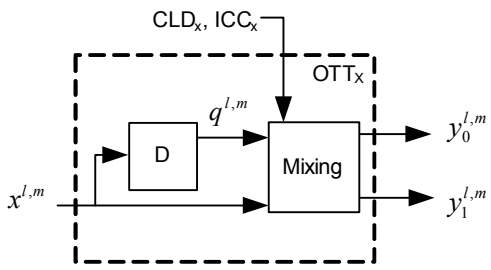

Figure 35 — Equivalent model of a conceptual OTT box

Based on this illustration, the output for an arbitrary OTT box can be defined according to

$$\begin{bmatrix} y_0^{l,m} \\ y_1^{l,m} \end{bmatrix} = \mathbf{H} \begin{bmatrix} x^{l,m} \\ q^{l,m} \end{bmatrix} = \begin{bmatrix} H11_{\mathrm{OTT}_X}^{l,m} & H12_{\mathrm{OTT}_X}^{l,m} \\ H21_{\mathrm{OTT}_X}^{l,m} & H22_{\mathrm{OTT}_X}^{l,m} \end{bmatrix} \begin{bmatrix} x^{l,m} \\ q^{l,m} \end{bmatrix}.$$

Arbitrary matrix elements, $H11_{OTT_X}^{l,m} \cdots H22_{OTT_X}^{l,m}$, for the OTT box labelled OTT_X (where $0 \le X < numOttBoxes$), for the time slot l, and a parameter band m in the post gain matrix are defined according to:

$$\begin{bmatrix} H11_{\mathrm{OTT}_X}^{l,m} & H12_{\mathrm{OTT}_X}^{l,m} \\ H21_{\mathrm{OTT}_X}^{l,m} & H22_{\mathrm{OTT}_X}^{l,m} \end{bmatrix} = \begin{cases} \begin{bmatrix} c_{1,X}^{l,m}\cos\left(\alpha_X^{l,m}+\beta_X^{l,m}\right) & 1 \\ c_{2,X}^{l,m}\cos\left(-\alpha_X^{l,m}+\beta_X^{l,m}\right) & -1 \end{bmatrix} & ,m < resBands_X \\ \begin{bmatrix} c_{1,X}^{l,m}\cos\left(\alpha_X^{l,m}+\beta_X^{l,m}\right) & c_{1,X}^{l,m}\sin\left(\alpha_X^{l,m}+\beta_X^{l,m}\right) \\ c_{2,X}^{l,m}\cos\left(-\alpha_X^{l,m}+\beta_X^{l,m}\right) & c_{2,X}^{l,m}\sin\left(-\alpha_X^{l,m}+\beta_X^{l,m}\right) \end{bmatrix} & ,otherwise \end{cases}$$

where

$$c_{1,X}^{l,m} = \sqrt{\frac{10^{\frac{CLD_X^{l,m}}{10}}}{1+10^{\frac{CLD_X^{l,m}}{10}}}}, \text{ and } c_{2,X}^{l,m} = \sqrt{\frac{1}{1+10^{\frac{CLD_X^{l,m}}{10}}}},$$

and where:

$$\beta_X^{l,m} = \arctan\left(\tan\left(\alpha_X^{l,m}\right)\frac{c_{2,X}^{l,m}-c_{1,X}^{l,m}}{c_{2,X}^{l,m}+c_{1,X}^{l,m}}\right), \text{ and } \alpha_X^{l,m} = \frac{1}{2}\arccos\left(\rho_X^{l,m}\right),$$

and where

$$\rho_X^{l,m} = \begin{cases} \max\left\{ICC_X^{l,m}, \lambda_0\left(10^{\frac{CLD_X^{l,m}}{20}}+10^{\frac{-CLD_X^{l,m}}{20}}\right)\right\} & ,m < resBands_X \\ ICC_X^{l,m} & ,otherwise \end{cases}, \text{ with } \lambda_0 = -11/72.$$

for $0 \le m < M_{\mathrm{proc}}, 0 \le l < L$.

And where

$$resBands_X = \begin{cases} \mathbf{m}_{\mathrm{resProc}}(X) & ,\mathbf{bsResidualPresent}(X)=1, bsResidualCoding=1 \\ 0 & ,otherwise \end{cases}.$$

6.5.3.3 5-1-5$_1$ configuration

The $\mathbf{R}_2^{l,m}$ matrix for the 5-1-5$_1$ configuration is defined according to:

$$\mathbf{R}_2^{l,m} = \begin{bmatrix} H11^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H11^{l,m}_{\mathrm{OTT}_0} & H11^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H12^{l,m}_{\mathrm{OTT}_0} & H11^{l,m}_{\mathrm{OTT}_3} H12^{l,m}_{\mathrm{OTT}_1} & H12^{l,m}_{\mathrm{OTT}_3} & 0 \\ H21^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H11^{l,m}_{\mathrm{OTT}_0} & H21^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H12^{l,m}_{\mathrm{OTT}_0} & H21^{l,m}_{\mathrm{OTT}_3} H12^{l,m}_{\mathrm{OTT}_1} & H22^{l,m}_{\mathrm{OTT}_3} & 0 \\ c^{l,m}_{1,\mathrm{OTT}_4} H21^{l,m}_{\mathrm{OTT}_1} H11^{l,m}_{\mathrm{OTT}_0} & c^{l,m}_{1,\mathrm{OTT}_4} H21^{l,m}_{\mathrm{OTT}_1} H12^{l,m}_{\mathrm{OTT}_0} & c^{l,m}_{1,\mathrm{OTT}_4} H22^{l,m}_{\mathrm{OTT}_1} & 0 & 0 \\ c^{l,m}_{2,\mathrm{OTT}_4} c^{l,m}_{2,\mathrm{OTT}_1} c^{l,m}_{1,\mathrm{OTT}_0} & 0 & 0 & 0 & 0 \\ H11^{l,m}_{\mathrm{OTT}_2} H21^{l,m}_{\mathrm{OTT}_0} & H11^{l,m}_{\mathrm{OTT}_2} H22^{l,m}_{\mathrm{OTT}_0} & 0 & 0 & H12^{l,m}_{\mathrm{OTT}_2} \\ H21^{l,m}_{\mathrm{OTT}_2} H21^{l,m}_{\mathrm{OTT}_0} & H21^{l,m}_{\mathrm{OTT}_2} H22^{l,m}_{\mathrm{OTT}_0} & 0 & 0 & H22^{l,m}_{\mathrm{OTT}_2} \end{bmatrix}$$

where, the elements are defined using the definition of arbitrary matrix elements $H11^{l,m}_{OTT_X} \cdots H22^{l,m}_{OTT_X}$ as outlined in subclause 6.5.3.2, for which,

$$\begin{aligned} CLD_X^{l,m} &= \mathbf{D}_{\mathrm{CLD}}\left(X,l,m\right) \\ ICC_X^{l,m} &= \mathbf{D}_{\mathrm{ICC}}\left(X,l,m\right) \end{aligned},$$

for $0 \le X < 5, 0 \le m < M_{\mathrm{proc}}, 0 \le l < L$. The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 23.

6.5.3.4 5-1-5$_2$ configuration

The $\mathbf{R}_2^{l,m}$ matrix for the 5-1-5$_2$ configuration is defined according to:

$$\mathbf{R}_2^{l,m} = \begin{bmatrix} H11^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H11^{l,m}_{\mathrm{OTT}_0} & H11^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H12^{l,m}_{\mathrm{OTT}_0} & H11^{l,m}_{\mathrm{OTT}_3} H12^{l,m}_{\mathrm{OTT}_1} & H12^{l,m}_{\mathrm{OTT}_3} & 0 \\ H21^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H11^{l,m}_{\mathrm{OTT}_0} & H21^{l,m}_{\mathrm{OTT}_3} H11^{l,m}_{\mathrm{OTT}_1} H12^{l,m}_{\mathrm{OTT}_0} & H21^{l,m}_{\mathrm{OTT}_3} H12^{l,m}_{\mathrm{OTT}_1} & H22^{l,m}_{\mathrm{OTT}_3} & 0 \\ H11^{l,m}_{\mathrm{OTT}_4} H21^{l,m}_{\mathrm{OTT}_1} H11^{l,m}_{\mathrm{OTT}_0} & H11^{l,m}_{\mathrm{OTT}_4} H21^{l,m}_{\mathrm{OTT}_1} H12^{l,m}_{\mathrm{OTT}_0} & H11^{l,m}_{\mathrm{OTT}_4} H22^{l,m}_{\mathrm{OTT}_1} & 0 & H12^{l,m}_{\mathrm{OTT}_4} \\ H21^{l,m}_{\mathrm{OTT}_4} H21^{l,m}_{\mathrm{OTT}_1} H11^{l,m}_{\mathrm{OTT}_0} & H21^{l,m}_{\mathrm{OTT}_4} H21^{l,m}_{\mathrm{OTT}_1} H12^{l,m}_{\mathrm{OTT}_0} & H21^{l,m}_{\mathrm{OTT}_4} H22^{l,m}_{\mathrm{OTT}_1} & 0 & H22^{l,m}_{\mathrm{OTT}_4} \\ c^{l,m}_{1,\mathrm{OTT}_2} H21^{l,m}_{\mathrm{OTT}_0} & c^{l,m}_{1,\mathrm{OTT}_2} H22^{l,m}_{\mathrm{OTT}_0} & 0 & 0 & 0 \\ c^{l,m}_{2,\mathrm{OTT}_0} c^{l,m}_{2,\mathrm{OTT}_2} & 0 & 0 & 0 & 0 \end{bmatrix}$$

where, the elements are defined using the definition of arbitrary matrix elements $H11^{l,m}_{\mathrm{OTT}_X} \cdots H22^{l,m}_{\mathrm{OTT}_X}$ as outlined in subclause 6.5.3.2, for which,

$$\begin{aligned} CLD_X^{l,m} &= \mathbf{D}_{\mathrm{CLD}}\left(X,l,m\right) \\ ICC_X^{l,m} &= \mathbf{D}_{\mathrm{ICC}}\left(X,l,m\right) \end{aligned}$$

for $0 \le X < 5, 0 \le m < M_{\mathrm{proc}}, 0 \le l < L$. The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 23.

6.5.3.5 5-2-5 configuration

For the 5-2-5 configuration the definition of the $\mathbf{R}_2^{l,m}$ depends on **bsTttModeLow** and **bsTttModeHigh**. As outlined in subclause 6.5.2.2.3 these correspond to different two-to-three up-mix styles in the TTT box implemented by the $\mathbf{R}_1^{l,m}$ matrix. Since for the prediction-based upmix in the TTT box, the prediction error can be replaced by an additional decorrelated signal (part of the same TTT box), the different modes of the TTT box affects the $\mathbf{R}_2^{l,m}$ matrix as well, since the $\mathbf{R}_2^{l,m}$ matrix does the mixing of all direct and diffuse (decorrelated) signals, including the decorrelated signal corresponding to the TTT box.

The decorrelator in the TTT box and the residual signal in the TTT box are mutually exclusive for the same frequency range (as all decorrelated signals and residual signals corresponding to the same OTT or TTT box).,

The matrix is consequently defined according to:

$$\mathbf{R}_2^{l,m} = \begin{bmatrix} \kappa_{\mathrm{TTT}_0}^{l,m} H11_{\mathrm{OTT}_1}^{l,m} & 0 & 0 & H12_{\mathrm{OTT}_1}^{l,m} & 0 & g_{\mathrm{TTT}_0}^{l,m} c_{1,\mathrm{OTT}_1}^{l,m} \\ H21_{\mathrm{OTT}_1}^{l,m} & 0 & 0 & H22_{\mathrm{OTT}_1}^{l,m} & 0 & 0 \\ 0 & \kappa_{\mathrm{TTT}_0}^{l,m} H11_{\mathrm{OTT}_2}^{l,m} & 0 & 0 & H12_{\mathrm{OTT}_2}^{l,m} & g_{\mathrm{TTT}_0}^{l,m} c_{1,\mathrm{OTT}_2}^{l,m} \\ 0 & H21_{\mathrm{OTT}_2}^{l,m} & 0 & 0 & H22_{\mathrm{OTT}_2}^{l,m} & 0 \\ 0 & 0 & \kappa_{\mathrm{TTT}_0}^{l,m} c_{1,\mathrm{OTT}_0}^{l,m} & 0 & 0 & -\sqrt{2} g_{\mathrm{TTT}_0}^{l,m} c_{1,\mathrm{OTT}_0}^{l,m} \\ 0 & 0 & c_{2,\mathrm{OTT}_0}^{l,m} & 0 & 0 & 0 \end{bmatrix}$$

where, the elements are defined using the definition of arbitrary matrix elements $H11_{\mathrm{OTT}_X}^{l,m} \cdots H22_{\mathrm{OTT}_X}^{l,m}$ as outlined in subclause 6.5.3.2, for which,

$$\begin{aligned} CLD_X^{l,m} &= \mathbf{D}_{\mathrm{CLD}}\left(X,l,m\right) \\ ICC_X^{l,m} &= \mathbf{D}_{\mathrm{ICC}}\left(X,l,m\right) \end{aligned},$$

for $0 \le X < 3, 0 \le m < M_{\mathrm{proc}}, 0 \le l < L$. The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 26.

The variables $\kappa_{TTT_0}^{l,m}$ and $g_{TTT_0}^{l,m}$ depend on **bsTttModeLow** or **bsTttModeHigh** as well as the presence of a residual signal. If no residual signal is present for the TTT box, i.e. $\mathbf{bsResidualPresent}(3) = 0$ and $bsResidualCoding = 0$, the variables are defined according to:

$$g_{\mathrm{TTT}_0}^{l,m} = \begin{cases} \sqrt{\dfrac{1-\left(\mathbf{D}_{\mathrm{ICC}}\left(3,l,m\right)\right)^2}{3}} & ,0 \le m < \mathbf{m}_{\mathrm{tttLowProc}}\left(0\right), \mathbf{bsTttModeLow}\left(0\right) = 0 \\ \sqrt{\dfrac{1-\left(\mathbf{D}_{\mathrm{ICC}}\left(3,l,m\right)\right)^2}{3}} & ,\mathbf{m}_{\mathrm{tttLowProc}}\left(0\right) \le m <, \mathbf{m}_{\mathrm{tttHighProc}}\left(0\right), \mathbf{bsTttModeHigh}\left(0\right) = 0 \\ 0 & otherwise \end{cases}$$

$$\kappa_{\mathrm{TTT}_0}^{l,m} = \begin{cases} \mathbf{D}_{\mathrm{ICC}}\left(3,l,m\right) & ,0 \le m < \mathbf{m}_{\mathrm{tttLowProc}}\left(0\right), \mathbf{bsTttModeLow}\left(0\right) = 0 \\ \mathbf{D}_{\mathrm{ICC}}\left(3,l,m\right) & ,\mathbf{m}_{\mathrm{tttLowProc}}\left(0\right) \le m < \mathbf{m}_{\mathrm{tttHighProc}}\left(0\right), \mathbf{bsTttModeHigh}\left(0\right) = 0 \\ 1 & otherwise \end{cases}$$

If a residual signal is present for the TTT box, i.e. **bsResidualPresent**$(3)=1$ and $bsResidualCoding=1$, the variables are defined according to:

$g^{l,m}_{\mathrm{TTT}_0}=0$ and $\kappa^{l,m}_{\mathrm{TTT}_0}=1$.

6.5.3.6 7-2-7 configuration

For the 7-2-7 configurations the definition of the $\mathbf{R}_2^{l,m}$ depends on **bsTttModeLow** and **bsTttModeHigh**, according to the same principles as for the 5-2-5 configuration.

For the 7-2-7_1 configuration the $\mathbf{R}_2^{l,m}$ matrix is defined according to:

$$\mathbf{R}_2^{l,m}=\begin{bmatrix}
\kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_1}H11^{l,m}_{OTT_3} & 0 & 0 & H12^{l,m}_{OTT_1}H11^{l,m}_{OTT_3} & 0 & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_1}H11^{l,m}_{OTT_3} & H12^{l,m}_{OTT_3} & 0\\
\kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_1}H21^{l,m}_{OTT_3} & 0 & 0 & H12^{l,m}_{OTT_1}H21^{l,m}_{OTT_3} & 0 & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_1}H21^{l,m}_{OTT_3} & H22^{l,m}_{OTT_3} & 0\\
H21^{l,m}_{OTT_1} & 0 & 0 & H22^{l,m}_{OTT_1} & 0 & 0 & 0 & 0\\
0 & \kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_2}H11^{l,m}_{OTT_4} & 0 & 0 & H12^{l,m}_{OTT_2}H11^{l,m}_{OTT_4} & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_2}H11^{l,m}_{OTT_4} & 0 & H12^{l,m}_{OTT_4}\\
0 & \kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_2}H21^{l,m}_{OTT_4} & 0 & 0 & H12^{l,m}_{OTT_2}H21^{l,m}_{OTT_4} & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_2}H21^{l,m}_{OTT_4} & 0 & H22^{l,m}_{OTT_4}\\
0 & H21^{l,m}_{OTT_2} & 0 & 0 & H22^{l,m}_{OTT_2} & 0 & 0 & 0\\
0 & 0 & \kappa^{l,m}_{TTT_0}c^{l,m}_{1,OTT_0} & 0 & 0 & -\sqrt{2}g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_0} & 0 & 0\\
0 & 0 & c^{l,m}_{2,OTT_0} & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

and for the 7-2-7_2 configuration the $\mathbf{R}_2^{l,m}$ matrix is defined according to:

$$\mathbf{R}_2^{l,m}=\begin{bmatrix}
\kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_1} & 0 & 0 & H12^{l,m}_{OTT_1} & 0 & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_1} & 0 & 0\\
\kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_1}H11^{l,m}_{OTT_3} & 0 & 0 & H12^{l,m}_{OTT_1}H11^{l,m}_{OTT_3} & 0 & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_1}H11^{l,m}_{OTT_3} & H12^{l,m}_{OTT_3} & 0\\
\kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_1}H21^{l,m}_{OTT_3} & 0 & 0 & H12^{l,m}_{OTT_1}H21^{l,m}_{OTT_3} & 0 & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_1}H21^{l,m}_{OTT_3} & H22^{l,m}_{OTT_3} & 0\\
0 & \kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_2} & 0 & 0 & H12^{l,m}_{OTT_2} & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_2} & 0 & 0\\
0 & \kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_2}H11^{l,m}_{OTT_4} & 0 & 0 & H12^{l,m}_{OTT_2}H11^{l,m}_{OTT_4} & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_2}H11^{l,m}_{OTT_4} & 0 & H12^{l,m}_{OTT_4}\\
0 & \kappa^{l,m}_{TTT_0}H11^{l,m}_{OTT_2}H21^{l,m}_{OTT_4} & 0 & 0 & H12^{l,m}_{OTT_2}H21^{l,m}_{OTT_4} & g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_2}H21^{l,m}_{OTT_4} & 0 & H22^{l,m}_{OTT_4}\\
0 & 0 & \kappa^{l,m}_{TTT_0}c^{l,m}_{1,OTT_0} & 0 & 0 & -\sqrt{2}g^{l,m}_{TTT_0}c^{l,m}_{1,OTT_0} & 0 & 0\\
0 & 0 & c^{l,m}_{2,OTT_0} & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

where, the elements are defined using the definition of arbitrary matrix elements $H11^{l,m}_{\mathrm{OTT}_X}\cdots H22^{l,m}_{\mathrm{OTT}_X}$ as outlined in subclause 6.5.3.2, for which,

$$CLD^{l,m}_X=\mathbf{D}_{\mathrm{CLD}}\left(X,l,m\right)$$
$$ICC^{l,m}_X=\mathbf{D}_{\mathrm{ICC}}\left(X,l,m\right)$$

for $0\le X<5, 0\le m<M_{\mathrm{proc}}, 0\le l<L$. The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 26.

The variables $\kappa_{\mathrm{TTT}_0}^{l,m}$ and $g_{\mathrm{TTT}_0}^{l,m}$ depend on **bsTttModeLow** or **bsTttModeHigh** as well as the presence of a residual signal. If no residual signal is present for the TTT box, i.e. $\mathbf{bsResidualPresent}(5)=0$ and $bsResidualCoding=0$, the variables are defined according to:

$$g_{\mathrm{TTT}_0}^{l,m}=\begin{cases}\sqrt{\dfrac{1-\left(\mathbf{D}_{\mathrm{ICC}}(5,l,m)\right)^2}{3}} & ,0\le m<\mathbf{m}_{\mathrm{tttLowProc}}(0),\mathbf{bsTttModeLow}(0)=0\\ \sqrt{\dfrac{1-\left(\mathbf{D}_{\mathrm{ICC}}(5,l,m)\right)^2}{3}} & ,\mathbf{m}_{\mathrm{tttLowProc}}(0)\le m<,\mathbf{m}_{\mathrm{tttHighProc}}(0),\mathbf{bsTttModeHigh}(0)=0\\ 0 & otherwise\end{cases}$$

$$\kappa_{\mathrm{TTT}_0}^{l,m}=\begin{cases}\mathbf{D}_{\mathrm{ICC}}(5,l,m) & ,0\le m<\mathbf{m}_{\mathrm{tttLowProc}}(0),\mathbf{bsTttModeLow}(0)=0\\ \mathbf{D}_{\mathrm{ICC}}(5,l,m) & ,\mathbf{m}_{\mathrm{tttLowProc}}(0)\le m<\mathbf{m}_{\mathrm{tttHighProc}}(0),\mathbf{bsTttModeHigh}(0)=0\\ 1 & otherwise\end{cases}$$

If a residual signal is present for the TTT box, i.e. $\mathbf{bsResidualPresent}(5)=1$ and $bsResidualCoding=1$, the variables are defined according to:

$g_{\mathrm{TTT}_0}^{l,m}=0$ and $\kappa_{\mathrm{TTT}_0}^{l,m}=1$.

6.5.3.7 7-5-7 configuration

For the 7-5-7 configurations the $\mathbf{R}_2^{l,m}$ matrix is defined according to the same principles as for the 5-1-5 configurations.

For the 7-5-7$_1$ configuration the $\mathbf{R}_2^{l,m}$ matrix is defined according to:

$$\mathbf{R}_2^{l,m}=\begin{bmatrix}H11_{\mathrm{OTT}_0}^{l,m} & 0 & 0 & 0 & 0 & 0 & H12_{\mathrm{OTT}_0}^{l,m} & 0\\ H21_{\mathrm{OTT}_0}^{l,m} & 0 & 0 & 0 & 0 & 0 & H22_{\mathrm{OTT}_0}^{l,m} & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & H11_{\mathrm{OTT}_1}^{l,m} & 0 & 0 & 0 & 0 & H12_{\mathrm{OTT}_1}^{l,m}\\ 0 & 0 & H21_{\mathrm{OTT}_1}^{l,m} & 0 & 0 & 0 & 0 & H22_{\mathrm{OTT}_1}^{l,m}\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}$$

and for the 7-5-7_2 configuration the $\mathbf{R}_2^{l,m}$ matrix is defined according to:

$$\mathbf{R}_2^{l,m} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & H11_{\mathrm{OTT}_0}^{l,m} & 0 & 0 & 0 & 0 & H12_{\mathrm{OTT}_0}^{l,m} & 0 \\ 0 & H21_{\mathrm{OTT}_0}^{l,m} & 0 & 0 & 0 & 0 & H22_{\mathrm{OTT}_0}^{l,m} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & H11_{\mathrm{OTT}_1}^{l,m} & 0 & 0 & 0 & H12_{\mathrm{OTT}_1}^{l,m} \\ 0 & 0 & 0 & H21_{\mathrm{OTT}_1}^{l,m} & 0 & 0 & 0 & H22_{\mathrm{OTT}_1}^{l,m} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

where, the elements are defined using the definition of arbitrary matrix elements $H11_{\mathrm{OTT}_X}^{l,m} \cdots H22_{\mathrm{OTT}_X}^{l,m}$ as outlined in subclause 6.5.3.2, for which,

$$\begin{aligned} CLD_X^{l,m} &= \mathbf{D}_{\mathrm{CLD}}(X,l,m) \\ ICC_X^{l,m} &= \mathbf{D}_{\mathrm{ICC}}(X,l,m) \end{aligned},$$

for $0 \le X < 2, 0 \le m < M_{\mathrm{proc}}, 0 \le l < L$. The OTT_X indexing corresponds to the labels of the OTT boxes as given in Figure 26.

6.5.4 Definition of post-matrix M3 for arbitrary trees

6.5.4.1 Introduction

The post-matrix $\mathbf{M}_3^{n,k}$ defines how the output from the pre-defined trees are extended in accordance with the arbitrary trees into the wanted number of output channel. It is defined according to:

$$\mathbf{M}_3^{n,k} = \begin{cases} \mathbf{W}_3^{l,k}\alpha(n,l) + (1-\alpha(n,l))\mathbf{W}_3^{-1,k}, & ,0 \le n \le \mathbf{t}(l), l = 0 \\ \mathbf{W}_3^{l,k}\alpha(n,l) + (1-\alpha(n,l))\mathbf{W}_3^{l-1,k}, & ,\mathbf{t}(l-1) < n \le \mathbf{t}(l), 1 \le l < L \end{cases}$$

where

$$\alpha(n,l) = \begin{cases} \dfrac{n+1}{\mathbf{t}(l)+1} & ,l = 0 \\ \dfrac{n-\mathbf{t}(l-1)}{\mathbf{t}(l)-\mathbf{t}(l-1)} & ,otherwise \end{cases}$$

and $\mathbf{W}_3^{l,k}$ can be processed by smoothing according to:

$$\mathbf{W}_3(l,k) = \begin{cases} \mathbf{s}_{\text{delta}}(l) \cdot \mathbf{R}_3\left(l,\kappa(k)\right) + \left(1-\mathbf{s}_{\text{delta}}(l)\right) \cdot \mathbf{W}_3(l-1,k) & ,\mathbf{S}_{\text{proc}}\left(l,\kappa(k)\right)=1 \\ \mathbf{R}_3\left(l,\kappa(k)\right) & ,\mathbf{S}_{\text{proc}}\left(l,\kappa(k)\right)=0 \end{cases}$$

and where $\kappa(k)$ is given in (Table A.30), where the first row is the hybrid band k, and the second row is the corresponding processing band.

$\mathbf{W}_3^{-1,k}$ corresponds to the last parameter set of the previous frame (zero for the first frame) and $\mathbf{W}_3(-1,k)$ refers to the last parameter set in the previous frame.

6.5.4.2 Calculation of $\mathbf{R}_3$ for an arbitrary configuration

The $\mathbf{R}_3^{l,m}$ matrix is defined by the CLD parameters for the arbitrary trees named ATD corresponding to the OTT modules as indicated in the bitstream.

The $\mathbf{R}_3^{l,m}$ matrix is defined according to:

$$\mathbf{R}_3^{l,m}(i,v) = \begin{cases} \prod_{p=0}^{\mathbf{Tree}_{\text{depth}}\left(v,i-i_{\text{offset}}(v)\right)-1} X_{\mathbf{Tree}\left(v,p,i-i_{\text{offset}}(v)\right)} & ,i_{\text{offset}}(v) \le i < i_{\text{offset}}(v) + \mathbf{Tree}_{\text{outChan}}(v) \\ 0 & ,otherwise \end{cases},$$

for $0 \le i < numChanOutAT$ and $0 \le v < numOutChan$ where

$$i_{\text{offset}}(v) = \begin{cases} \sum_{k=0}^{v-1} \mathbf{Tree}_{\text{outChan}}(k) & ,v>0 \\ 0 & otherwise \end{cases} \text{ and}$$

$$X_{\mathbf{Tree}\left(v,p,i_{\text{tmp}}\right)} = \begin{cases} c_{l,idx\left(v,p,i_{\text{tmp}}\right)}, & \mathbf{Tree}_{\text{sign}}\left(v,p,i_{\text{tmp}}\right)=1 \\ c_{r,idx\left(v,p,i_{\text{tmp}}\right)}, & \mathbf{Tree}_{\text{sign}}\left(v,p,i_{\text{tmp}}\right)=-1 \end{cases} \text{ where}$$

$$idx\left(v,p,i_{\text{tmp}}\right) = \begin{cases} \sum_{k=0}^{v-1} \mathbf{Tree}_{\text{outChan}}(k) + \mathbf{Tree}\left(v,p,i_{\text{tmp}}\right) & ,v>0 \\ \mathbf{Tree}\left(v,p,i_{\text{tmp}}\right) & otherwise \end{cases}$$

and where

$$c_{l,X} = \sqrt{\frac{CLD_{\text{lin},X}^2}{1+CLD_{\text{lin},X}^2}} \text{ and } c_{r,X} = \sqrt{\frac{1}{1+CLD_{\text{lin},X}^2}}, \text{ where } CLD_{\text{lin},X} = 10^{\frac{CLD_X}{20}}$$

and where

$$CLD_X^{l,m} = \mathbf{D}_{\mathrm{ATD}}(X,l,m), \quad 0 \le m < M, 0 \le l < L .$$

6.6 Decorrelators

6.6.1 Introduction

The decorrelators as depicted in Figure 22, Figure 25, and Figure 35 are implemented as reverberation filters in the QMF subband domain, with a reverberation filter in every hybrid subband with different filter characteristics depending on which hybrid subband it is situated in. The reverberation filters are IIR lattice filters, with filter coefficients chosen differently for the different decorrelators in order to have decorrelators that produce orthogonal signals, i.e. mutually decorrelated.

The de-correlation procedure consists of a number of steps. First of all, the outputs of the M1 pre-matrices, $\mathbf{v}^{n,k}$, are fed through a set of all-pass de-correlation filters. The filtered signals on their turn are input to an energy-shaping procedure, which spectrally shapes the de-correlated signals to more closely match the input signals in terms of its spectral envelope.

In the following subclauses the decorrelator procedure is outlined for an arbitrary decorrelator input signal $v_X^{n,k}$ as part of the $\mathbf{v}^{n,k}$ vector. In order to ensure orthogonal decorrelated signals, different decorrelators are derived from different filter coefficients, as given in Table A.26 to Table A.29 for the decorrelators $X = 0,...,9$. They correspond to the decorrelators in the different configurations (as outlined in subclause 6.4) according to Table 89.

Table 89 — Decorrelator-index as a function of decoder configuration

	Decorrelator $X = 0,...,9$									
configuration	0	1	2	3	4	5	6	7	8	9
5-1-5$_1$	$D_0^{\mathrm{OTT}}(\ )$	$D_1^{\mathrm{OTT}}(\ )$	$D_3^{\mathrm{OTT}}(\ )$	$D_2^{\mathrm{OTT}}(\ )$						
5-1-5$_2$	$D_0^{\mathrm{OTT}}(\ )$	$D_1^{\mathrm{OTT}}(\ )$	$D_3^{\mathrm{OTT}}(\ )$	$D_4^{\mathrm{OTT}}(\ )$						
5-2-5	$D_2^{\mathrm{OTT}}(\ )$	$D_1^{\mathrm{OTT}}(\ )$	$D_0^{\mathrm{TTT}}(\ )$							
7-2-7$_1$	$D_2^{\mathrm{OTT}}(\ )$	$D_1^{\mathrm{OTT}}(\ )$	$D_0^{\mathrm{TTT}}(\ )$	$D_3^{\mathrm{OTT}}(\ )$	$D_4^{\mathrm{OTT}}(\ )$					
7-2-7$_2$	$D_2^{\mathrm{OTT}}(\ )$	$D_1^{\mathrm{OTT}}(\ )$	$D_0^{\mathrm{TTT}}(\ )$	$D_3^{\mathrm{OTT}}(\ )$	$D_4^{\mathrm{OTT}}(\ )$					
7-5-7$_1$	$D_0^{\mathrm{OTT}}(\ )$	$D_1^{\mathrm{OTT}}(\ )$								
7-5-7$_2$	$D_0^{\mathrm{OTT}}(\ )$	$D_1^{\mathrm{OTT}}(\ )$								

6.6.2 De-correlation

6.6.2.1 Introduction

The de-correlation filters consist of a number of all-pass (IIR) sections preceded by a constant frequency-dependent delay. In principle the frequency axis is divided into a maximum of four different regions as a function of *bsDecorrConfig* variables as shown in Table 90, which correspond to the QMF split frequencies given in Table 48. In each region the length of the delay and the length of the filter coefficients vectors are

identical. Furthermore, in case of fractional delay decorrelators the filter coefficients are dependent on the hybrid subband index due to an additional phase rotation that is being applied.

Table 90 — Division of hybrid subbands as a function of *bsDecorrConfig*

bsDecorrConfig	0	1	2	3
k_0	0-7	0-7	-	reserved
k_1	8-20	8-55	0-20	reserved
k_2	21-29	56-70	21-70	reserved
k_3	30-70	-	-	reserved

6.6.2.2 Decorrelator IIR filtering

The reverberation filters for the different frequency regions given by Table 90, are implemented by a delay and a subsequent IIR filter. The delayed hybrid subband domain samples are obtained as:

$$d_{X,\mathrm{delay}}^{n,k} = \begin{cases} v_X^{n-8,k} & ,k \in k_0 \\ v_X^{n-7,k} & ,k \in k_1 \\ v_X^{n-2,k} & ,k \in k_2 \\ v_X^{n-1,k} & ,k \in k_3 \end{cases},$$

where $v_X^{n,k}$ for $n<0$ contains the buffered values of the last frame at position $numSlots - n$. The delayed hybrid subband domain samples are filtered as:

$$d_{X,\mathrm{filt}}^{n,k} = \frac{1}{a_X^{0,k}} \cdot \left(\sum_{l=0}^{L_{l_X}} b_X^{l,k} \cdot d_{X,\mathrm{delay}}^{n-l,k} - \sum_{l=1}^{L_{l_X}} a_X^{l,k} \cdot d_{X,\mathrm{filt}}^{n-l,k} \right),$$

where L_{l_X} is the length of the lattice coefficient vectors $l_{X,i}^n$ and the filter coefficients $a_X^{n,k}$ and $b_X^{n,k}$ are derived from the lattice coefficient vectors $l_{X,i}^n$ as outlined in the following subclause, where:

- For region k_0 and decorrelators $X = 0,...,9$, the length of the coefficient vector is given by $L_{l_X} = 20$, and the lattice coefficients $l_{X,0}^n$ are defined in Table A.26.
- For region k_1 and decorrelators $X = 0,...,9$, the length of the coefficient vector is given by $L_{l_X} = 15$, and the lattice coefficients $l_{X,1}^n$ are defined in Table A.27.
- For region k_2 and decorrelators $X = 0,...,9$, the length of the coefficient vector is given by $L_{l_X} = 6$, and the lattice coefficients $l_{X,1}^n$ are defined in Table A.28.
- For region k_3 and decorrelators $X = 0,...,9$, the length of the coefficient vector is given by $L_{l_X} = 3$, and the lattice coefficients $l_{X,1}^n$ are defined in Table A.29.

6.6.2.3 Derivation of filter coefficients from lattice coefficients

The filter coefficients are derived from the lattice coefficients differently depending on if fractional delay is used or not. For a fractional delay decorrelator, a fractional delay is applied by adding a frequency dependent phase-offset to the lattice coefficients.

First the lattice (reflection) coefficients $\phi_X^{n,k}$ are calculated as shown in Table 91 with q^k as given in Table 92, where the phase coefficients φ_X^n for $X = 0,...,9$ are defined in Table A.30.

Table 91 — Calculation of lattice coefficients

k	**Normal decorrelator**	**Fractional delay decorrelator**	n
k_0	$\phi_X^{n,k} = l_{X,0}^n$	$\phi_X^{n,k} = \exp\left(\frac{j}{2} \cdot \varphi_X^n \cdot q^k\right) \cdot l_{X,0}^n$	0,…,19
k_1	$\phi_X^{n,k} = l_{X,1}^n$	$\phi_X^{n,k} = \exp\left(\frac{j}{2} \cdot \varphi_X^n \cdot q^k\right) \cdot l_{X,1}^n$	0,…,14
k_2	$\phi_X^{n,k} = l_{X,2}^n$	$\phi_X^{n,k} = \exp\left(\frac{j}{2} \cdot \varphi_X^n \cdot q^k\right) \cdot l_{X,2}^n$	0,…,5
k_3	$\phi_X^{n,k} = l_{X,3}^n$	$\phi_X^{n,k} = \exp\left(\frac{j}{2} \cdot \varphi_X^n \cdot q^k\right) \cdot l_{X,3}^n$	0,…,2

Table 92 — Definition of q^k

k	q^k
0-5	0
6-7	1
8-9	2
10-70	$k-7$

The reflection coefficients are converted into the filter coefficients $a_X^{n,k}$ and $b_X^{n,k}$ according to:

$$a_X^{i,k} = \alpha_p(i)$$

$$b_X^{i,k} = \left(a_X^{L_{1_X}-i,k}\right)^*$$

for $0 \le i < L_{1_X}$, $p = L_{1_X}$,

where $\left(a_X^{L_{1X}-i,k}\right)^*$ denotes the complex conjugate of $a_X^{L_{1X}-i,k}$, and where $\alpha_p(i)$ are filter coefficients for a filter of order p, given by the following recursion:

$$\alpha_p(0) = 1$$
$$\alpha_p(p) = \phi_X^{p-1,k}$$
$$\alpha_p(i) = \alpha_{p-1}(i) + \phi_X^{i-1,k}\alpha_{p-1}^*(p-i)$$

for $1 \le i \le p-1$, $p = 1,2,\ldots,L_{1_X}$

6.6.3 Energy adjustment

The decorrelated signal undergoes energy adjustment in order to e.g. avoid audible reverberation tails after transients. A gain value applied to the decorrelated signal is calculated based on smoothed energy estimates of the input to the decorrelator and the corresponding output, as well as a given detection threshold $\gamma^{l,m}$. Hence, after the signals $d_{X,\text{filt}}^{n,k}$ have been obtained for $0 \le n < numSlots$, the following procedure is applied, again for $0 \le n < numSlots$. First the power per processing band m ($m = 0..28$) is calculated as:

$$E_{\text{v}}^{n,m} = \sum_{\forall k \in \kappa(k)=m} \left| v_X^{n,k} \right|^2 ,$$

$$E_{\text{d}}^{n,m} = \sum_{\forall k \in \kappa(k)=m} \left| d_{X,\text{filt}}^{n,k} \right|^2 ,$$

with $\kappa(k)$ defined in Table A.31. Next, low-pass filtering on the powers is applied as:

$$E_{\text{v,smooth}}^{n,m} = \alpha \cdot E_{\text{v,smooth}}^{n-1,m} + (1-\alpha) \cdot E_{\text{v}}^{n,m} ,$$

$$E_{\text{d,smooth}}^{n,m} = \alpha \cdot E_{\text{d,smooth}}^{n-1,m} + (1-\alpha) \cdot E_{\text{d}}^{n,m} ,$$

with $\alpha = 0.8$. For the first slot of the first frame both $E_{\text{v,smooth}}^{n-1,m}$ and $E_{\text{d,smooth}}^{n-1,m}$ with $n = 0$ are initialized as zero vectors. For the first slots of all other frames, both $E_{\text{v,smooth}}^{n-1,m}$ and $E_{\text{d,smooth}}^{n-1,m}$ (with $n = 0$) are set to the values of $E_{\text{v,smooth}}^{n,m}$ and $E_{\text{d,smooth}}^{n,m}$ of the previous frame at $n = numSlots - 1$. The energy-shaping gain vector is calculated as:

$$g_{\text{s}}^{n,m} = \begin{cases} \sqrt{\dfrac{\gamma E_{\text{v,smooth}}^{n,m}}{E_{\text{d,smooth}}^{n,m} + \varepsilon}} & , E_{\text{d,smooth}}^{n,m} > E_{\text{v,smooth}}^{n,m}\gamma \\ \min\left(\sqrt{\dfrac{E_{\text{v,smooth}}^{n,m}}{\gamma E_{\text{d,smooth}}^{n,m} + \varepsilon}}, 2 \right) & , \gamma E_{\text{d,smooth}}^{n,m} < E_{\text{v,smooth}}^{n,m} , \\ 1 & , otherwise \end{cases}$$

with $\gamma = 1.5$ and $\varepsilon = 1e-9$. Finally the decorrelator outputs are constructed as:

$$d_{X,\mathrm{s}}^{n,k} = g_{\mathrm{s}}^{n,\kappa(k)} \cdot d_{X,\mathrm{filt}}^{n,k}.$$

6.7 Subband Domain Temporal Processing (STP)

6.7.1 Introduction

The subband domain temporal processing tool is used for shaping the envelope of the diffuse signal portion of each output channel to substantially match the temporal shape of the transmitted downmix signal. It consists of envelope estimation for both direct and diffuse sound respectively, the computation of the envelope ratio, and a shaping of the upper spectral part of the diffuse signal.

The STP tool is activated by signalling $bsTempShapeConfig = 1$ and it is used to process the diffuse portion of the output signal of an upmixed output channel if $\mathbf{bsTempShapeEnableChannel}(ch) = 1$ is signaled for this channel.

6.7.2 Overview Diagram

Figure 36 provides a schematic block diagram of the different processing steps for STP:

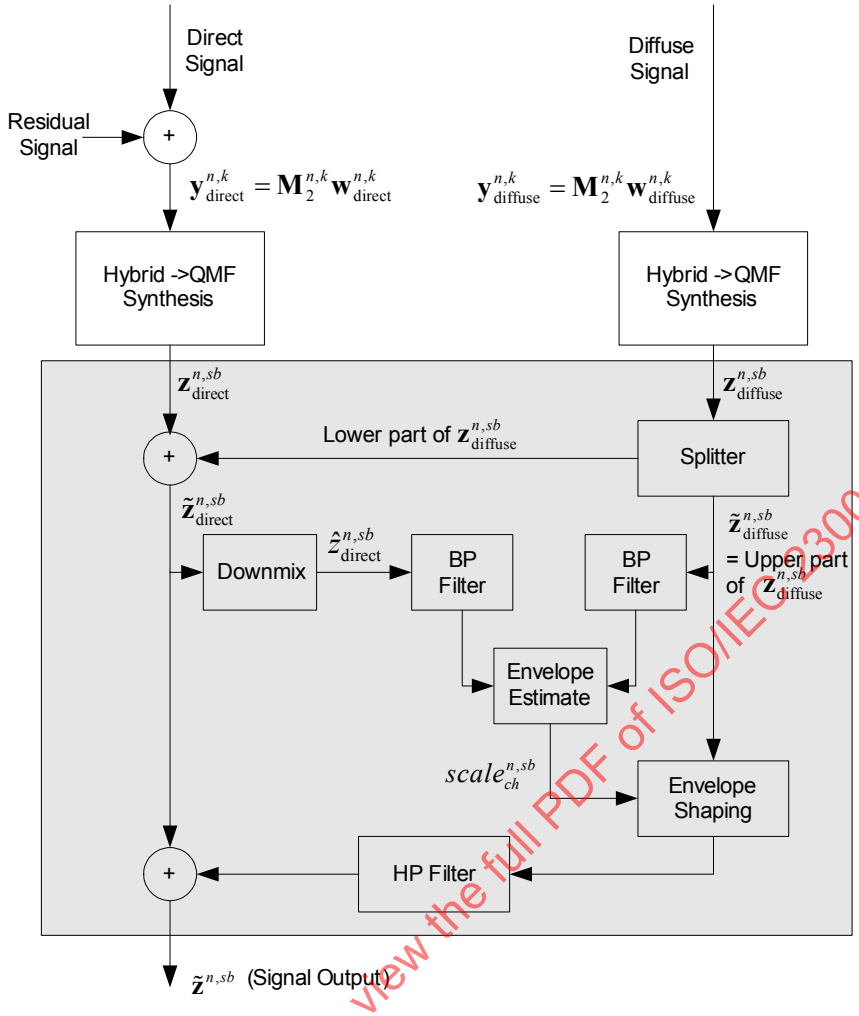

Figure 36 — Subband Domain Temporal Processing

For both HQ MPEG Surround and LP MPEG Surround, the STP operates on the diffuse signal $\mathbf{z}^{n,sb}_{\text{diffuse}}$ and the direct signal $\mathbf{z}^{n,sb}_{\text{direct}}$ output by the Nyquist synthesis filterbank. Since the shaping of the diffuse signal is done on the high band part of $\mathbf{z}^{n,sb}_{\text{diffuse}}$ only, a splitter is employed to remix $\mathbf{z}^{n,sb}_{\text{direct}}$ and $\mathbf{z}^{n,sb}_{\text{diffuse}}$ into $\tilde{\mathbf{z}}^{n,sb}_{\text{direct}}$ and $\tilde{\mathbf{z}}^{n,sb}_{\text{diffuse}}$ according to:

$$\tilde{\mathbf{z}}^{n,sb}_{\text{direct}} = \begin{cases} \mathbf{z}^{n,sb}_{\text{direct}} + \mathbf{z}^{n,sb}_{\text{diffuse}} & ,0 \le sb < 5 \\ \mathbf{z}^{n,sb}_{\text{direct}} & ,5 \le sb < 64 \end{cases}$$

$$\tilde{\mathbf{z}}^{n,sb}_{\text{diffuse}} = \begin{cases} 0 & ,0 \le sb < 5 \\ \mathbf{z}^{n,sb}_{\text{diffuse}} & ,5 \le sb < 64 \end{cases}$$

Furthermore, $\tilde{z}^{n,sb}_{\text{direct},ch}$ and $\tilde{z}^{n,sb}_{\text{diffuse},ch}$ respectively denote the direct and diffuse output for a channel *ch* from the splitter.

6.7.3 Downmix

STP is functional to shape the temporal envelope of the diffuse signal portion of the upmixed signal to match the temporal envelope of the transmitted downmix. To alleviate the need for delay alignment of the original transmitted downmix with respect to the spatial upmix, a downmix of the spatial upmix is computed being an approximation of the transmitted downmix:

In the following, unless stated otherwise, the indices *n* and *sb* are defined as

$0 \le n < numSlots \quad 6 \le sb < 24$ with $numSlots = bsFrameLength + 1$.

For 5-1-5, a common direct downmix signal is obtained as follows:

$$\hat{z}^{n,sb}_{\text{direct}} = \sum_{ch_m} \tilde{z}^{n,sb}_{\text{direct},ch_m}$$

where ch_m comprises the following upmix channels:

Table 93 — Defining ch_m for 5-1-5

Configuration	ch_m
5-1-5	L, Ls, C, R, Rs

For 5-2-5, 7-2-7_1, 7-2-7_2, 7-5-7_1 and 7-5-7_2, two direct downmix signals are obtained as follows:

$$\hat{z}^{n,sb}_{\text{direct\_l}} = \sum_{ch_l} \tilde{z}^{n,sb}_{\text{direct},ch_l}$$

$$\hat{z}^{n,sb}_{\text{direct\_r}} = \sum_{ch_r} \tilde{z}^{n,sb}_{\text{direct},ch_r}$$

where ch_l and ch_r comprise the following upmix channels for various configurations:

Table 94 — Defining ch_l and ch_r for X-2-X and X-5-X

Configuration	ch_l	ch_r
5-2-5	L, Ls	R, Rs
7-2-7_1	L, Lc, Ls	R, Rc, Rs
7-2-7_2	L, Lsr, Ls	R, Rsr, Rs
7-5-7_1	L, Lc	R, Rc
7-5-7_2	Lsr, Ls	Rsr, Rs

6.7.4 Envelope Estimation

In a second step, the broadband envelopes of the downmix and the envelopes of the diffuse signal portion of each upmix channel are estimated:

For 5-1-5, for each sample of a direct signal, its direct energy is computed as

$$E_{\text{direct}}^{n,sb} = \left| \hat{z}_{\text{direct}}^{n,sb} \cdot BP^{sb} \cdot GF^{sb} \right|^2$$

where a bandpass factor BP^{sb} and a spectral flattening factor GF^{sb} are defined in Table 95.

Table 95 — Defining BP^{sb} and GF^{sb}

sb	BP^{sb}	GF^{sb}	sb	BP^{sb}	GF^{sb}
0	0.0000	0.0000	13	0.9984	0.0075
1	0.0005	0.0000	14	0.9908	0.0086
2	0.0092	0.0000	15	0.9639	0.0099
3	0.0587	0.0000	16	0.8952	0.0111
4	0.2580	0.0000	17	0.7711	0.0125
5	0.7392	0.0000	18	0.6127	0.0141
6	0.9791	0.0001	19	0.4609	0.0158
7	0.9993	0.0009	20	0.3391	0.0179
8	1.0000	0.0019	21	0.2493	0.0203
9	1.0000	0.0029	22	0.1848	0.0232
10	1.0000	0.0040	23	0.1387	0.0266
11	1.0000	0.0052	24	0.1053	0.0307
12	0.9999	0.0063			

The direct energy for each time slot is computed as follows:

$$En_{\text{direct}}^{n} = \delta \sum_{sb=6}^{7} E_{\text{direct}}^{n,sb} + \sum_{sb=8}^{24} E_{\text{direct}}^{n,sb}$$

where $\delta = 1$ for the HQ MPEG Surround and $\delta = 0.5$ for the LP MPEG Surround.

The energy envelope for the direct signal is defined as

$$\mathbf{Env}_{\text{direct}}(n) = \alpha \cdot \mathbf{Env}_{\text{direct}}(n-1) + (1-\alpha)\mathbf{En}_{\text{direct}}(n)$$

where $\alpha = 0.95$ and $\mathbf{Env}_{\text{direct}}(-1)$ is defined as the last value in the previous frame and initialized as $\mathbf{Env}_{\text{direct}}(-1) = 0$.

The energy envelope is sampled once every 32 consecutive time slots, regardless of *numSlots*, and stored in a holding parameter:

$$Env_{\text{direct\_hold}} = \begin{cases} Env_{\text{direct}}(n) & ,if\ \tilde{n}(n) = 0 \\ Env_{\text{direct\_hold}} & ,otherwise \end{cases}$$

To achieve this, a counter parameter is defined as

$$\tilde{n}(n) = \mathrm{mod}\left(\tilde{n}(n-1)+1, 32\right)$$

where $\tilde{n}(-1)$ corresponds to the $\tilde{n}$ of the last time slot of the previous frame and initialized as $\tilde{n}(-1) = 0$.

The normalized direct energy is computed as follows:

$$E^{n}_{\mathrm{direct\_norm}} = \frac{En^{n}_{\mathrm{direct}}}{Env_{\mathrm{direct\_hold}} + \varepsilon}$$

For 5-2-5, 7-2-7$_1$, 7-2-7$_2$, 7-5-7$_1$ and 7-5-7$_2$, since there are two direct signals, $E_{\mathrm{direct\_norm\_l}}$ and $E_{\mathrm{direct\_norm\_r}}$ can be obtained in a similar manner.

Likewise, for each sample of a diffuse signal, its diffuse energy is computed as follows:

$$E^{n,sb}_{\mathrm{diffuse},ch} = \left|\tilde{z}^{n,sb}_{\mathrm{diffuse},ch} \cdot BP^{sb} \cdot GF^{sb}\right|^2 .$$

The diffuse energy for each time slot and channel (according to Table 67) is computed as follows:

$$En^{n}_{\mathrm{diffuse},ch} = \delta \sum_{sb=6}^{7} E^{n,sb}_{\mathrm{diffuse},ch} + \sum_{sb=8}^{24} E^{n,sb}_{\mathrm{diffuse},ch}$$

where $\delta = 1$ for the HQ MPEG Surround and $\delta = 0.5$ for the LP MPEG Surround.

The energy envelope for each diffuse signal is defined as

$$Env^{n}_{\mathrm{diffuse},ch} = \alpha \cdot Env^{n-1}_{\mathrm{diffuse},ch} + (1-\alpha) \cdot En^{n}_{\mathrm{diffuse},ch}$$

where $\alpha = 0.95$ and $Env_{\mathrm{diffuse},ch}(-1)$ is defined as the last value in the previous frame of the corresponding channel and initialized as $Env_{\mathrm{diffuse},ch}(-1) = 0$.

The energy envelope of the diffuse signal is also sampled once every 32 consecutive time slots and stored in a holding parameter:

$$Env_{\mathrm{diffuse\_hold},ch} = \begin{cases} Env_{\mathrm{diffuse},ch}(n) & ,if\ \tilde{n}(n) = 0 \\ Env_{\mathrm{diffuse\_hold},ch} & ,otherwise \end{cases}$$

The normalized diffuse energy is computed as follows:

$$E^{n}_{\mathrm{diffuse\_norm},ch} = \frac{En^{n}_{\mathrm{diffuse},ch}}{Env_{\mathrm{diffuse\_hold},ch} + \varepsilon}$$

6.7.5 Scale Factors

In a next step, the ratio of the envelope estimates is calculated by relating the appropriate downmix envelope to each of the upmix envelopes. This yields the scale factors for the final envelope processing:

For each time slot of a diffuse signal, a scale factor is computed as follows:

For 5-1-5

$$scale_{ch}^{n} = \sqrt{\frac{E_{\text{direct\_norm}}^{n}}{E_{\text{diffuse\_norm},ch}^{n} + \varepsilon}} \qquad ch \in \{ch_m\}$$

For 5-2-5, 7-2-7$_1$, 7-2-7$_2$, 7-5-7$_1$ and 7-5-7$_2$

$$scale_{ch}^{n} = \sqrt{\frac{E_{\text{direct\_norm\_l}}^{n}}{E_{\text{diffuse\_norm},ch}^{n} + \varepsilon}} \quad ,ch \in \{ch_l\}$$

$$scale_{ch}^{n} = \sqrt{\frac{E_{\text{direct\_norm\_r}}^{n}}{E_{\text{diffuse\_norm},ch}^{n} + \varepsilon}} \quad ,ch \in \{ch_r\}$$

The scale factor further undergoes damping, limiting and smoothing:

$$scale_{\text{damp},ch}^{n} = \lambda_1 + (1-\lambda_1)\cdot scale_{ch}^{n}$$
$$scale_{\text{limit},ch}^{n} = \min\left(\max\left(scale_{\text{damp},ch}^{n}, 1/\lambda_2\right), \lambda_2\right)$$
$$scale_{\text{smooth},ch}(n) = \lambda_3 \cdot scale_{\text{limit},ch}^{n} + (1-\lambda_3)\cdot scale_{\text{smooth},ch}(n-1)$$

where $\lambda_1 = 0.1$, $\lambda_2 = 2.82$, $\lambda_3 = 0.45$, and $scale_{\text{smooth},ch}^{-1}$ is defined as the last value in the previous frame of the corresponding channel and initialized as $scale_{\text{smooth},ch}(-1) = 0$.

6.7.6 Envelope Processing and Signal Mixing

In the last step, the diffuse signal portion of each channel is processed by applying the scale factors (such that the diffuse signal portion of each channel when processed using the scale factors has a temporal envelope substantially matching the temporal envelope of the downmix signal). Finally, the processed diffuse signal of each channel is mixed with the corresponding direct signal portion:

If envelope processing is signaled by $\mathbf{bsTempShapeEnableChannel}(ch) = 1$ to be active for a particular channel ch its diffuse signal portion is scaled by the processed scale factors, otherwise, and for channels that are not processed by STP, no scaling is performed:

$$scale_{\text{apply},ch}^{n} = \begin{cases} 1 & ,if\ \mathbf{bsTempShapeEnableChannel}(ch) = 0 \\ scale_{\text{smooth},ch}^{n} & ,if\ \mathbf{bsTempShapeEnableChannel}(ch) = 1 \end{cases}$$

In the same step the diffuse signal is added to the direct signals as follows,

$$\tilde{z}_{ch}^{n,sb} = \tilde{z}_{\text{direct},ch}^{n,sb} + \tilde{z}_{\text{diffuse},ch}^{n,sb} \cdot scale_{\text{apply},ch}^{\text{n}} \cdot BP^{sb} \quad ,if\ 0 \le sb \le 8$$
$$\tilde{z}_{ch}^{n,sb} = \tilde{z}_{\text{direct},ch}^{n,sb} + \tilde{z}_{\text{diffuse},ch}^{n,sb} \cdot scale_{\text{apply},ch}^{\text{n}} \quad ,if\ 9 \le sb \le 63$$

For channels that are not processed by STP,

$$\tilde{z}_{ch}^{n,sb} = \tilde{z}_{\text{direct},ch}^{n,sb} + \tilde{z}_{\text{diffuse},ch}^{n,sb} \qquad 0 \le sb \le 63$$

Finally, in HQ MPEG Surround, $\tilde{z}_{ch}^{n,sb}$ is passed to the synthesis QMF filterbank to produce the time domain signals. In LP MPEG Surround, $\tilde{z}_{ch}^{n,sb}$ is passed to the complex to real converter (subclause 6.10.2.4) for further processing.

6.8 Guided Envelope Shaping (GES)

6.8.1 Introduction

The guided envelope shaping (GES) restores the broadband envelope of the synthesized output signal. It comprises a modified upmix procedure, followed by envelope flattening and reshaping for the direct signal portion of each output channel.

For the reshaping, parametric broadband envelope side information contained in the bitstream is used. The side information consists of ratios $envRatio_{ch}$ relating the transmitted downmix signal's envelope to the original input channel signal's envelope. In the decoder, from these ratios gain factors are derived to be applied to the direct signal on each time slot in a frame of a given output channel. The diffuse sound portion of each channel is not altered by the tool. Figure 37 shows the position of the GES tool in the decoder signal flow.

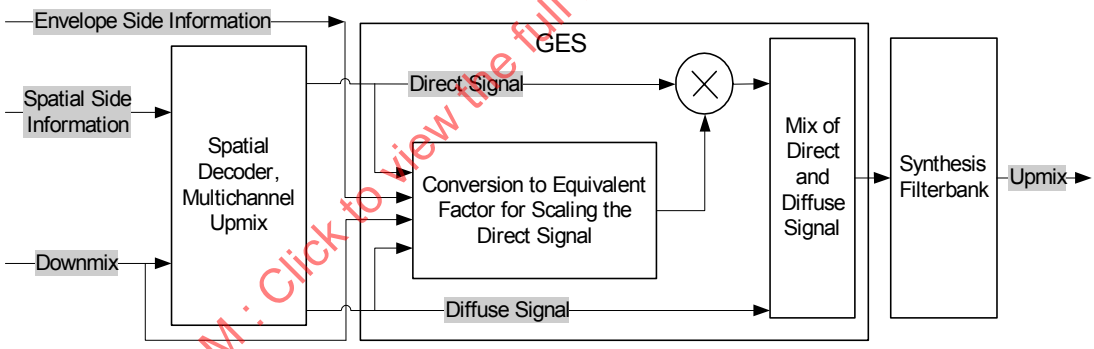

Figure 37 — Placement of the Guided Envelope Shaping tool in the decoder

The GES calculations described in detail in the following subclauses are performed if $bsTempShapeConfig=2$. However, the actual envelope shaping by application of these gain factors to the upmixed signal of a channel is only performed if $\mathbf{bsTempShapeEnableChannel}(ch)=1$.

If the GES is enabled, direct and diffuse signals are synthesized separately using a modified post mixing in the hybrid subband domain according to

$$\mathbf{y}_{\text{direct}}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}_{\text{direct}}^{n,k}$$

$$\mathbf{y}_{\text{diffuse}}^{n,k} = \mathbf{M}_2^{n,k}\mathbf{w}_{\text{diffuse}}^{n,k}$$

for $0 \le k < K$ and $0 \le n < numSlots$. The direct outputs hold the direct signal and the residual signal (if present). The diffuse outputs provide the diffuse signal. Solely the direct signal is further processed by the GES tool. Nevertheless the mix is needed for the normalized envelope extraction and is defined as:

$$\mathbf{y}_{\text{ges}}^{n,k} = \mathbf{y}_{\text{direct}}^{n,k} + \mathbf{y}_{\text{diffuse}}^{n,k}$$

The GES tool extracts an envelope for certain downmix channels dependant on tree configuration and for those output channels that are upmixed from corresponding downmix channels by the spatial decoder except the LFE channel. The envelope extraction process is described in detail first and is subsequently referred to in the description of the actual shaping process.

6.8.2 Estimation of normalized envelopes

The normalized envelope $Env_{ZZZ,ch_{ZZZ}}^{n}$ is a smoothly whitened envelope, or spectrally flattened envelope. To estimate the normalized envelope firstly, for each slot n in a frame, it is needed to calculate the parameter bands energy $E_{\text{slot},ch}^{n,\kappa}$ for a certain range of parameter bands κ .and a total average (broadband) energy $E_{\text{total},ch}^{n}$. This is done for all input ch_{input} and output ch_{output} channels defined in

Table 97 and Table 98.

$$E_{\text{slot}ZZZ,ch}^{n,\kappa} = \sum_{\tilde{k}=kfunc(\kappa)}^{kfunc(\kappa+1)-1} u_{ZZZ,ch}^{n,\tilde{k}} \left(u_{ZZZ,ch}^{n,\tilde{k}}\right)^{*} \qquad ,ch \in \{ch_{ZZZ}\}$$

$$E_{\text{total}ZZZ,ch}^{n} = \frac{1}{\kappa_{\text{stop}} - \kappa_{\text{start}} + 1} \sum_{\kappa=\kappa_{\text{start}}}^{\kappa_{\text{stop}}} E_{\text{slot}ZZZ,ch}^{n,\kappa} \qquad ,ch \in \{ch_{ZZZ}\}$$

where

$$\kappa_{\text{start}} = 10 \text{ and } \kappa_{\text{stop}} = 18$$

and

$$u_{ZZZ,ch_{ZZZ}}^{n,\tilde{k}} = \begin{cases} y_{\text{ges},ch_{\text{output}}}^{n,\tilde{k}} & ,if\ ZZZ = \text{output} \\ x_{ch_{\text{input}}}^{n,\tilde{k}} & ,if\ ZZZ = \text{input} \end{cases}$$

for $ZZZ \in \{\text{input},\text{output}\}$, $\kappa_{\text{start}} \le \kappa \le \kappa_{\text{stop}}$ and $0 \le n < numSlots$ where $numSlots = bsFrameLength + 1$. The function $kfunc(\kappa)$ is defined in Table 96 . ch_{input} and ch_{output} are defined in Table 97 and Table 98.

Table 96 — Hybrid to GES band function $kfunc(\kappa)$

κ	10	11	12	13	14	15	16	17	18	19
$kfunc(\kappa)$	12	13	14	15	16	18	21	25	30	42

Table 97 — Output channels ch_{output} for various configurations

Configuration	ch_{output}
5-1-5	L,Ls,C,R,Rs
5-2-5	L,Ls,C,R,Rs
7-2-7_1	L,Lc,Ls,C,R,Rc,Rs
7-2-7_2	L,Lsr,Ls,C,R,Rsr,Rs
7-5-7_1	L,Lc,R,Rc
7-5-7_2	Lsr,Ls,Rsr,Rs

Table 98 — Input channels ch_{input} for various configurations

Configuration	ch_{input}
5-1-5	M_0
5-2-5	L_0,R_0
7-2-7	L_0,R_0
7-5-7_1	L_0,R_0
7-5-7_2	Ls_0,Rs_0

Subsequently a long-term energy average is defined for all input and output channels for both the parameter bands energy $E^{n,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}}$ and the total average (broadband) energy $E^{n}_{\text{total}ZZZ,ch_{ZZZ}}$ as:

$$\bar{E}^{n,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}} = (1-\alpha)\cdot E^{n,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}} + \alpha\cdot \bar{E}^{n-1,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}}$$
$$\bar{E}^{n}_{\text{total}ZZZ,ch_{ZZZ}} = (1-\alpha)\cdot E^{n}_{\text{total}ZZZ,ch_{ZZZ}} + \alpha\cdot \bar{E}^{n-1}_{\text{total}ZZZ,ch_{ZZZ}}$$

for $ZZZ \in \{\text{input},\text{output}\}$, $\kappa_{\text{start}} \le \kappa \le \kappa_{\text{stop}}$ and $0 \le n < numSlots$ with $\kappa_{\text{start}} = 10$, $\kappa_{\text{stop}} = 18$ and where α is a weighting factor corresponding to a first order IIR lowpass (approx. 400 ms time constant):

$$\alpha = \exp\left(-\frac{64}{0.4\cdot 44100}\right)$$

where $\bar{E}^{-1,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}}$ and $\bar{E}^{-1}_{\text{total}ZZZ,ch_{ZZZ}}$ are defined as the last values in the previous frame of the corresponding channel, initialized as $\bar{E}^{-1,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}} = 0$ and $\bar{E}^{-1}_{\text{total}ZZZ,ch_{ZZZ}} = 0$.

The parameter bands energy $\bar{E}^{n,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}}$ and averaged broadband energy $\bar{E}^{n}_{\text{total}ZZZ,ch_{ZZZ}}$ are used to calculate whitening weights as defined below:

$$w^{n,\kappa}_{ZZZ,ch_{ZZZ}} = \frac{\bar{E}^{n}_{\text{total}ZZZ,ch_{ZZZ}}}{\bar{E}^{n,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}} + \varepsilon}$$

The broadband envelope estimate is obtained by summation of the weighted contributions of the parameter bands, normalizing on a long-term energy average and calculation of the square root. Furthermore the 5-2-5 and 7-2-7 trees need a centre-channel envelope for C_0 which is not available as an input signal and therefore the envelope for C_0 is defined as an average of L_0 and R_0 envelopes:

$$Env^{n}_{ZZZ,ch_{ZZZ}} = \begin{cases} \dfrac{Env^{n}_{\text{input},L_0} + Env^{n}_{\text{input},R_0}}{2} & , \begin{cases} ZZZ = \text{input} \\ ch_{\text{input}} = C_0 \end{cases} \\ \sqrt{\dfrac{EnvAbs^{n}_{ZZZ,ch_{ZZZ}}}{\overline{Env}^{n}_{ZZZ,ch_{ZZZ}}}} & , otherwise \end{cases}$$

where

$$EnvAbs^{n}_{ZZZ,ch_{ZZZ}} = \sum_{\kappa=\kappa_{\text{start}}}^{\kappa_{\text{stop}}} w^{n,\kappa}_{ZZZ,ch_{ZZZ}} \cdot E^{n,\kappa}_{\text{slot}ZZZ,ch_{ZZZ}}$$

$$\overline{Env}^{n}_{ZZZ,ch_{ZZZ}} = (1-\beta)\cdot EnvAbs^{n}_{ZZZ,ch_{ZZZ}} + \beta \cdot \overline{Env}^{n-1}_{ZZZ,ch_{ZZZ}}$$

for $ZZZ \in \{\text{input},\text{output}\}$, $0 \le n < numSlots$ and where β is a weighting factor corresponding to a first order IIR lowpass (approx. 40 ms time constant):

$$\beta = \exp\left(-\frac{64}{0.04 \cdot 44100}\right)$$

Any negative index is defined as the last value in the previous frame and the initial value is set to:

$$\overline{Env}^{-1}_{ZZZ,ch_{ZZZ}} = 32768^2$$

6.8.3 Time Envelope Shaping

6.8.3.1 Introduction

The time envelope shaping process consists of a flattening of the direct sound envelope for each output channel followed by a reshaping towards a target envelope.

6.8.3.2 Estimation of the envelope adjustment gain

For the corresponding configuration, the target envelope is obtained by estimating the envelope of the transmitted downmix $Env^{n}_{\text{input},ch_{\text{input}}}$ for each input channel (defined in Table 98) and subsequently scaling it with encoder transmitted and requantized envelope ratios $envRatio^{n}_{ch}$.

The gain curve g^{n}_{ch} for all slots in a frame is calculated for each output channel (defined in Table 99) by estimating its envelope Env^{n}_{ch} and relating it to the target envelope.

Finally, this gain curve is converted into an effective gain curve for solely scaling the direct part of the upmixed channel:

$$ratio^{n}_{ch_{\text{output}}} = \min\left(\max\left(g^{n}_{ch_{\text{output}}} + ampRatio^{n}_{ch_{\text{output}}} \cdot \left(g^{n}_{ch_{\text{output}}} - 1\right), 1/\lambda_4\right), \lambda_4\right)$$

where $\lambda_4 = 4$, for $0 \le n < numSlots$ and where

$$ampRatio^{n}_{ch_{\text{output}}} = \sqrt{\frac{\sum_{k=8}^{K-1} \left| y^{n,k}_{\text{diffuse},ch_{\text{output}}} \right|^2}{\sum_{k=8}^{K-1} \left| y^{n,k}_{\text{direct},ch_{\text{output}}} \right|^2 + \varepsilon}},$$

$$g^{n}_{ch_{\text{output}}} = \frac{envRatio^{n}_{ch_{\text{output}}} \cdot Env^{n}_{\text{output},Dch\left(ch_{\text{output}}\right)}}{Env^{n}_{\text{input},ch_{\text{output}}}},$$

Where the downmix channel mapping function $Dch\left(ch_{\text{output}}\right)$ is defined in Table 99.

Table 99 — Downmix channels $Dch\left(ch_{\text{output}}\right)$ for various configurations

Configuration	bsTreeConfig	$Dch\left(ch_{\text{output}}\right)$
5-1-5	0,1	$Dch\left(ch_{\text{output}}\right) = \mathrm{M}_0 \quad ,if\ ch_{\text{output}} \in \{\mathrm{L,Ls,C,R,Rs}\}$
5-2-5	2	$Dch\left(ch_{\text{output}}\right) = \begin{cases} \mathrm{C}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{C}\} \\ \mathrm{L}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{L,Ls}\} \\ \mathrm{R}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{R,Rs}\} \end{cases}$
7-2-7$_1$	3	$Dch\left(ch_{\text{output}}\right) = \begin{cases} \mathrm{C}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{C}\} \\ \mathrm{L}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{L,Lc,Ls}\} \\ \mathrm{R}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{R,Rc,Rs}\} \end{cases}$
7-2-7$_2$	4	$Dch\left(ch_{\text{output}}\right) = \begin{cases} \mathrm{C}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{C}\} \\ \mathrm{L}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{L,Lsr,Ls}\} \\ \mathrm{R}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{R,Rsr,Rs}\} \end{cases}$
7-5-7$_1$	5	$Dch\left(ch_{\text{output}}\right) = \begin{cases} \mathrm{L}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{L,Lc}\} \\ \mathrm{R}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{R,Rc}\} \end{cases}$
7-5-7$_2$	6	$Dch\left(ch_{\text{output}}\right) = \begin{cases} \mathrm{Ls}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{Lsr,Ls}\} \\ \mathrm{Rs}_0 & ,if\ ch_{\text{output}} \in \{\mathrm{Rsr,Rs}\} \end{cases}$

6.8.3.3 Application of the envelope adjustment gain curve

The envelope adjustment gain curve is applied to the upper subbands of the direct part of the upmixed channel $ch \in \{ch_{\text{output}}\}$ if $\mathbf{bsTempShapeEnableChannel}(ch) = 1$ is signaled for this channel in the side information, otherwise and for all remaining $ch \notin \{ch_{\text{output}}\}$ (according to Table 67) the direct signal is copied:

$$\tilde{y}^{n,k}_{\text{direct},ch} = \begin{cases} ratio^{n}_{ch} \cdot y^{n,k}_{\text{direct},ch} & ,if \begin{cases} \mathbf{bsTempShapeEnableChannel}(ch) = 1 \\ 8 \le k < K \end{cases} \\ y^{n,k}_{\text{direct},ch} & ,otherwise \end{cases}$$

6.8.4 Mixing of direct and diffuse signal

In a last step, the (processed) direct signals and the unaltered diffuse signals are mixed in the hybrid subband domain:

$$\mathbf{y}^{n,k} = \tilde{\mathbf{y}}^{n,k}_{\text{direct}} + \mathbf{y}^{n,k}_{\text{diffuse}}$$

for $0 \le k < K$ and $0 \le n < numSlots$, where $numSlots = bsFrameLength + 1$.

6.9 Residual coding

6.9.1 Introduction

This subclause describes the transformation of the critically sampled MDCT coefficients, as obtained by decoding an individual channel stream (ICS) element (subclause 6.1.5) for residual coding or either an individual channel stream or a channel pair element (CPE) for arbitrary downmix enhancement (6.1.6), to the non-critically sampled complex valued hybrid QMF domain signal $x^{n,k}_{\text{res0}}$ in the case of a TTT box, $v^{n,k}_{\text{res1}}$, $v^{n,k}_{\text{res2}}$, ... in the case of an OTT box or $x^{n,k}_{\text{res1}}$, $x^{n,k}_{\text{res2}}$ in the case of an arbitrary downmix enhancement residual, as defined in subclause 6.4.

The 1024 (or 2048 in case two elements have been decoded, see Table 22, Table 34 and Table 100) MDCT coefficients are first transformed to the QMF domain using windowing, overlap-add and inverse MDCT and MDST transforms, resulting in $\tilde{x}^{n,k}_{res0}$ in the case of a TTT box or $\tilde{v}^{n,k}_{res1}$, $\tilde{v}^{n,k}_{res2}$, ... in the case of an OTT box or $\tilde{x}^{n,k}_{res1}$, $\tilde{x}^{n,k}_{res2}$ in the case of an arbitrary downmix enhancement residual. This transformation is dependent on the *window_sequence* corresponding to the ICS or CPE, the spatial frame length (*bsFrameLength*) and the number of residual frames per spatial frame (*bsResidualFramesPerSpatialFrame* for residual coding and *bsArbitraryDownmixResidualFramesPerSpatialFrame* for downmix enhancement residual coding).

The MDCT to QMF transformation is described in subclause 6.9.2. The QMF domain samples are then converted to the hybrid QMF domain as described in subclause 6.9.3. An overview of the residual decoding process is given in Figure 38, and Figure 39 shows an overview of the decoding process for artistic downmix enhancement residuals.

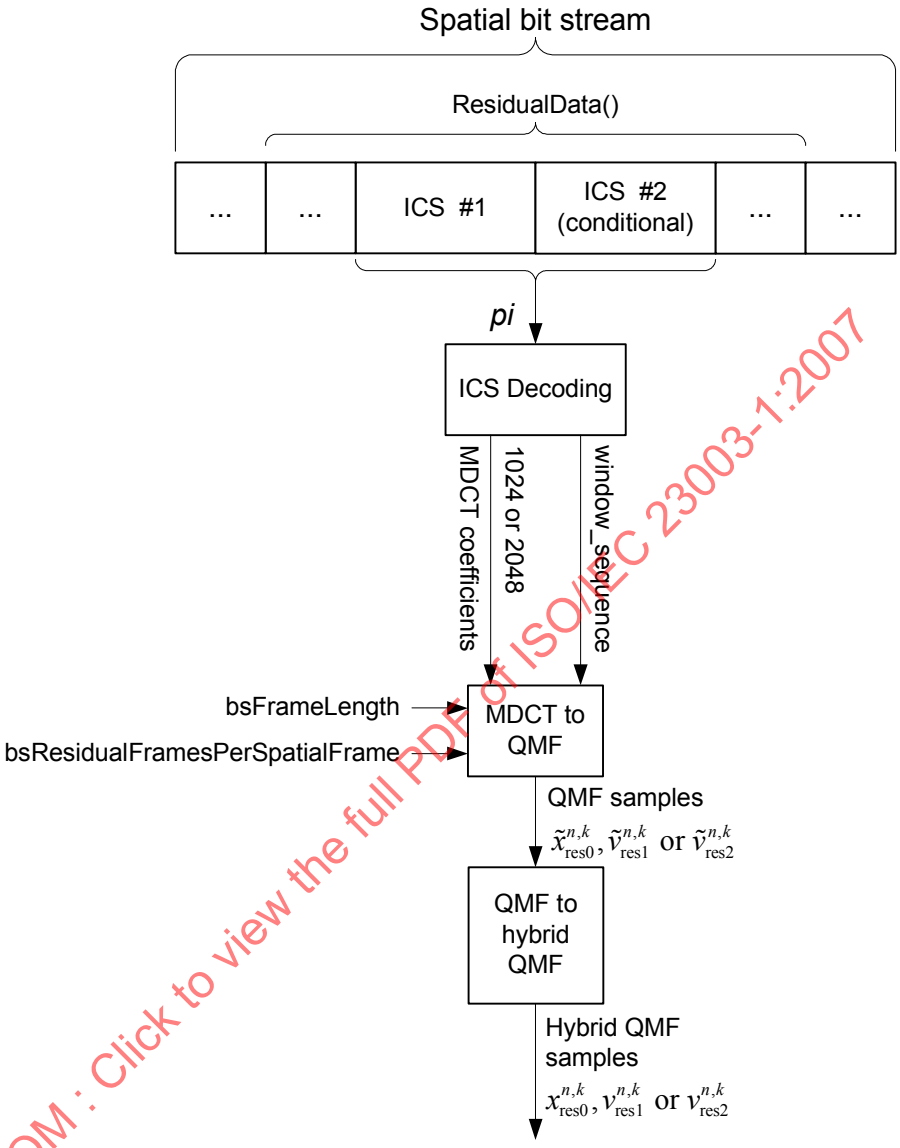

Figure 38 — Overview of residual decoding process.

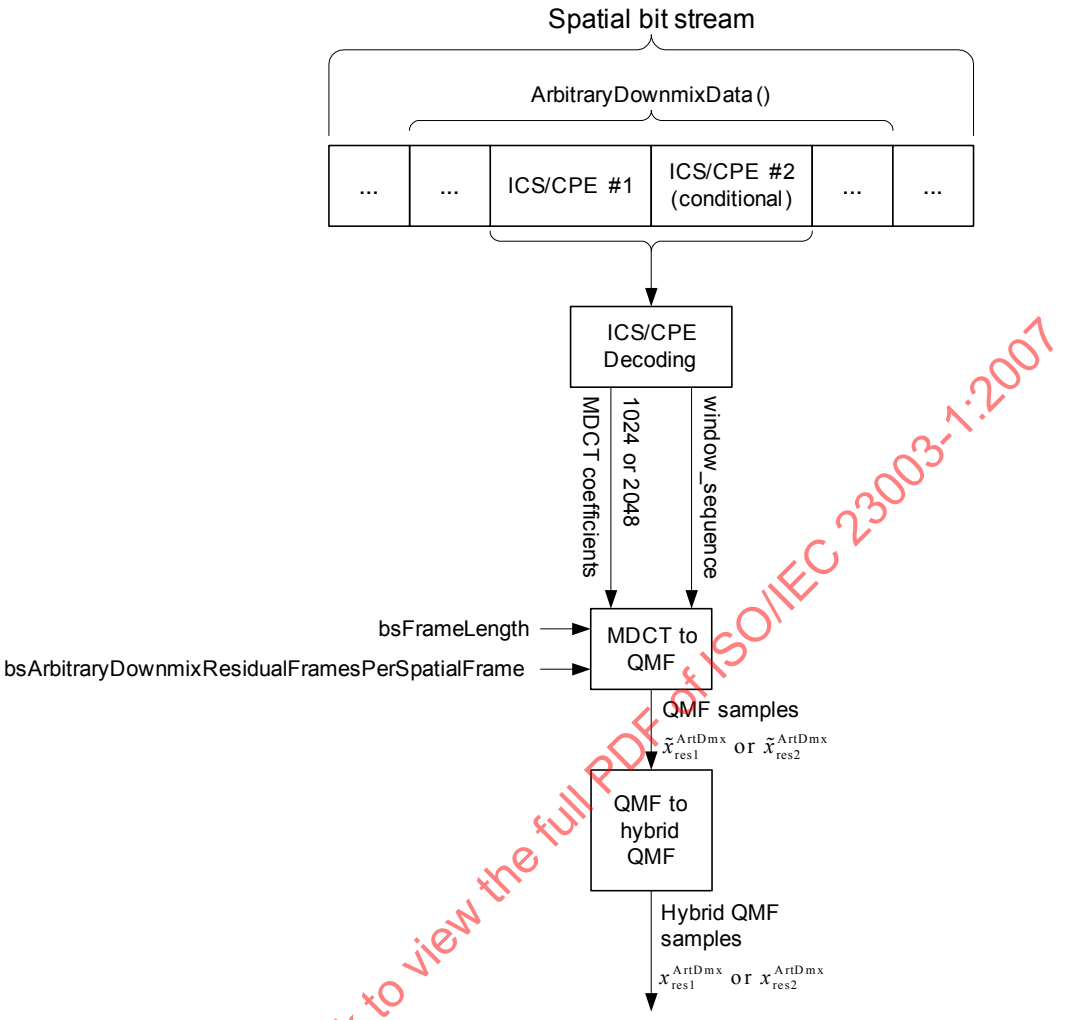

Figure 39 — Overview of artistic enhancement residual decoding process of residual decoding process.

6.9.2 Transforming MDCT coefficients to QMF samples

General definitions required in the description of the transformation are given in subclause 6.9.2.1. The QMF and MDCT windows are defined in subclause 6.9.2.2 and subclause 6.9.2.3 respectively. The MDCT to QMF transformation applies the inverse MDCT and inverse MDST transformations, which are defined in subclause 6.9.2.4. The MDCT to QMF transformation is described in subclause 6.9.2.5.

6.9.2.1 General definitions

The residual QMF frame length, denoted by $N_{\mathrm{qmf,LONG}}$, specifies the number of time slots spanned by the QMF samples after the MDCT to QMF transformation for long windows (*window_sequence* equals ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE or LONG_STOP_SEQUENCE, according to

ISO/IEC 13818-7). The variable $N_{qmf,LONG}$ is derived from the spatial frame length ($bsFrameLength$) and the number of residual frames per spatial frame ($bsResidualFramesPerSpatialFrame$):

$$N_{\text{qmf,LONG}} = \frac{bsFrameLength + 1}{bsResidualFramesPerSpatialFrame + 1}.$$

The allowed values of $N_{\text{qmf,LONG}}$ are derived from the restrictions specified in Table 87.

For short windows (i.e. *window_sequence* equals EIGHT_SHORT_SEQUENCE), the values for the residual QMF frame length, denoted by $N_{\text{qmf,SHORT}}$ are a function of the short window index i_{short} as given in Table 100. For the elements marked with an asterisk two ICS elements are decoded to obtain the proper amount of MDCT coefficients.

Table 100 — Number of short windows N_{short} as a function of residual frame length $N_{\text{qmf,LONG}}$ and the corresponding half window lengths for short windows $N_{\text{qmf,SHORT}}(i_{\text{short}})$ as a function of short window index i_{short}.

$N_{\text{qmf,LONG}}$	N_{short}	i_{short}	$N_{\text{qmf,SHORT}}(i_{\text{short}})$
15	7	0-5	2
		6	3
16	8	0-7	2
18	9*	0-8	2
24	12*	0-11	2
30	15*	0-15	2
32	8	0-7	4

The MDCT coefficients are denoted by $\mathbf{s}$ with length N_{mdct}. The long and short window sequences are defined in subclause 6.9.2.2.

6.9.2.2 Definition of QMF windows

The QMF window $\mathbf{w}_t$ is the sine window, with left and right halves defined according to

$$w_{\text{LEFT},N}[n] = \sin\left(\frac{\pi}{N}\left(n + \frac{1 + oddflag}{2}\right)\right) \quad \text{for } 0 \le n < N/2$$

$$w_{\text{RIGHT},N}[n] = \sin\left(\frac{\pi}{N}\left(n + \frac{1 + oddflag}{2}\right)\right) \quad \text{for } N/2 \le n < N.$$

where

$$oddflag = \begin{cases} 1 & \text{if } N/2 \text{ is odd;} \\ 0 & \text{if } N/2 \text{ is even.} \end{cases}$$

For *window_sequence* equal to ONLY_LONG_SEQUENCE, the QMF window is defined according to

$$w_t[n] = \begin{cases} w_{\mathrm{LEFT},2N_{\mathrm{qmf,LONG}}}[n] & \text{for } 0 \le n < N_{\mathrm{qmf,LONG}} \\ w_{\mathrm{RIGHT},2N_{\mathrm{qmf,LONG}}}[n] & \text{for } N_{\mathrm{qmf,LONG}} \le n < 2N_{\mathrm{qmf,LONG}} \end{cases}.$$

For *window_sequence* equal to LONG_START_SEQUENCE, the QMF window is defined according to

$$w_t[n] = \begin{cases} w_{\mathrm{LEFT},2N_{\mathrm{qmf,LONG}}}[n] & ,for\ 0 \le n < N_{\mathrm{qmf,LONG}} \\ 1.0 & ,for\ N_{\mathrm{qmf,LONG}} \le n < N_{\mathrm{qmf,LONG}} + N_1 \\ w_{\mathrm{RIGHT},2N_{\mathrm{qmf,SHORT}}}\left[n - N_{\mathrm{qmf,LONG}} - N_1 + N_{\mathrm{qmf,SHORT}}\right] & ,for\ N_{\mathrm{qmf,LONG}} + N_1 \le n < N_{\mathrm{qmf,LONG}} + N_1 + N_{\mathrm{qmf,SHORT}} \\ 0.0 & ,for\ N_{\mathrm{qmf,LONG}} + N_1 + N_{\mathrm{qmf,SHORT}} \le n < 2N_{\mathrm{qmf,LONG}} \end{cases}$$

where

$$N_1 = \begin{cases} N_{\mathrm{qmf,LONG}}/2 - 2 & \text{if } N_{\mathrm{qmf,LONG}} = 32 \\ \left\lfloor N_{\mathrm{qmf,LONG}}/2 \right\rfloor - 1 & \text{else.} \end{cases}.$$

For *window_sequence* equal to EIGHT_SHORT_SEQUENCE, the QMF window is defined according to

$$w_{t,i_{\mathrm{short}}}[n] = \begin{cases} w_{\mathrm{LEFT},2N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}))}[n] & \text{for} \quad 0 \le n < N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) \\ w_{\mathrm{RIGHT},2N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}})}[n] & \text{for} \quad N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) \le n < 2N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) \end{cases}$$

in case $N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}})$ is even, and

$$w_{t,i_{\mathrm{short}}}[n] = \begin{cases} w_{\mathrm{LEFT},2N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}})-2}[n] & \text{for} \quad 0 \le n < N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) - 1 \\ 1 & \text{for} \quad n = N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) - 1 \\ w_{\mathrm{RIGHT},2N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}})-2}[n-1] & \text{for} \quad N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) \le n < 2N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) - 1 \\ 0 & \text{for} \quad n = 2N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}}) - 1 \end{cases}$$

in case $N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}})$ is odd.

For *window_sequence* equal to LONG_STOP_SEQUENCE, the QMF window is defined according to

$$w_t[n] = \begin{cases} 0.0 & ,for\ 0 \le n < N_1 \\ w_{\mathrm{LEFT},2N_{\mathrm{qmf,SHORT}}}[n - N_1] & ,for\ N_1 \le n < N_1 + N_{\mathrm{qmf,SHORT}} \\ 1.0 & ,for\ N_1 + N_{\mathrm{qmf,SHORT}} \le n < N_{\mathrm{qmf,LONG}} \\ w_{\mathrm{RIGHT},2N_{\mathrm{qmf,LONG}}}[n] & ,for\ N_{\mathrm{qmf,LONG}} \le n < 2N_{\mathrm{qmf,LONG}} \end{cases}$$

Figure 40 illustrates a stylistic view of the QMF windows for the case where $N_{qmf,LONG}$ is even

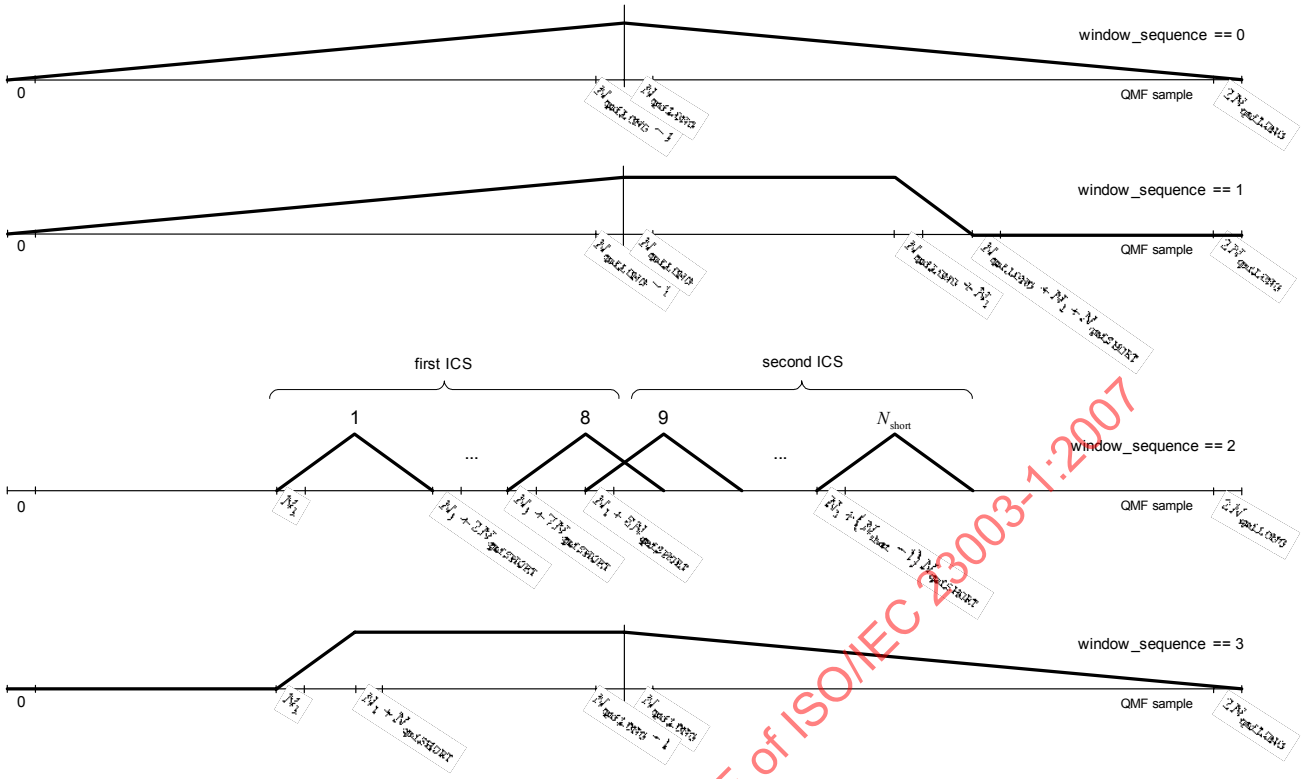

Figure 40 — Stylistic view of the QMF windows

6.9.2.3 Definition of MDCT windows

The MDCT window is denoted by $\mathbf{w}_f$. Let the vector $\mathbf{h}_{qmf} \in R^{640}$ contain the QMF filter prototype coefficients, as defined in ISO/IEC 14496-3, subclause 4.A.6.2, Table 4.A.87. The vector $\mathbf{h} \in R^{640}$ is derived from $\mathbf{h}_{qmf}$ as follows:

$$h[n] = \begin{cases} h_{qmf}[n] & \text{if } 0 \le n \le 127 \\ -h_{qmf}[n] & \text{if } 128 \le n \le 255 \\ h_{qmf}[n] & \text{if } 256 \le n \le 383 \\ -h_{qmf}[n] & \text{if } 384 \le n \le 511 \\ h_{qmf}[n] & \text{if } 512 \le n \le 639. \end{cases}$$

The vector $\mathbf{h}$ is normalized, resulting in $\mathbf{h}_{norm}$:

$$h_{norm}[n] = \frac{h[n]}{\sum_{n=0}^{639} h[n]}.$$

Let

$$N_{\text{qmf}} = \begin{cases} N_{\text{qmf,LONG}} & \text{for long windows} \\ N_{\text{qmf,SHORT}}\left(i_{\text{short}}\right) & \text{for short windows} \end{cases}$$

and let

$$oddflag = \begin{cases} 1 & \text{if } N_{\text{qmf}} \text{ is odd;} \\ 0 & \text{if } N_{\text{qmf}} \text{ is even.} \end{cases}$$

The MDCT window is then given by

$$w_{\text{f}}[n] = \sum_{m=-319}^{319} h_{\text{norm}}[320+m]\cos\left(\frac{\pi\left(2n+1+oddflag-2N_{\text{qmf}}\right)m}{128N_{\text{qmf}}}\right)$$

for $n = 0,1,\ldots,2N_{\text{qmf}}-1-oddflag$. If $oddflag = 1$, then

$$w_{\text{f}}\left[2N_{\text{qmf}}-1\right] = 0.$$

6.9.2.4 Definition of inverse MDCT and inverse MDST transformations

The analytical expression for the inverse MDCT f_{IMDCT} of a vector $\mathbf{x}_{\text{mdct}} \in R^N$ under the window $\mathbf{w} \in R^N$ is as follows

$$\begin{aligned} x_{\text{time}}[n] &= \left(f_{\text{IMDCT}}\left(\mathbf{x}_{\text{mdct}}, \mathbf{w}\right)\right)[n] \\ &= \sqrt{\frac{2}{N}} w[n] \sum_{k=0}^{N-1} \mathbf{x}_{\text{mdct}}[k] \cos\left(\frac{\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right) \quad \text{for} \quad 0 \le n < 2N. \end{aligned}$$

The analytical expression for the inverse MDST f_{IMDST} of a vector $\mathbf{x}_{\text{mdst}} \in R^N$ under the window $\mathbf{w} \in R^N$ is as follows

$$\begin{aligned} x_{\text{time}}[n] &= \left(f_{\text{IMDST}}\left(\mathbf{x}_{\text{mdst}}, \mathbf{w}\right)\right)[n] \\ &= \sqrt{\frac{2}{N}} w[n] \sum_{k=0}^{N-1} x_{\text{mdst}}[k] \sin\left(\frac{\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right) \quad \text{for} \quad 0 \le n < 2N. \end{aligned}$$

For both the inverse MDCT as well as the inverse MDST, $n_0 = 1/2 - \lfloor N/2 \rfloor$.

6.9.2.5 Description of MDCT to QMF transformation

6.9.2.5.1 Introduction

The MDCT coefficients for long windows are mapped to the QMF domain and are contained in the matrix $\mathbf{Z}_0$, having $2N_{\text{qmf,LONG}}$ rows (representing the time slots) and L_{qmf} columns (representing the QMF sub bands).

For short windows, each of the N_{short} sets of 128 MDCT coefficients are mapped to a QMF domain matrix $\mathbf{Z}_{i_{\text{short}}}$ where i_{short} is the short window index, having $2N_{\text{qmf,SHORT}}(i_{\text{short}})$ rows and L_{qmf} columns. The number of QMF sub bands is given by:

$$L_{\text{qmf}} = \begin{cases} \max\left(64, ceil\left(\dfrac{1024}{N_{\text{qmf,LONG}}}\right)\right) & \text{for long windows} \\ \max\left(64, ceil\left(\dfrac{128}{N_{\text{qmf,SHORT}}}\right)\right) & \text{for short windows.} \end{cases}$$

After L_{qmf} is determined, for each window (one window in case of a long window) the vector $\mathbf{s}$ containing the MDCT coefficients is padded with zeros, or truncated, until the length equals

$$N_{\text{mdct}} = \begin{cases} L_{\text{qmf}} \cdot N_{\text{qmf,LONG}} + \lfloor N_{\text{qmf,LONG}} / 2 \rfloor & \text{for long windows, if } L_{\text{qmf}} < 64 \\ L_{\text{qmf}} \cdot N_{\text{qmf,SHORT}} + \lfloor N_{\text{qmf,SHORT}} / 2 \rfloor & \text{for short windows, if } L_{\text{qmf}} < 64 \\ 64 \cdot N_{\text{qmf,LONG}} & \text{for long windows, if } L_{\text{qmf}} = 64 \\ 64 \cdot N_{\text{qmf,SHORT}} & \text{for short windows, if } L_{\text{qmf}} = 64 \end{cases}.$$

In case of $N_{\text{qmf,SHORT}}(i_{\text{short}}) = 3$, which occurs only for $N_{\text{qmf,LONG}} = 15$ and $i_{\text{short}} = 6$, the MDCT coefficients required are merged from the two last AAC short blocks distributed in the manner depicted in Figure 41.

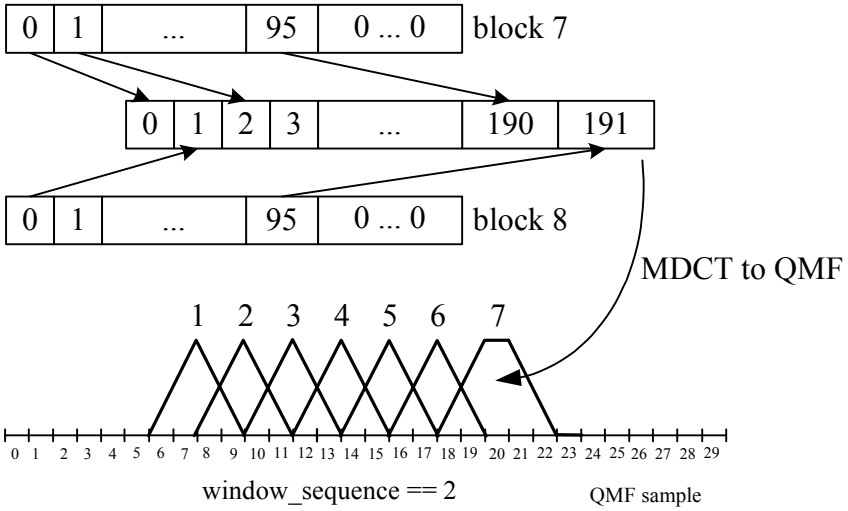

Figure 41 — Merging the MDCT coefficients from two last AAC short blocks into one set of 192 MDCT coefficients for window 7.

The MDCT to QMF mapping is described in subclause 6.9.2.5.2. Updating the QMF samples by overlap-add is described in subclause 6.9.2.5.3.

6.9.2.5.2 The MDCT to QMF mapping

A flowchart of the MDCT to QMF transformation is given in Figure 42.

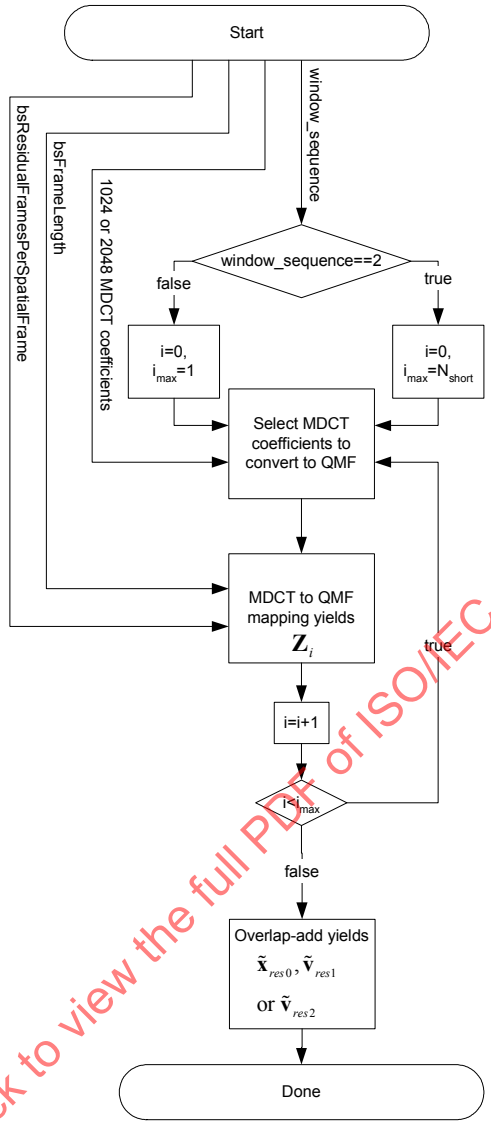

Figure 42 — Flowchart of MDCT to QMF transformation

The MDCT to QMF mapping is described for windows with residual QMF frame length N_{qmf}. For long and short windows, N_{qmf} is to be replaced by $N_{\mathrm{qmf,LONG}}$ and $N_{\mathrm{qmf,SHORT}}(i_{\mathrm{short}})$ respectively in the description below. In case of short windows, the procedure outlined below is performed for $i_{\mathrm{short}} = 0 \ldots N_{\mathrm{short}} - 1$. Otherwise, this procedure is performed only once. Furthermore, an integer valued split $N_{\mathrm{qmf}} = N_{\mathrm{w}} + N_{\mathrm{a}}$ is formed by putting $N_{\mathrm{w}} = \lfloor N_{\mathrm{qmf}} / 2 \rfloor$ and $N_{\mathrm{a}} = \lceil N_{\mathrm{qmf}} / 2 \rceil$.

The entries of the MDCT window w_f are distributed into two butterfly coefficient sequences β and α of length N_{qmf},

$$\beta[r] = w_{\mathrm{f}}[r + N_{\mathrm{w}}], \quad r = 0, \ldots, N_{\mathrm{qmf}} - 1,$$

$$\alpha[r] = \begin{cases} w_{\mathrm{f}}[r + N_{\mathrm{w}} + N_{\mathrm{qmf}}], & r = 0,\ldots,N_{\mathrm{a}} - 1; \\ w_{\mathrm{f}}[r - N_{\mathrm{a}}], & r = N_{\mathrm{a}},\ldots,N_{\mathrm{qmf}} - 1. \end{cases}$$

The MDCT coefficients in $\mathbf{s}$ are mapped into blocks according to

$$s_k[r] = \sigma_k s[kN_{\mathrm{qmf}} + r], \quad ,k = 0,\ldots,L_{\mathrm{qmf}} - 1, \quad ,r = 0,\ldots,N_{\mathrm{qmf}} - 1,$$

where $\sigma_k = \mathrm{sign}\left[\cos\left(\frac{\pi(k+1/2)}{2}\right)\right]$ and additionally, if $L_{qmf} < 64$,

$$s_{L_{\mathrm{qmf}}}[r] = \sigma_{L_{\mathrm{qmf}}} s\left[L_{\mathrm{qmf}} N_{\mathrm{qmf}} + r\right]$$

for $0 \le r < N_w$. These MDCT blocks are windowed to form L_{qmf} pass band signals b_k

$$b_k[r] = \beta[r] s_k[r],$$

for $0 \le k < L_{\mathrm{qmf}}$, $0 \le r < N_{\mathrm{qmf}}$. Similarly, L_{qmf} stop band signals a_k are computed from

$$a_k[r] = \begin{cases} \alpha[r] s_0[N_{\mathrm{qmf}} - 1 - r] & ,N_{\mathrm{w}} \le r < N_{\mathrm{qmf}}, \quad k = 0 \\ \alpha[r] s_{k-1}[r] & ,N_{\mathrm{w}} \le r < N_{\mathrm{qmf}}, \quad 1 \le k < L_{\mathrm{qmf}} \\ \alpha[r] s_{k+1}[r] & ,0 \le r < N_{\mathrm{w}}, \quad 0 \le k < L_{\mathrm{qmf}} - 1 \\ \alpha[r] s_{L_{\mathrm{qmf}}}[r] & ,0 \le r < N_{\mathrm{w}}, \quad k = L_{\mathrm{qmf}} - 1, \quad L_{\mathrm{qmf}} < 64 \\ \alpha[r] s_{L_{\mathrm{qmf}}-1}[N_{\mathrm{qmf}} - 1 - r], & ,0 \le r < N_{\mathrm{w}}, \quad k = L_{\mathrm{qmf}} - 1, \quad L_{\mathrm{qmf}} = 64 \end{cases}$$

The pass band and stop band signals are combined into MDCT and MDST data vectors ξ_k and $\boldsymbol{\eta}_k$ as follows for $k = 0,\ldots,L_{\mathrm{qmf}} - 1, \quad r = 0,\ldots,N_{\mathrm{qmf}} - 1$,

$$v_k^-[r] = \begin{cases} a_k[N_{\mathrm{qmf}} - 1 - r], & k \text{ even} \\ b_k[N_{\mathrm{qmf}} - 1 - r], & k \text{ odd} \end{cases}$$

$$v_k^+[r] = \begin{cases} b_k[r], & k \text{ even} \\ a_k[r], & k \text{ odd} \end{cases}$$

$$\xi_k[r] = \left(v_k^+[r] + v_k^-[r]\right)/2$$

$$\eta_k[r] = \left(v_k^+[r] - v_k^-[r]\right)/2$$

Next, the IMDCT transformation is applied to the vectors ξ_k, and the IMDST transformation is applied to the vectors η_k under the window $\mathbf{w}_t$. The result is phase-adjusted and placed in the QMF matrix $\mathbf{Z}$:

$$\mathbf{x}_k = f_{\text{IMDCT}}(\xi_k, \mathbf{w}_t) \in R^{2N_{\text{qmf}}}$$
$$\mathbf{y}_k = f_{\text{IMDST}}(\eta_k, \mathbf{w}_t) \in R^{2N_{\text{qmf}}}$$
$$\mathbf{z}_k = \mathbf{x}_k + j\mathbf{y}_k, \quad k = 0,\ldots,L_{\text{qmf}} - 1 \quad .$$
$$\mathbf{Z} = \left[\exp\left(-\frac{j\pi}{256}(258\cdot 0 + 385)\right)\mathbf{z}_0^T \quad \exp\left(-\frac{j\pi}{256}(258\cdot 1 + 385)\right)\mathbf{z}_1^T \quad \cdots \right.$$
$$\left. \exp\left(-\frac{j\pi}{256}(258\cdot(L_{\text{qmf}} - 1) + 385)\right)\mathbf{z}_{L_{\text{qmf}}-1}^T \right] \in C^{2N_{\text{qmf}}, L_{\text{qmf}}}$$

6.9.2.5.3 Updating the QMF output

Construction of the residual frame output QMF samples $\tilde{\mathbf{x}}_{\text{res0}}$, $\tilde{\mathbf{v}}_{\text{res1}}$ or $\tilde{\mathbf{v}}_{\text{res2}}$ takes place using a couple of stages, first of all the individual windows $\mathbf{Z}_i$ of the current residual frame are overlap-added in a buffer $\mathbf{Z}_{\text{residualframe}}$ consisting of L_{qmf} columns and $2N_{\text{qmf,LONG}}$ rows. Then, for all residual frames within the current frame overlap-add is applied, resulting in a buffer $\mathbf{Z}_{\text{spatialframe}}$ consisting of L_{qmf} columns and $(N_{\text{RFSF}} + 1)N_{\text{qmf,LONG}}$ rows. Finally this buffer is overlap-added with the previous buffer $\mathbf{Z}_{\text{spatialframe,prev}}$.

In case the window sequence for the current residual frame indicates a long window, the buffer $\mathbf{Z}_{\text{residualframe}}$ is constructed as:

$$Z^{n,m}_{\text{residualframe}} = Z^{n,m}_0 \quad \textit{for } 0 \le m < L_{\text{qmf}}, 0 \le n < N_{\text{qmf,LONG}} \ .$$

In case the window sequence for the current residual frame indicates short windows, the buffer $\mathbf{Z}_{residualframe}$ is constructed using the following recursion. First a zero buffer $\mathbf{Z}_{tmp,0}$ is defined. Then for $i_{short} = 0 \ldots N_{short} - 1$ this buffer is updated as:

$$Z^{n,m}_{\text{tmp},i_{\text{short}}+1} = Z^{n,m}_{\text{tmp},i_{\text{short}}} + Z^{n-n_{\text{start}}(i_{\text{short}}),m}_{i_{\text{short}}} \quad ,$$

for $0 \le m < L_{\text{qmf}}$, $n_{\text{start}}(i_{\text{short}}) \le n < n_{\text{start}}(i_{\text{short}}) + 2N_{\text{qmf,SHORT}}(i_{\text{short}})$

where the short window start position for short window i_{short} is given by

$$n_{\text{start}}(i_{\text{short}}) = n_{\text{offset}}(N_{\text{qmf,LONG}}) + i_{\text{short}} \cdot N_{\text{qmf,SHORT}}(0)$$

and the start-position of the first window $n_{\text{offset}}(N_{\text{qmf,LONG}})$ is given in Table 101.

Table 101 — Short window offset position $n_{offset}(N_{qmf,LONG})$

$N_{\text{qmf,LONG}}$	$n_{\text{offset}}\left(N_{\text{qmf,LONG}}\right)$
15	6
16	7
18	8
24	11
30	14
32	14

After the above procedure is completed, the temporary buffer is copied to the current residual frame buffer as $\mathbf{Z}_{\text{residualframe}} = \mathbf{Z}_{\text{tmp},N_{\text{short}}}$.

In order to construct the $\mathbf{Z}_{spatialframe}$ buffer a temporary zero buffer $\mathbf{Z}_{\text{tmp},0}$ consisting of L_{qmf} columns and $\left(N_{\text{RFSF}}+1\right)N_{\text{qmf,LONG}}$ rows is initialized. This buffer is updated for all residual frames in the current frame, i.e., $rf = 0...N_{\text{RFSF}}-1$ as:

$$Z^{n+rf\cdot N_{\text{qmf,LONG}},m}_{\text{tmp},rf+1} = Z^{n+rf\cdot N_{\text{qmf,LONG}},m}_{\text{tmp},rf} + Z^{n,m}_{\text{residualframe},rf} \quad .$$

for $0 \le m < L_{\text{qmf}}$, $0 \le n < 2N_{\text{qmf,LONG}}$.

After this procedure is completed, this temporary buffer is copied to the buffer for the current spatial frame as $\mathbf{Z}_{\text{spatialframe}} = \mathbf{Z}_{\text{tmp},N_{\text{RFSF}}}$.

In the final overlap-add stage the output buffer $\mathbf{Z}_{out}$ is constructed by overlap-add of the previous spatial frame denoted by $\mathbf{Z}_{spatialframe,prev}$ with the current spatial frame $\mathbf{Z}_{spatialframe}$ as:

$$Z^{n,m}_{\text{out}} = Z^{n+N_{\text{qmf,LONG}},m}_{\text{spatialframe,prev}} + Z^{n,m}_{\text{spatialframe}} \quad .$$

for $0 \le m < L_{\text{qmf}}$, $0 \le n < N_{\text{qmf,LONG}}$

After all residual frames in a spatial frame are processed, the elements of $\mathbf{Z}_{\text{out}}$ are copied to $\tilde{\mathbf{x}}_{\text{res0}}$, $\tilde{\mathbf{v}}_{\text{res1}}$ or $\tilde{\mathbf{v}}_{\text{res2}}$, depending on which parameter instance pi is being processed.

6.9.3 Transforming the QMF samples to hybrid QMF samples

The first $N_{\text{RFSF}}N_{\text{qmf,LONG}}$ QMF time slots in the residual QMF matrix ($\tilde{\mathbf{x}}_{\text{res}}$ for TTT residuals and $\tilde{\mathbf{v}}_{\text{res1}}$ or $\tilde{\mathbf{v}}_{\text{res2}}$ for OTT residuals) are transformed to the hybrid QMF domain through the transformation described in 6.3.2, resulting in the hybrid QMF buffers $\mathbf{x}_{res}$ (for TTT residuals) and $\mathbf{v}_{\text{res1}}$ or $\mathbf{v}_{\text{res2}}$ (for OTT residuals), described in subclause 6.4. Note that the hybrid QMF filter states, corresponding to each residual, are stored for the next frame.

6.10 Low Power MPEG Surround decoding

6.10.1 Introduction

The following subclauses outline the differences for the implementation of the Low Power version of the MPEG Surround decoder compared to the High Quality version outlined in previous chapters. The Low Power MPEG Surround decoder operates on real-valued subband domain signals above the K_c-th QMF subband, corresponding to the M_{proc}^{c}-th processing band. A real-valued QMF filterbank is used in combination with a real to complex converter to achieve this partially complex subband domain representation. Furthermore, the Low Power MPEG Surround decoder incorporates additional modules in order to reduce aliasing introduced due to the real-valued processing, and replaces some existing modules in order to minimize computational complexity.

6.10.2 Time / frequency transforms

6.10.2.1 Introduction

The Low Power MPEG Surround coding system employs time/frequency transforms according to Figure 43 and Figure 44.

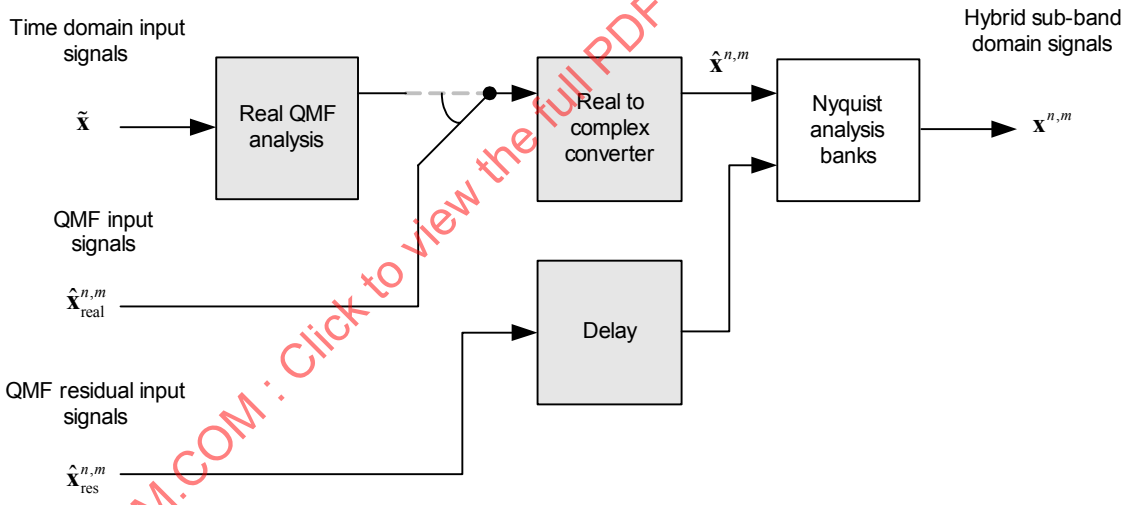

Figure 43 — Time/frequency transforms for Low Power MPEG Surround, hybrid QMF analysis bank

Apart from the regular mode of operation where the MPEG Surround decoder is fed with time-domain samples $\tilde{\mathbf{x}}$, also intermediate real valued (QMF) subband domain samples $\hat{\mathbf{x}}_{\text{real}}^{n,m}$ from a Low Power HE-AAC decoder can be taken. In that case the subband domain samples prior to HE-AAC QMF synthesis are taken [ISO/IEC 14496-3]. The real QMF samples are converted to partially complex samples $\hat{\mathbf{x}}^{n,m}$ by the real to complex converter described in subclause 6.10.2.3. Furthermore, if enabled, the residual decoding module provides subband domain samples $\hat{\mathbf{x}}_{\text{res}}^{n,m}$ that also need to be delayed in order to compensate for the delay of the real to complex converter and transformed to the hybrid domain ($\mathbf{x}^{n,m}$).

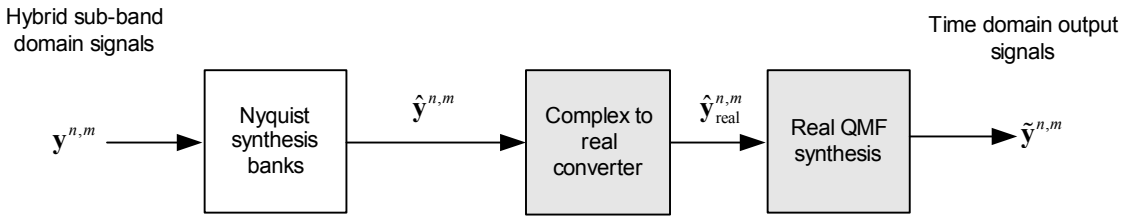

Figure 44 — Time/frequency transforms for Low Power MPEG Surround, hybrid QMF synthesis bank

At the synthesis side the hybrid subband domain samples $\mathbf{y}^{n,m}$ are transformed to partially complex QMF subband domain samples $\hat{\mathbf{y}}^{n,m}$ which are converted to real QMF samples $\hat{\mathbf{y}}^{n,m}_{\text{real}}$ by the complex to real converter described in subclause 6.10.2.4. Those real QMF samples are transformed back to the time domain samples $\tilde{\mathbf{y}}$ by the real QMF synthesis bank.

For Low Power MPEG Surround, real-valued QMF filterbanks are used. The analysis filterbank uses 64 channels and is outlined below. The synthesis filterbank also has 64 channels and is identical to the filterbank used in Low Power HE-AAC (subclause 4.6.18.8.2.3 of ISO/IEC 14496-3).

6.10.2.2 Real-valued analysis QMF bank

The real-valued QMF bank is used to split the time domain signal output from the core decoder into 64 subband signals. The output from the filterbank, i.e. the subband samples, are real-valued and critically sampled. The flowchart of the operation is given in Figure 45. The filtering involves the following steps, where an array $\mathbf{x}$ consisting of 640 time domain input samples is assumed. A higher index into the array corresponds to older samples.

- Shift the samples in the array $\mathbf{x}$ by 64 positions. The oldest 64 samples are discarded and 64 new samples are stored in positions 0 to 63.
- Multiply the samples of array $\mathbf{x}$ by the coefficients of window $\mathbf{c}$. The window coefficients can be found in Table 4.A.87 of ISO/IEC 14496-3.
- Sum the samples according to the formula in the flowchart to create the 128-element array $\mathbf{u}$.
- Calculate new 64 subband samples by the matrix operation $\mathbf{Mu}$, where

$$\mathbf{M}_r(k,n) = 2 \cdot \cos\left(\frac{\pi \cdot (k+0.5) \cdot (2 \cdot n - 192)}{128}\right), \begin{cases} 0 \le k < 64 \\ 0 \le n < 128 \end{cases}$$

Every loop in the flowchart produces 64 subband samples, each representing the output from one filterbank subband. In the flowchart $\mathbf{X}_{\text{real}}[k][l]$ corresponds to subband sample l of QMF subband k. Hence $\hat{x}^{n,m}_{\text{real},k} = \mathbf{X}_{\text{real}}[m][n]$.

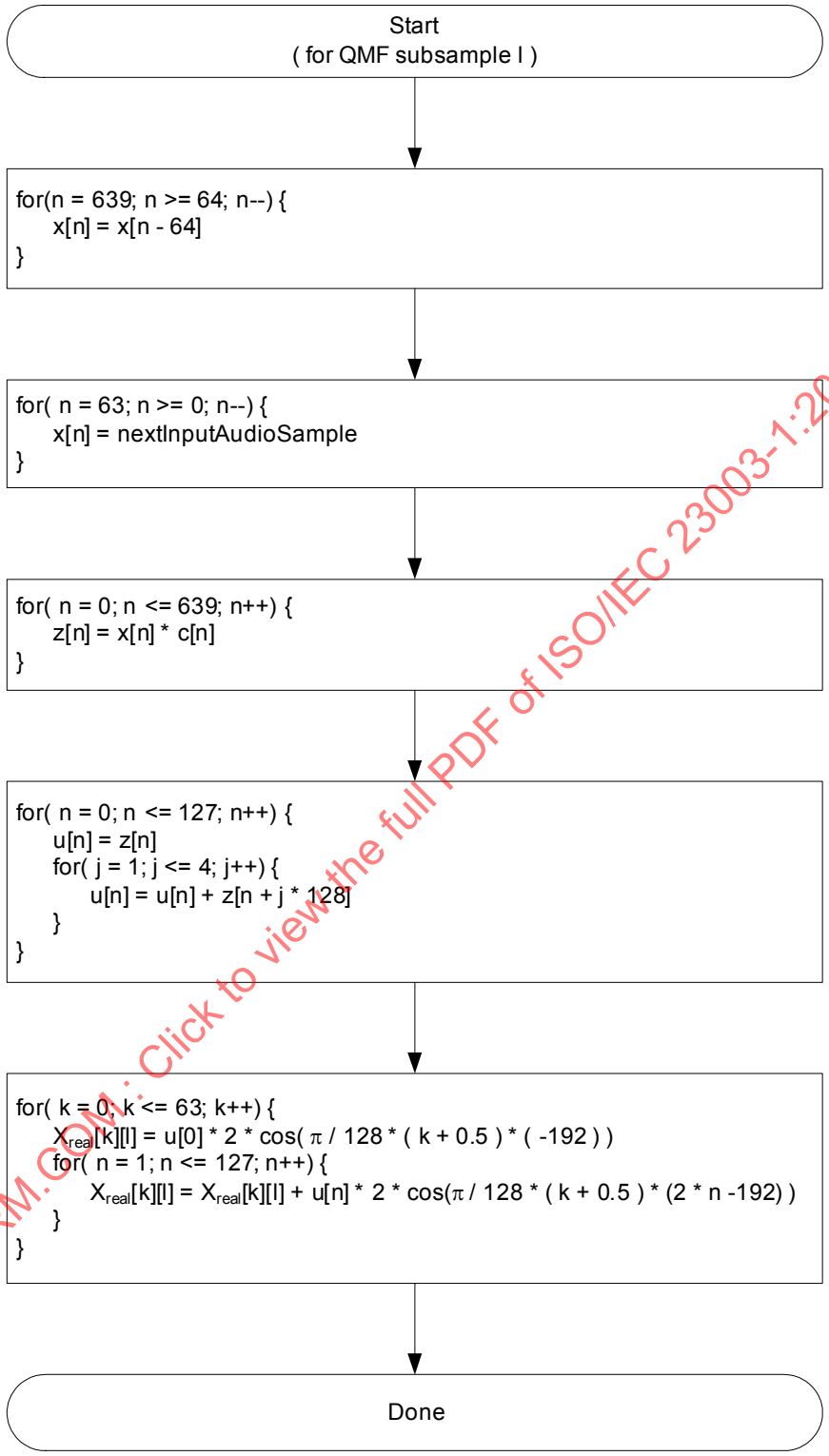

Figure 45 — Flowchart of real-valued analysis QMF bank

6.10.2.3 Real to complex converter

The real QMF subband signals are transformed into partially complex QMF subbands according to Figure 46.

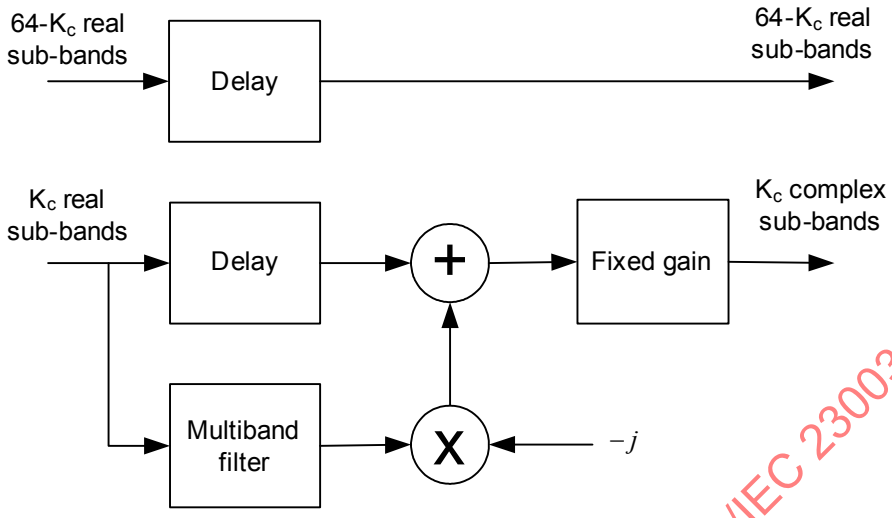

Figure 46 — Real to complex converter

The first group of K_c real subband signals is filtered by a multiband filter, multiplied by the negative of the imaginary unit and added to the K_c delayed real subband signals in order to produce K_c complex subband signals. Those subband signals are gain adjusted by a fixed real gain and output as the K_c complex subbands of the partially complex analysis. The second group consisting of $64 - K_c$ real subband signals are just delayed. The role of both delays is to compensate for the delay introduced by the multiband filter. The length of this delay is half of the order of the multiband prototype filters $a^{\nu}[n]$ given in Table 102. This amounts to 5 subband samples.

The multiband filter operates on the K_c first QMF subband signals in the following way,

$$\hat{x}^{n,m}_{\mathrm{imag},k} = \sum_{r=-q(m)}^{p(m)} \sum_{\nu=0}^{10} f_{m,r}[\nu]\hat{x}^{n-\nu,m+r}_{\mathrm{real},k}, \quad m = 0,1,\ldots,K_c - 1,$$

where the QMF subband summation limits are defined by

$$q(m) = \begin{Bmatrix} 0, & for\ m = 0 \\ 1, & for\ m = 1,\ldots,K_c - 1 \end{Bmatrix}, \text{ and } \quad p(m) = \begin{Bmatrix} 1, & for\ m = 0,\ldots,K_c - 2 \\ 0, & for\ m = K_c - 1 \end{Bmatrix}.$$

The filters $f_{m,r}$ are derived from them prototype filters of Table 1 via

$$f_{m,r}[\nu] = \begin{cases} \sin\left[\frac{\pi}{2}\left[-(2m+1)(\nu-5)\right]\right] a^0[\nu] + (-1)^m a^1[\nu], & if \quad (m,r) \in \{(0,0),(K_c-1,0)\}. \\ \sin\left[\frac{\pi}{2}\left[-r-(2m+1+r)(\nu-5)\right]\right] a^{|r|}[\nu], & else. \end{cases}$$

Table 102 — Multiband filter prototypes $a^{v}[n]$

n	$a^{0}[n]$	$a^{1}[n]$
0	0.00375672984184	0.00087709635503
1	0	0.00968961250934
2	-0.07159908629242	0.04670597747406
3	0	0.12080166385305
4	0.56743883685217	0.20257613284430
5	0	0.23887175675672
6	0.56743883685217	0.20257613284430
7	0	0.12080166385305
8	-0.07159908629242	0.04670597747406
9	0	0.00968961250934
10	0.00375672984184	0.00087709635503

The output of the multiband filter is combined with the delayed real valued QMF subband samples according to Figure 46 to form the partially complex QMF subband samples

$$\hat{x}_k^{n,m} = \begin{cases} \frac{1}{\sqrt{2}}\left(\hat{x}_{\mathrm{real},k}^{n-5,m} - j\,\hat{x}_{\mathrm{imag},k}^{n,m}\right), & m = 0,1,\ldots,K_{\mathrm{c}}-1; \\ \hat{x}_{\mathrm{real},k}^{n-5,m}, & m = K_{\mathrm{c}},\ldots,63. \end{cases}$$

6.10.2.4 Complex to real converter

Prior to the real QMF synthesis, the partially complex subband QMF signals are transformed into real QMF signals according to Figure 47.

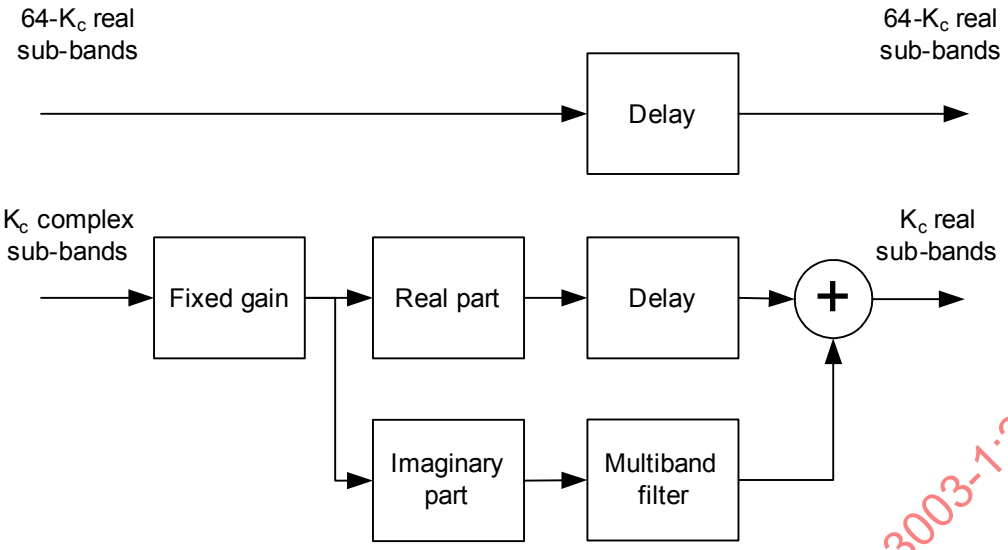

Figure 47 — Complex to real converter

The complex part of the partially complex QMF subband signals $\hat{y}_k^{n,m}$ are gained and split into real and imaginary parts according to

$$\hat{u}_k^{n,m} + j\hat{v}_k^{n,m} = \frac{1}{\sqrt{2}}\hat{y}_k^{n,m}, \quad m = 0,1,\ldots,K_c - 1.$$

The multiband filter operates on the imaginary parts as follows,

$$\hat{w}_k^{n,m} = \sum_{r=-q(m)}^{p(m)} \sum_{\nu=0}^{10} g_{m,r}[\nu]\hat{v}_k^{n-\nu,m+r}, \quad m = 0,1,\ldots,K_c - 1,$$

Where $p(m)$ and $q(m)$ are defined in the previous subclause and the filters $g_{m,r}$ are derived from prototype filters of Table 1 via

$$g_{m,r}[\nu] = \begin{cases} \sin\left[\frac{\pi}{2}\left[-(2m+1)(\nu-5)\right]\right] a^0[\nu] - (-1)^m a^1[\nu] & \text{,if } (m,r) \in \{(0,0),(K_c-1,0)\}. \\ \sin\left[\frac{\pi}{2}\left[-r-(2m+1+r)(\nu-5)\right]\right] a^{|r|}[\nu] & \text{,otherwise} \end{cases}$$

The QMF signals $\hat{y}_{\text{real},k}^{n,m}$ to be fed into the real QMF synthesis are then obtained by

$$\hat{y}_{\text{real},k}^{n,m} = \begin{cases} \hat{u}_k^{n-5,m} + \hat{w}_k^{n,m}, & m = 0,1,\ldots,K_c - 1; \\ \hat{y}_k^{n-5,m}, & m = K_c,\ldots,63. \end{cases}$$

6.10.2.5 Support for higher and lower sampling frequencies

Support for low sampling frequencies, that is if bsSamplingFrequency < 27713, and for high sampling frequencies, that is if bsSamplingFrequency >= 55426, is provided by using downsampled 32 band QMF banks or upsampled 128 band QMF banks, respectively, instead of the 64 band QMF banks that are used for normal sampling frequencies like 32, 44.1, or 48 kHz.

For downsampled operation, the 32 band QMF analysis bank is defined in 14496-3 subclause 4.6.18.8.2.1. The 32 band QMF synthesis bank is defined in 14496-3 subclause 4.6.18.8.2.3. The upper 32 QMF bands are set to zero prior to MPEG Surround processing and only the lower 32 QMF bands are processed by the QMF synthesis bank.

For upsampled operation, the 128 band QMF analysis bank is defined by replacing the modulation of the 64 band QMF analysis bank with

$$\mathbf{M}_r(k,n) = 2 \cdot \cos\left(\frac{\pi \cdot (k+0.5) \cdot (2 \cdot n - 384)}{256}\right), \begin{cases} 0 \le k < 128 \\ 0 \le n < 256 \end{cases}$$

using a 1280 sample version of the window function c[i] where the additional intermediate samples are obtained by linear interpolation of neighboring samples of the original 640 sample window function specified in 14496-3 subclause 4.A.6.2Table 4.A.87. The 128 band QMF synthesis bank is defined by replacing the modulation of the 64 band QMF synthesis bank with

$$\frac{1}{32}\cos\left(\frac{pi}{256}(k+0.5)(2n-128)\right), \quad 0 \le k < 128, 0 \le n < 256$$

using a 1280 sample version of the window function c[i] as defined above. The upper border of the highest processing band is moved from the 64th QMF band to the 128th QMF band.

6.10.3 Aliasing reduction

6.10.3.1 Introduction

The purpose of the aliasing reduction is to suppress aliasing emerging in the borders between parameter bands. Given the critically sampled real-valued QMF filterbank used, subband channels in the filterbank overlap, and aliasing will be introduced if adjacent subbands are modified independently. This will occur for QMF subbands on the parameter borders. An aliasing condition is said to exist when tonal signals are detected in the QMF subbands on the parameter borders. When an aliasing condition is signaled, the parameters for the neighbouring hybrid bands (belonging to different parameter bands) are modified dependent on each other by setting both to the average of the data for the two parameter bands, thus avoiding independent adjustment of the neighbouring QMF subbands, and hence avoiding the introduction of aliasing.

6.10.3.2 Aliasing detection

The aliasing detector operates on real-valued QMF subband samples. A weighted correlation sum $\mathbf{r}^m(l)$ is calculated for every parameter time slot l and all QMF subbands m that have an adjoining parameter border.

$$\begin{cases} \mathbf{r}^m(0) = \mathbf{r}^m(-1)\alpha^{t(1)+1} + \sum_{n=0}^{t(1)} \alpha^{t(1)-n} \hat{w}_{\text{real}}^{n,m} \hat{w}_{\text{real}}^{n-1,m}, & l = 0 \\ \mathbf{r}^m(l) = \mathbf{r}^m(l-1)\alpha^{t(l+1)+1} + \sum_{n=t(l)+1}^{t(l+1)} \alpha^{t(l+1)-n} \hat{w}_{\text{real}}^{n,m} \hat{w}_{\text{real}}^{n-1,m}, & 1 \le l < L \end{cases}$$

where

$\alpha = \exp\left(-\frac{dt}{\tau}\right)$ and $dt = \frac{64}{F_S}$ and $\tau = 23.22$. Negative time index (-1) for a subband sample equals the last subband sample from the previous frame. In the same manner, negative time index for the correlation sum equals the last sum from the previous frame. Moreover, the real-valued QMF signal is computed as the sum of the downmix input channels, according to:

$$\hat{w}_{\text{real}}^{n,m} = \sum_{ch} \hat{\mathbf{x}}_{\text{real}}^{n,m}(ch)$$

The aliasing condition is set according to the following rule

$$\mathbf{a}^m(l) = \begin{cases} 1, & if \quad \mathbf{r}^m(l) > thres, \mathbf{r}^{m+1}(l) > thres, m\ odd \\ 1, & if \quad \mathbf{r}^m(l) < -thres, \mathbf{r}^{m+1}(l) < -thres, m\ even \\ 0, & otherwise \end{cases}$$

where $thres = \frac{1.0e^{-9}}{1-\alpha}$ for all parameter sets $0 \le l < L$ and for all QMF subbands m that are the last subband (highest in frequency) within a parameter band.

6.10.3.3 Aliasing equalization

The aliasing equalization is applied on copies of the matrices $\mathbf{W}_1^{l,k}$ and $\mathbf{W}_2^{l,k}$. The alias equalized matrices are first created by copying $\mathbf{W}_1^{l,k}$ and $\mathbf{W}_2^{l,k}$ as

$$\tilde{\mathbf{W}}_1^{l,k} = \mathbf{W}_1^{l,k} \text{ and } \tilde{\mathbf{W}}_2^{l,k} = \mathbf{W}_2^{l,k}$$

for $0 \le k < K$ and $0 \le l < L$.

For the QMF bands where aliasing condition is signaled the alias equalized matrices are modified according to

$$\begin{cases} \tilde{\mathbf{W}}_1^{l,k(m)} = \tilde{\mathbf{W}}_1^{l,k(m)+1} = \frac{1}{2}\left(\mathbf{W}_1^{l,k(m)} + \mathbf{W}_1^{l,k(m)+1}\right) \\ \tilde{\mathbf{W}}_2^{l,k(m)} = \tilde{\mathbf{W}}_2^{l,k(m)+1} = \frac{1}{2}\left(\mathbf{W}_2^{l,k(m)} + \mathbf{W}_2^{l,k(m)+1}\right) \end{cases}, \quad if \quad \mathbf{a}^m(l) = 1, \quad 0 \le l < L$$

where $k(m) = m + K - 64$. After the modification, the two matrices $\tilde{\mathbf{W}}_1^{l,k}$ and $\tilde{\mathbf{W}}_2^{l,k}$ replace the matrices $\mathbf{W}_1^{l,k}$ and $\mathbf{W}_2^{l,k}$.

6.10.4 Integrated matrixing

6.10.4.1 Introduction

The integrated matrixing is only used for the 5-1-5 configurations and reduces the complexity by setting the $\mathbf{M}_1^{n,k}$ matrix to a unity vector and modifying the $\mathbf{R}_2^{n,k}$ matrix, which in turn enables considerable complexity reductions of the decorrelators and decorrelator structure. For other configurations the definitions given in subclause 6.5 apply.

6.10.4.2 Pre-matrix M1

For the 5-1-5 configurations the $\mathbf{M}_1^{n,k}$ matrix is defined according to:

$$\mathbf{M}_1^{n,k} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{t}(l-1) \le n < \mathbf{t}(l), 0 \le l < L, 0 \le k < K\,.$$

6.10.4.3 Mix-matrix M2

Given that the $\mathbf{M}_1^{n,k}$ matrix is set to a unity vector for the 5-1-5 configurations, the mix-matrix $\mathbf{M}_2^{n,k}$ is redefined for the 5-1-5_1 and 5-1-5_2 configurations. Hence, the $\mathbf{M}_2^{n,k}$ matrix is defined according to:

$$\mathbf{M}_2^{n,k} = \begin{cases} \mathbf{W}_2^{l,k}\alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{W}_2^{-1,k} & ,0 \le n \le \mathbf{t}(l), l = 0 \\ \mathbf{W}_2^{l,k}\alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{W}_2^{l-1,k} & ,\mathbf{t}(l-1) < n \le \mathbf{t}(l), 1 \le l < L \end{cases}$$

for $0 \le l < L$, $0 \le k < K$ where

$$\alpha(n,l) = \begin{cases} \dfrac{n+1}{\mathbf{t}(l)+1} & ,l = 0 \\ \dfrac{n-\mathbf{t}(l-1)}{\mathbf{t}(l)-\mathbf{t}(l-1)} & ,otherwise \end{cases}$$

and $\mathbf{W}_2^{l,k}$ can be processed by smoothing according to:

$$\mathbf{W}_2^{l,k} = \begin{cases} \mathbf{s}_{\text{delta}}(l) \cdot \mathbf{R}_{2\text{LP}}^{l,\kappa(k)} + \left(1-\mathbf{s}_{\text{delta}}(l)\right) \cdot \mathbf{W}_2^{l-1,k} & ,\mathbf{S}_{\text{proc}}\left(l,\kappa(k)\right) = 1 \\ \mathbf{R}_{2\text{LP}}^{l,\kappa(k)} & ,\mathbf{S}_{\text{proc}}\left(l,\kappa(k)\right) = 0 \end{cases}$$

and where $\kappa(k)$ is given in Table A.31, where the first row is the hybrid band k, and the second row is the corresponding processing band.

$\mathbf{W}_2^{-1,k}$ corresponds to the last parameter set of the previous frame (zero for the first frame).

Where

$$\mathbf{R}_{2LP}^{l,m} = \mathbf{R}_2^{l,m}\mathbf{I}_5\mathbf{R}_1^{l,m}\mathbf{G}_1^{l,m}$$

and where $\mathbf{R}_2^{l,m}$, $\mathbf{R}_1^{l,m}$, and $\mathbf{G}_1^{l,m}$ are defined in subclause 5.5, and where $\mathbf{I}_5$ is the 5x5 unity matrix.

6.10.5 Matrix encoded surround compatibility

Similar as for normal operation mode, in low power mode, the matrix surround inversion matrix is a 3 by 3 matrix given by:

$$\mathbf{H}^{l,m} = \begin{bmatrix} h_{11}^{l,m} & h_{12}^{l,m} & 0 \\ h_{21}^{l,m} & h_{22}^{l,m} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

For $m < M_{\text{Proc}}^{c}$, i.e., for the complex-valued bands, the entries $h_{11}^{l,m}$, $h_{12}^{l,m}$, $h_{21}^{l,m}$ and $h_{22}^{l,m}$ are described by the equations in subclause 6.5.2.4. For $m \geq M_{\text{proc}}^{c}$, i.e., for the real-valued bands, the entries $h_{11}^{l,m}$, $h_{12}^{l,m}$, $h_{21}^{l,m}$ and $h_{22}^{l,m}$ are described as following.

First, the following intermediate variables are defined:

$$q_1^{l,m} = \frac{\left(w_1^{l,m}\right)^2}{1 - 2w_1^{l,m} + 2\left(w_1^{l,m}\right)^2},$$

$$q_2^{l,m} = \frac{\left(w_{21}^{l,m}\right)^2}{1 - 2w_2^{l,m} + 2\left(w_2^{l,m}\right)^2},$$

$$s^{l,m} = q_1^{l,m} + q_2^{l,m},$$

$$p^{l,m} = \frac{q_1^{l,m} \cdot q_2^{l,m}}{9},$$

$$b^{l,m} = 1 - 5p^{l,m}\sqrt{-11\left(p^{l,m}\right)^2 + \left(4s^{l,m} - 14\right)p^{l,m} + 1}.$$

Furthermore depending on $q_1^{l,m}$ and $q_2^{l,m}$ two more intermediate variables $r_\alpha^{l,m}$ and $r_\beta^{l,m}$ are calculated:

$$r_\alpha^{l,m} = \begin{cases} \dfrac{3b^{l,m}}{2\left(q_1^{l,m} - \left(q_1^{l,m}\right)^2\right)}, & if \quad 0 < q_1^{l,m} < 1; \\ \dfrac{q_2^{l,m} - \left(q_2^{l,m}\right)^2}{3\left(1 - 5p^{l,m}\right)}, & if \quad q_1^{l,m} \in \{0,1\}. \end{cases}$$

$$r_{\beta}^{l,m} = \begin{cases} \dfrac{3b^{l,m}}{2\left(q_2^{l,m} - \left(q_2^{l,m}\right)^2\right)}, & if \quad 0 < q_2^{l,m} < 1; \\ \dfrac{q_1^{l,m} - \left(q_1^{l,m}\right)^2}{3\left(1 - 5p^{l,m}\right)}, & if \quad q_2^{l,m} \in \{0,1\}. \end{cases}$$

The non-normalized entries of $\mathbf{H}_{l,m}$, denoted by $g_{11}^{l.m}$, $g_{12}^{l.m}$, $g_{21}^{l.m}$ and $g_{22}^{l.m}$ can then be determined as:

$$g_{11}^{l.m} = 1 - \frac{q_1^{l.m} \cdot r_{\alpha}^{l.m}}{3}, \qquad g_{12}^{l.m} = \frac{q_2^{l.m} - q_1^{l.m} \cdot r_{\alpha}^{l.m}}{\sqrt{3}},$$

$$g_{21}^{l.m} = \frac{q_1^{l.m} - q_2^{l.m} \cdot r_{\beta}^{l.m}}{\sqrt{3}}, \qquad g_{22}^{l.m} = 1 - \frac{q_2^{l.m} \cdot r_{\beta}^{l.m}}{3}.$$

The normalization coefficients $c_1^{l.m}$ and $c_2^{l.m}$ are calculated as:

$$c_1^{l.m} = 1 / \sqrt{\left(1 - q_1^{l.m}\right)\left(g_{11}^{l.m}\right)^2 + q_1^{l.m}\left(1 - \frac{q_2^{l.m}}{3}\right)^2},$$

$$c_2^{l.m} = 1 / \sqrt{\left(1 - q_2^{l.m}\right)\left(g_{22}^{l.m}\right)^2 + q_2^{l.m}\left(1 - \frac{q_1^{l.m}}{3}\right)^2}.$$

Finally, the matrix $\mathbf{H}_{l,m}$ is given as:

$$\mathbf{H}_{l,m} = \begin{bmatrix} h_{11}^{l,m} & h_{12}^{l,m} \\ h_{21}^{l,m} & h_{22}^{l,m} \end{bmatrix} = \begin{bmatrix} c_1 \cdot g_{11}^{l,m} & c_1 \cdot g_{12}^{l,m} \\ c_2 \cdot g_{21}^{l,m} & c_2 \cdot g_{22}^{l,m} \end{bmatrix}.$$

The way the variables $w_1^{l,m}$ and $w_2^{l,m}$ are determined is not affected in the low power version.

6.10.6 Enhance Matrix Mode of Low Power MPEG Surround

6.10.6.1 Introduction

In a similar fashion as the parameters are applied as described in subclause 6.10.3, the aliasing condition flags $\mathbf{a}^k(l)$ are determined and applied.

6.10.6.2 Parameter processing and interpolation

The process of state updates, aliasing flags $\mathbf{a}^k(l)$,estimation, and parameter positioning is depicted in Figure 48. A new down-mix frame is received starting at hybrid QMF down mix slot '0' up to N-1 (N=32). The correlation states $\mathbf{r}^m(l)$ are updated for every down-mix slot. For every four slots, the aliasing flags $\mathbf{a}^k(l)$ are set and applied four slots in advance.

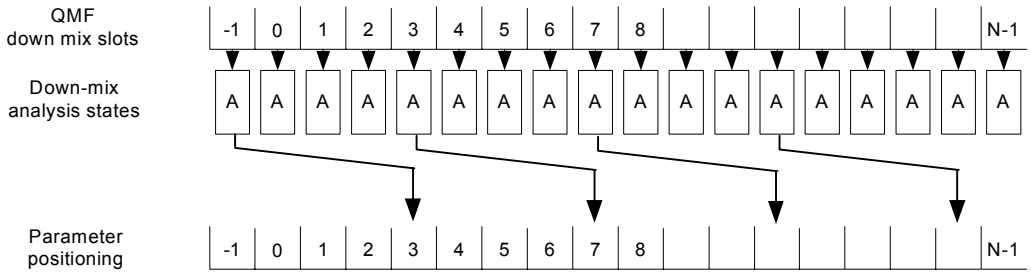

Figure 48 — Stereo down mix analysis procedure

Note that for Enhanced Matrix Mode low power decoding also the changes in subclause 6.10.5 hold. Furthermore, even though for the bands where $m \geq M_{\text{proc}}^{\text{c}}$ the sub-band signals are real-valued, the updating of the down mix analysis states is done using the equations of 6.2.2.

6.10.7 Low Power decorrelators

6.10.7.1 Introduction

Table 103 shows the decorrelator configurations used for the different decoding configurations in low power mode. This table corresponds to Table 89 for the normal (not low power) decoding mode. The decorrelators listed in the table, PS and Lattice IIR (LP0 and LP1) are defined in subclause 6.10.7.2 and subclause 6.10.7.3, respectively.

Table 103 — Decorrelator configuration as a function of decoder configuration

	Low power decorrelators			
Configuration	Lattice IIR, LP0	Lattice IIR LP1	PS	No decorrelation
5-1-5$_1$	$D_0(\,)$	$D_1(\,)$	$D_2(\,)$, $D_3(\,)$	-
5-1-5$_2$	$D_0(\,)$	$D_3(\,)$, $D_4(\,)$	$D_1(\,)$	-
5-2-5	-	-	$D_1^{\text{OTT}}(\,)$, $D_2^{\text{OTT}}(\,)$	$D_0^{\text{TTT}}(\,)$
7-2-7$_1$	$D_3(\,)$, $D_4(\,)$	-	$D_1^{\text{OTT}}(\,)$, $D_2^{\text{OTT}}(\,)$	$D_0^{\text{TTT}}(\,)$
7-2-7$_2$	$D_3(\,)$, $D_4(\,)$	-	$D_1^{\text{OTT}}(\,)$, $D_2^{\text{OTT}}(\,)$	$D_0^{\text{TTT}}(\,)$
7-5-7$_1$	-	-	$D_0(\,)$, $D_1(\,)$	-
7-5-7$_2$	-	-	$D_0(\,)$, $D_1(\,)$	-

As the TTT decorrelator, $D_0^{\text{TTT}}(\,)$ is disabled in low power mode, the pre-matrix, $\mathbf{M}_1$ and the mix-matrix, $\mathbf{M}_2$ defined in subclause 6.5 shall in case of $\mathbf{bsTttModeLow}(0)=0$ or $\mathbf{bsTttModeHigh}(0)=0$ be defined as if $\mathbf{bsTttModeLow}(0)=1$ or $\mathbf{bsTttModeHigh}(0)=1$ for their associated parameter bands, respectively.

The low power decoding mode utilizes two different decorrelator types according to subclause 6.10.7.2 and subclause 6.10.7.3. For all references to the parametric stereo standard (ISO/IEC 14496-3:2005(E)) in the subclauses below, the baseline configuration is assumed.

6.10.7.2 Low power parametric stereo decorrelator

The parametric stereo decorrelator as described in ISO/IEC 14496-3:2005(E) (Subpart 8: Technical description of parametric coding for high quality audio), subclause 8.6.4.5 De-correlation. In that text the full description of the decorrelator filter and transient reducer can be found. However, the constant $NR\_ALLPASS\_BANDS$ should be changed from 30 to M^{c}_{proc}, and $\mathbf{H}_k(z)$ should be changed from

$$H_k(z) = z^{-2} \cdot \varphi_{\text{Fract}}(k) \cdot \prod_{m=0}^{NR\_ALLPASS\_LINKS-1} \frac{Q_{\text{Fract\_allpass}}(k,m) z^{-d(m)} - a(m) g_{\text{DecaySlope}}(k)}{1 - a(m) g_{\text{DecaySlope}}(k) Q_{\text{Fract\_allpass}}(k,m) z^{-d(m)}}$$

to

$$H_k(z) = z^{-8} \cdot \varphi_{\text{Fract}}(k) \cdot \prod_{m=0}^{NR\_ALLPASS\_LINKS-1} \frac{Q_{\text{Fract\_allpass}}(k,m) z^{-d(m)} - a(m) g_{\text{DecaySlope}}(k)}{1 - a(m) g_{\text{DecaySlope}}(k) Q_{\text{Fract\_allpass}}(k,m) z^{-d(m)}}$$

This decorrelator gives the final output: $d^{n,k}_{\text{PS,s}}$.

6.10.7.3 Low power lattice IIR decorrelator

The low power lattice IIR decorrelators are having the same basic structure as described in subclause 6.6.2, but with the following modifications. The bitstream element bsDecorrConfig is interpreted differently, instead the corresponding QMF split frequencies in Table 90 are:

Table 104 — Division of hybrid subbands for low power

bsDecorrConfig	0-2	3-15
k_0	-	Reserved
k_1	0-13	Reserved
k_2	14-29	Reserved
k_3	30-70	Reserved

This gives the decorrelated outputs prior to the energy adjustment, $d^{n,k}_{\text{LP0,filt}}$ and $d^{n,k}_{\text{LP1,filt}}$. For those signals the energy adjustment according to subclause 6.6.3 is replaced by the transient reducer described in ISO/IEC 14496-3:2005(E) (Subpart 8: Technical description of parametric coding for high quality audio), subclauses 8.6.4.5.3-4, where the transient attenuator, $\mathbf{G}_{\text{TransientRatioMapped}}(k,n)$ is mapped to the gain factor $g^{n,\kappa(k)}_{\text{s}}$ according to:

$$g^{n,\kappa(k)}_{s} = \mathbf{G}_{\text{TransientRatioMapped}}(k,n)$$

This gives the final decorrelator outputs:

$$d^{n,k}_{\text{LP0,s}} = g^{n,\kappa(k)}_{\text{s}} \cdot d^{n,k}_{\text{LP0,filt}}$$
$$d^{n,k}_{\text{LP1,s}} = g^{n,\kappa(k)}_{\text{s}} \cdot d^{n,k}_{\text{LP1,filt}}$$

6.10.8 Residual coding

Residual coding is only applied for $M_{\text{proc}}^{\text{c}}$ processing bands. In addition, the complex subbands of the partially complex QMF representation correspond to modulation different from that that of the high quality decoder. Therefore, the final modification at the end of subclause 6.9.2.5.2 should be replaced by

$$\mathbf{Z} = \left[\exp\left(-\frac{j\pi}{4}(2\cdot 0+1)\right)\mathbf{z}_0^T \quad \exp\left(-\frac{j\pi}{4}(2\cdot 1+1)\right)\mathbf{z}_1^T \quad \cdots \right.$$
$$\left. \exp\left(-\frac{j\pi}{4}\left(2\cdot\left(L_{\text{qmf}}-1\right)+1\right)\right)\mathbf{z}_{L_{\text{qmf}}-1}^T \right] \in C^{2N_{\text{qmf}},L_{\text{qmf}}}.$$

6.10.9 Low Power Binaural decoding

6.10.9.1 Alias equalization

The aliasing detection as outlined in subclause 6.10.3 shall be used. Given the aliasing detection flags defined by $\mathbf{a}^m(l)$, the aliasing equalization is applied on a copy of the matrix $\mathbf{H}_1^{l,k}$. The alias equalized matrices are first created by copying $\mathbf{H}_1^{l,k}$ as

$$\tilde{\mathbf{H}}_1^{l,k} = \mathbf{H}_1^{l,k}$$

for $0 \le k < K$ and $0 \le l < L$. For the QMF bands where aliasing condition is signaled the alias equalized matrices are modified according to

$$\tilde{\mathbf{H}}_1^{l,k(m)} = \tilde{\mathbf{H}}_1^{l,k(m)+1} = \frac{1}{2}\left(\mathbf{H}_1^{l,k(m)} + \mathbf{H}_1^{l,k(m)+1}\right) \quad if \quad \mathbf{a}^m(l) = 1, \quad 0 \le l < L$$

where $k(m) = m + K - 64$. After the modification, the matrix $\tilde{\mathbf{H}}_1^{l,k}$ replaces the matrix $\mathbf{H}_1^{l,k}$.

6.10.9.2 Decorrelator

For mono based (5-1-5) binaural decoding, the low power parametric stereo decorrelator defined in subclause 6.10.7.2 shall be used.

6.11 MPEG Surround Binaural coder

6.11.1 Introduction

The MPEG Surround binaural processing, can either be done as a decoder process converting the MPEG Surround downmix signal into a binaural downmix signal, that provides a surround sound experience when listened to over headphones, or alternatively, the MPEG Surround encoder can create a binaural (or a 3D stereo) downmix, that provides a surround sound experience when listening to the downmix without any additional decoder processing. For the latter mode, information on the binaural processing done to the downmix is included in the MPEG Surround bitstream so that it can be inverted if the downmix and MPEG Surround bitstream is decoded into a true multi-channel representation.

The extension for binaural audio decoding of MPEG Surround is outlined in 6.11.4. A “binaural decoder” has as input the MPEG surround data stream. This stream may either be based on a mono (515 configuration) or stereo (525 configuration) downmix. The downmix signal is processed by a hybrid QMF analysis filterbank as outlined in subclause 6.3. A parameter conversion stage converts the transmitted MPEG Surround parameters as generated by MPEG surround encoder to spatial synthesis parameters that include 3D cues, based on HRTF parameter input to the MPEG Surround binaural decoder. In other words, HRTF processing is performed in the parameter domain. Subsequently, a spatial synthesis stage computes a stereo output signal by means of parameter-controlled matrixing, and decorrelation (only applied in the case of a mono downmix). For both the matrixing and decorrelation operations, standard MPEG surround building blocks are used. Finally, two QMF synthesis filter banks, as outlined in subclause 6.3 compute the (binaural) time-domain output signals. An overview is given in Figure 49, where the QMF Analysis and QMF Synthesis blocks include the Nyquist filterbanks.

The binaural decoder can furthermore be extended with either a time-domain or QMF-domain reverberation module.

The MPEG Surround binaural decoder operates on the MPEG Surround data stream. Although technically feasible, not all elements in the data stream are applicable for the binaural decoder since these have a poor complexity versus quality trade-off. An MPEG Surround decoder operating in binaural mode is not obliged to decode the data stream elements pertaining to; Guided Envelope Shaping (syntax element EnvelopeReshapeHuff(), Table 21), Shaping of decorrelator signal (syntax element TempShapeData(), Table 20), and residual coding data (syntax element ResidualData(), Table 34 and syntax element ArbitraryDownmixResidualData(), Table 36).

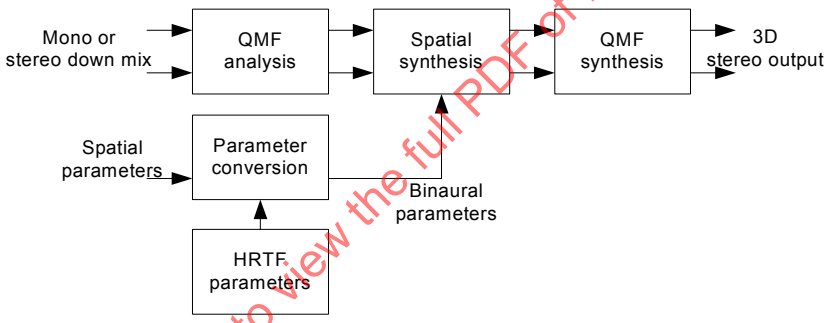

Figure 49 — MPEG Surround binaural decoder

The binaural MPEG Surround decoder can be operated in the Enhanced Matrix Mode, where the parameters are estimated according to subclause 6.2, as outlined in the following figure.

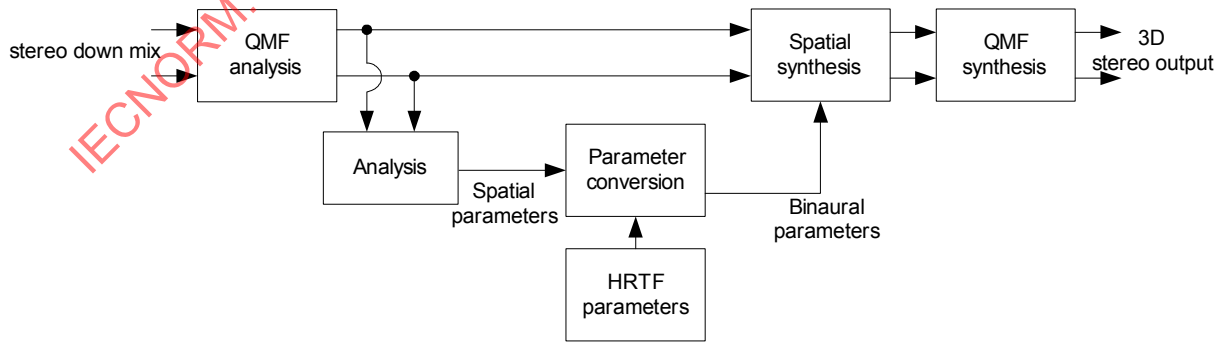

Figure 50 — MPEG Surround binaural decoder operated in Enhance Matrix Mode.

For all downmix channel configurations (mono or stereo), the spatial synthesis consists of a 2x2 (filter) matrix, according to:

$$\mathbf{y}_{\mathrm{B}}^{n,k} = \begin{bmatrix} y_{\mathrm{L_B}}^{n,k} \\ y_{\mathrm{R_B}}^{n,k} \end{bmatrix} = \sum_{i=0}^{N_q-1} \mathbf{H}_2^{n-i,k} \mathbf{y}_0^{n-i,k} = \sum_{i=0}^{N_q-1} \begin{bmatrix} \mathbf{h}_{11}^{n-i,k} & \mathbf{h}_{12}^{n-i,k} \\ \mathbf{h}_{21}^{n-i,k} & \mathbf{h}_{22}^{n-i,k} \end{bmatrix} \begin{bmatrix} y_{\mathrm{L_0}}^{n-i,k} \\ y_{\mathrm{R_0}}^{n-i,k} \end{bmatrix}, \quad 0 \le k < K$$

,

with $\mathbf{y}_0^{n,k}$ being the QMF-domain input channels and $\mathbf{y}_{\mathrm{B}}^{n,k}$ being the binaural output channels, k represents the hybrid QMF channel index, and i is the HRTF filter tap index, and n is the QMF slot index.

The number of filter taps (N_q) in the 2x2 matrix can be varied depending on the presence or absence of reverberation in the HRTF impulse responses, or the accuracy desired for the HRTF representation. The conversion of the HRTFs to one-tap matrix elements is outlined in subclause C.1. For multi-tap matrix elements the HRTF conversion as outlined in subclause B.1 shall be used.

If a one-tap representation of the HRTF is used, i.e. N_q=1, this allows for the creation of a 2x2 matrix that is invertible. This property has important advantages when using the spatial synthesis technique in the encoder, as outlined in Figure 51.

The fact that the spatial synthesis module comprises a 2x2 matrix in the filterbank domain is exploited in the MPEG Surround encoder for generating a 3D/binaural downmix (see upper panel of Figure 51) by binaural synthesis module in the encoder as post-process to the MPEG Surround encoder. Given this encoder binaural post-processing approach the MPEG Surround decoder (as shown in the lower panel) can apply the inverse processing on the received 3D/binaural stereo downmix such that no quality degradation on the 5.1-channel reconstruction occurs when transmitting a 3D/binaural downmix. Optionally, the 3D inverse stereo downmix can again be processed using the binaural decoder for personal HRTFs.

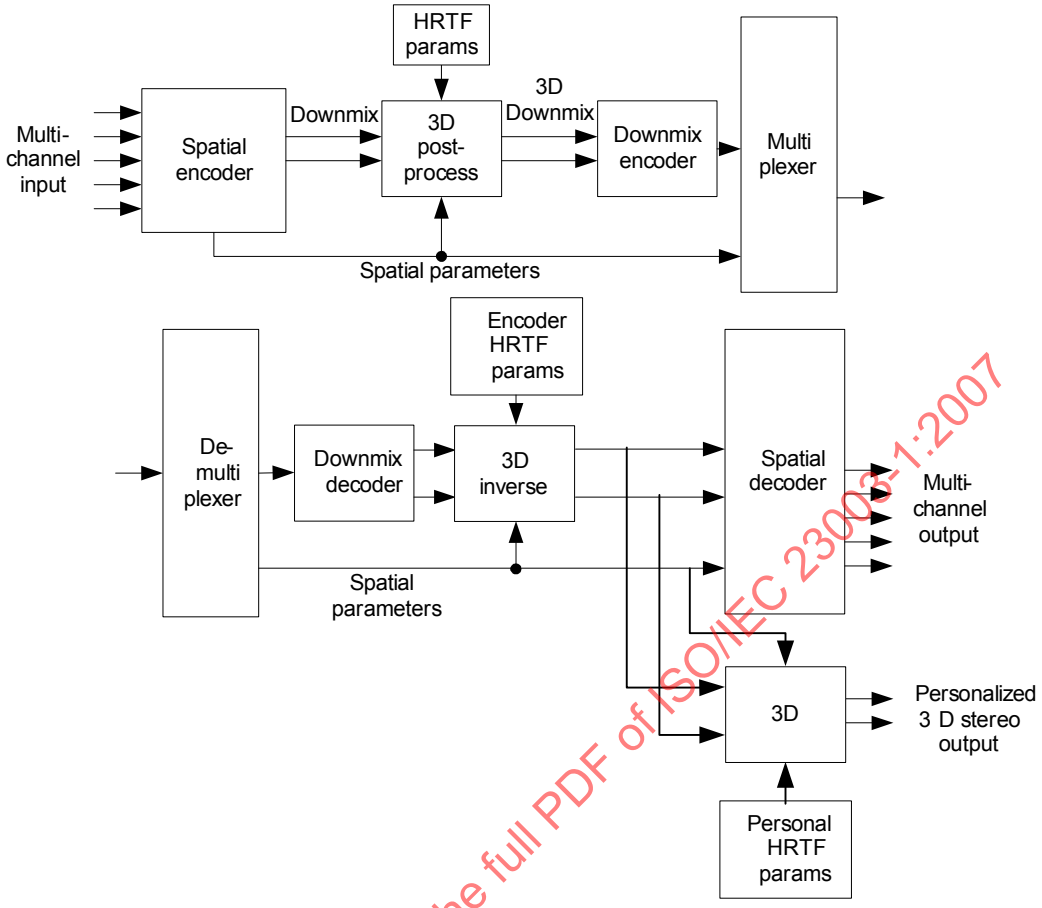

Figure 51 — 3D/binaural downmix as postprocessor of the spatial encoder (upper panel) and inverse 3D/binaural processing in the decoder (lower panel). In addition, Binaural decoding using personalized HRTFs can again be applied to the 3D inverted downmix.

6.11.2 HRTF representations

The MPEG Surround binaural decoder supports two different representations of the HRTF filters, a parametric representation and a complex QMF filter representation. Conversion from time domain HRTFs to the parametric representation is outlined in Annex C.1, conversion from the time-domain HRTFs to the complex QMF filter representation is outlined in Annex B.1.

The parameters have subscripts corresponding to the binaural output channels $Y \in \{\mathrm{L},\mathrm{R}\}$ and the virtual multi-channel source $X \in \{\mathrm{L},\mathrm{R},\mathrm{C},\mathrm{Ls},\mathrm{Rs}\}$. For the parametric representation, the HRTFs are described by $P^{m}_{\mathrm{Y,X}}$, ϕ^{m}_{X} and ρ^{m}_{X} for parameter bands $0 \le m < M_{\mathrm{Proc}}$ corresponding to an average level per parameter band, an average phase difference per parameter band, and a coherence measure per parameter band.

For the complex QMF filter representation the HRTFs are described by $\left(\hat{v}_{\mathrm{Y,X}}\right)_{m}(n)$ for QMF subband $m = 0,1,\ldots,63$ and QMF time slot $n = 0,1,\ldots,N_q - 1$.

6.11.3 Configurations of the binaural decoder

The binaural decoder can be implemented in a High Quality version or a Low Power version. The low-power version can utilize the partially complex filterbank as outlined in subclause 6.10.2 and the aliasing reduction algorithm outlined in subclause 6.10.3. Furthermore the Low Power version does not support HRTF filter representations in the QMF of more than 1 tap, i.e. the parametric HRTFs as outlined in subclause 6.11.4.1 and 6.11.4.2.2 shall be used. Finally, the Low Power version of the binaural decoder only supports the use of the Parametric Stereo decorrelator outlined in subclause 6.10.7.2.

Table 105 — Configurations of the binaural decoder

	LP Filterbanks	HQ filterbanks
Low power Binaural decoder	X	
High Quality binaural decoder, parametric approach		X
High Quality binaural decoder, filter approach		X

6.11.4 Binaural decoder description

6.11.4.1 Mono-based configuration and synthesis

6.11.4.1.1 Introduction

For mono based configurations, a 5.1-channel audio signal is encoded by an MPEG surround encoder as a mono signal plus MPEG Surround parameters according to a '5151' or '5152' configuration, respectively.

The decoding scheme of both configurations is outlined in the following subclauses. A QMF analysis filter bank processes the mono downmix signal. The resulting QMF-domain signal is fed into a matrixing block, combined with the output of a decorrelator. The decorrelator generates a signal with statistically very similar properties as its input, except for the fact that its input and output waveforms have no significant cross correlation. The selection of the decorrelator provides for a complexity versus quality trade-off. For the high quality version, the decorrelator 0 according to subclause 6.6 shall be used. For the low power version, the decorrelator as specified in subclause 6.10.7.2 shall be used.

The spatial parameters from the MPEG surround data stream are converted to binaural 3D parameters with the help of HRTF parameters. The resulting binaural parameters control the matrixing block to generate the QMF-domain binaural output signal, which is converted to the time domain using a QMF synthesis filterbank.

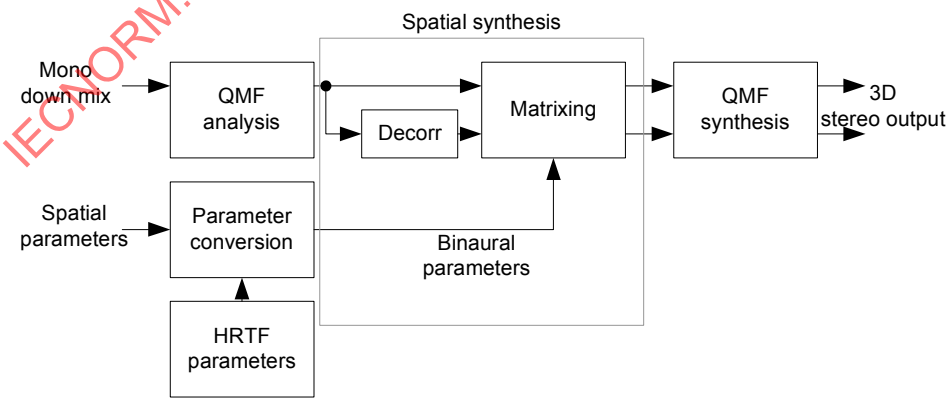

Figure 52 — 3D binaural decoder for '515' spatial configurations

The aim of the spatial synthesis stage is to reconstruct the binaural statistical properties at the output as if a 5.1-multi-channel decoded signal was filtered with the HRTFs and subsequently downmixed to binaural stereo.

The binaural synthesis step in case of a mono-based configuration is similar to the generalized synthesis equation given in subclause 6.11.1, with the difference that only one-tap (N_q=1) HRTF parameterizations is supported:

$$\mathbf{y}_{\mathrm{B}}^{n,k} = \begin{bmatrix} y_{\mathrm{L_B}}^{n,k} \\ y_{\mathrm{R_B}}^{n,k} \end{bmatrix} = \mathbf{H}_2^{n,k} \begin{bmatrix} y_{\mathrm{m}}^{n,k} \\ D\left(y_{\mathrm{m}}^{n,k}\right) \end{bmatrix} = \begin{bmatrix} h_{11}^{n,k} & h_{12}^{n,k} \\ h_{21}^{n,k} & h_{22}^{n,k} \end{bmatrix} \begin{bmatrix} y_{\mathrm{m}}^{n,k} \\ D\left(y_{\mathrm{m}}^{n,k}\right) \end{bmatrix}, \quad 0 \le k < K \,,$$

with $y_{\mathrm{m}}^{n,k}$ being the QMF-domain mono input channel, $D\left(y_{\mathrm{m}}^{n,k}\right)$ being the decorrelated version of the same, and $\mathbf{y}_{\mathrm{B}}^{n,k}$ being the binaural output channels, k represents the hybrid QMF channel index, and n is the QMF slot index.

The mixing matrix $\mathbf{H}_2^{n,k}$ is a time interpolated and frequency mapped version of a mixing matrix $\mathbf{H}_1^{l,m}$, defined for by the HRTF filters and the MPEG Surround spatial parameters for every time-slot and QMF hybrid band. This scheme is equivalent to a parametric stereo decoder.

The mixing matrix $\mathbf{H}_1^{l,m}$ defined for every parameter set l and processing band m. Equivalent to conventional MPEG surround decoding, the mixing matrix is linearly interpolated for QMF slots in between parameter positions, according to:

$$\mathbf{H}_2^{n,k} = \begin{cases} \mathbf{H}_{\mathrm{temp}}^{l,\kappa(k)} \alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{H}_{\mathrm{temp}}^{-1,\kappa(k)}, & ,0 \le n \le \mathbf{t}(l), l = 0 \\ \mathbf{H}_{\mathrm{temp}}^{l,\kappa(k)} \alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{H}_{\mathrm{temp}}^{l-1,\kappa(k)}, & ,\mathbf{t}(l-1) < n \le \mathbf{t}(l), 1 \le l < L \end{cases}$$

for $0 \le l < L$, $0 \le k < K$ where

$$\alpha(n,l) = \begin{cases} \dfrac{n+1}{\mathbf{t}(l)+1} & ,l = 0 \\ \dfrac{n-\mathbf{t}(l-1)}{\mathbf{t}(l)-\mathbf{t}(l-1)} & ,otherwise \end{cases}$$

and where

and $\mathbf{H}_{\mathrm{temp\_1}}^{l,\kappa(k)}$ can be processed by smoothing according to:

$$\mathbf{H}_{\mathrm{temp}}^{l,m} = \begin{cases} \mathbf{s}_{delta}(l)\mathbf{H}_{\mathrm{temp\_1}}^{l,m} + \left(1-\mathbf{s}_{delta}(l)\right)\mathbf{H}_{\mathrm{temp}}^{l-1,m} & ,\mathbf{S}_{proc}(l,m) = 1 \\ \mathbf{H}_{\mathrm{temp\_1}}^{l,m} & ,\mathbf{S}_{proc}(l,m) = 0 \end{cases}$$

where

$$\mathbf{H}_{\mathrm{temp\_1}}^{l,m} = \begin{cases} \mathrm{Re}\left\{\mathbf{H}_1^{l,m}\right\} & m \ge M_{\mathrm{proc}}^{\mathrm{c}} \\ \mathbf{H}_1^{l,m} & otherwise \end{cases}, \quad 0 \le m < M_{\mathrm{proc}}$$

and where $\kappa(k)$ is given by Table A.31, and maps the hybrid band k to the corresponding processing band.

Hence, for one tap (N_q=1) HRTF parameterizations in the QMF domain, the mixing matrix $\mathbf{H}_1^{l,m}$ is computed on a processing band basis (i.e., 28 bands) instead of a full hybrid-band basis (71 bands), and the matrix entries are real-valued for processing bands from $M_{\text{proc}}^{\text{c}}$ onwards.

The matrix $\mathbf{H}_1^{l,m}$ is defined in the following subclauses for the 5151 and 5152 configurations.

6.11.4.1.2 Matrix elements

The matrix elements of

$$\mathbf{H}_1^{l,m} = \begin{bmatrix} h_{11}^{l,m} & h_{12}^{l,m} \\ h_{21}^{l,m} & -\left(h_{12}^{l,m}\right)^* \end{bmatrix}, \quad 0 \le m < M_{\text{Proc}}, 0 \le l < L$$

are defined by the relative signal powers $\sigma_{\text{L}}^{l,m}$ and $\sigma_{\text{R}}^{l,m}$, and the complex cross-spectrum $\left\langle L_{\text{B}} R_{\text{B}}^* \right\rangle^{l,m}$ for the two binaural output channels, for each spatial parameter set estimated for each parameter position and processing band, according to:

$$h_{11}^{l,m} = \sigma_{\text{L}}^{l,m}\left(\cos\left(IPD_{\text{B}}^{l,m}/2\right) + j\sin\left(IPD_{\text{B}}^{l,m}/2\right)\right)\left(iid^{l,m} + ICC_{\text{B}}^{l,m}\right)d^{l,m}$$

$$h_{12}^{l,m} = \sigma_{\text{L}}^{l,m}\left(\cos\left(IPD_{\text{B}}^{l,m}/2\right) + j\sin\left(IPD_{\text{B}}^{l,m}/2\right)\right)\sqrt{1-\left(\left(iid^{l,m} + ICC_{\text{B}}^{l,m}\right)d^{l,m}\right)^2}$$

$$h_{21}^{l,m} = \sigma_{\text{R}}^{l,m}\left(\cos\left(IPD_{\text{B}}^{l,m}/2\right) - j\sin\left(IPD_{\text{B}}^{l,m}/2\right)\right)\left(1 + iid^{l,m} ICC_{\text{B}}^{l,m}\right)d^{l,m}$$

with $iid^{l,m}$ equal to

$$iid^{l,m} = \frac{\sigma_{\text{L}}^{l,m}}{\sigma_{\text{R}}^{l,m}},$$

and $d^{l,m}$ equal to:

$$d^{l,m} = \frac{1}{\sqrt{1+\left(iid^{l,m}\right)^2 + 2iid^{l,m} ICC_{\text{B}}^{l,m}}}.$$

The coherence of the binaural output $ICC_{\text{B}}^{l,m}$ and the IPD is computed as:

$$ICC_{\text{B}}^{l,m} = \begin{cases} \dfrac{\Re\left(\left\langle L_{\text{B}} R_{\text{B}}^* \right\rangle^{l,m}\right)}{\sigma_{\text{L}}^{l,m}\sigma_{\text{R}}^{l,m}} & ,if\ ICC_{\text{Btmp}}^{l,m} < 0.6 \\ ICC_{\text{Btmp}}^{l,m} & ,otherwise \end{cases}$$

$$IPD_{\mathrm{B}}^{l,m} = \begin{cases} 0 & ,m > 12,\ ICC_{\mathrm{Btmp}}^{l,m} < 0.6 \\ \arg\left(\left\langle L_{\mathrm{B}} R_{\mathrm{B}}^{*} \right\rangle^{l,m}\right) & ,otherwise \end{cases}.$$

where

$$ICC_{\mathrm{Btmp}}^{l,m} = \frac{\left|\left\langle L_{\mathrm{B}} R_{\mathrm{B}}^{*} \right\rangle^{l,m}\right|}{\sigma_{\mathrm{L}}^{l,m}\sigma_{\mathrm{R}}^{l,m}}$$

and $\Re(x)$ denotes the real value of x. For high-frequency processing bands (band $M_{\mathrm{proc}}^{\mathrm{c}}$ onwards), the IPD_{B} value is set to zero.

The variable $\sigma_{\mathrm{L}}^{l,m}$, $\sigma_{\mathrm{R}}^{l,m}$, and the complex cross-spectrum $\left\langle L_B R_B^{*} \right\rangle^{l,m}$ are defined differently for the 5151 and 5152 configurations in the following subclauses.

6.11.4.1.3 Power reconstruction and cross-spectrum variables for 5151 and 5152 configurations

6.11.4.1.3.1 5151 configuration

In order to define the (relative) power $\left(\sigma_{\mathrm{X}}^{l,m}\right)^2$ for $X \in \{\mathrm{L,R}\}$, i.e. the left and right 3D/binaural output signal, (with respect to the mono input signal), the transmitted CLD parameters are converted to (relative) signal powers $\left(\sigma_{\mathrm{X}}^{l,m}\right)^2$ for each virtual loudspeaker $X \in \{\mathrm{L,R,C,Ls,Rs}\}$, according to:

$$\left(\sigma_{\mathrm{L}}^{l,m}\right)^2 = r_1\left(CLD_0^{l,m}\right) r_1\left(CLD_1^{l,m}\right) r_1\left(CLD_3^{l,m}\right)$$

$$\left(\sigma_{\mathrm{R}}^{l,m}\right)^2 = r_1\left(CLD_0^{l,m}\right) r_1\left(CLD_1^{l,m}\right) r_2\left(CLD_3^{l,m}\right)$$

$$\left(\sigma_{\mathrm{C}}^{l,m}\right)^2 = r_1\left(CLD_0^{l,m}\right) r_2\left(CLD_1^{l,m}\right) / g_{\mathrm{c}}^2$$

$$\left(\sigma_{\mathrm{Ls}}^{l,m}\right)^2 = r_2\left(CLD_0^{l,m}\right) r_1\left(CLD_2^{l,m}\right) / g_{\mathrm{s}}^2$$

$$\left(\sigma_{\mathrm{Rs}}^{l,m}\right)^2 = r_2\left(CLD_0^{l,m}\right) r_2\left(CLD_2^{l,m}\right) / g_{\mathrm{s}}^2$$

with

$$r_1(CLD) = \frac{10^{CLD/10}}{1+10^{CLD/10}},$$

and

$$r_2(CLD) = \frac{1}{1+10^{CLD/10}},$$

for $0 \le m < M_{proc}, 0 \le l < L$,

and where

$$CLD_X^{l,m} = \mathbf{D}_{CLD}(X,l,m), \quad 0 \le X < 4, 0 \le m < M_{proc}, 0 \le l < L$$

.

The constants g_c and g_s represent the down-mix gain factors for centre and surround, respectively.

Given the above, and the HRTF parameters $P_{Y,X}^m$, ϕ_X^m and ρ_X^m, the (relative) power $\left(\sigma_L^{l,m}\right)^2$ and $\left(\sigma_R^{l,m}\right)^2$ for the left and right 3D/binaural output signal (with respect to the mono input signal) is given by:

$$\begin{aligned}\left(\sigma_X^{l,m}\right)^2 = &\left(P_{X,C}^m\right)^2\left(\sigma_C^{l,m}\right)^2 + \left(P_{X,L}^m\right)^2\left(\sigma_L^{l,m}\right)^2 + \left(P_{X,Ls}^m\right)^2\left(\sigma_{Ls}^{l,m}\right)^2 + \left(P_{X,R}^m\right)^2\left(\sigma_R^{l,m}\right)^2 + \left(P_{X,Rs}^m\right)^2\left(\sigma_{Rs}^{l,m}\right)^2 + \ldots \\ &P_{X,L}^m P_{X,R}^m \rho_L^m \sigma_L^{l,m} \sigma_R^{l,m} ICC_3^{l,m} \cos\left(\phi_L^m\right) + \ldots \\ &P_{X,L}^m P_{X,R}^m \rho_R^m \sigma_L^{l,m} \sigma_R^{l,m} ICC_3^{l,m} \cos\left(\phi_R^m\right) + \ldots \\ &P_{X,Ls}^m P_{X,Rs}^m \rho_{Ls}^m \sigma_{Ls}^{l,m} \sigma_{Rs}^{l,m} ICC_2^{l,m} \cos\left(\phi_{Ls}^m\right) + \ldots \\ &P_{X,Ls}^m P_{X,Rs}^m \rho_{Rs}^m \sigma_{Ls}^{l,m} \sigma_{Rs}^{l,m} ICC_2^{l,m} \cos\left(\phi_{Rs}^m\right)\end{aligned}$$

for $0 \le m < M_{Proc}, 0 \le l < L$,

and where

$$ICC_3^{l,m} = \mathbf{D}_{ICC}(3,l,m), \quad ,0 \le m < M_{proc}, 0 \le l < L$$

$$ICC_2^{l,m} = \mathbf{D}_{ICC}(2,l,m), \quad ,0 \le m < M_{proc}, 0 \le l < L .$$

The complex cross-spectrum $\left\langle L_B R_B^* \right\rangle^{l,m}$ for each spatial parameter set is estimated for each parameter position and processing band as:

$$\begin{aligned}\left\langle L_B R_B^* \right\rangle^{l,m} = &\left(\sigma_C^{l,m}\right)^2 P_{L,C}^m P_{R,C}^m \rho_C^m \exp\left(j\phi_C^m\right) + \ldots \\ &\left(\sigma_L^{l,m}\right)^2 P_{L,L}^m P_{R,L}^m \rho_L^m \exp\left(j\phi_L^m\right) + \ldots \\ &\left(\sigma_R^{l,m}\right)^2 P_{L,R}^m P_{R,R}^m \rho_R^m \exp\left(j\phi_R^m\right) + \ldots \\ &\left(\sigma_{Ls}^{l,m}\right)^2 P_{L,Ls}^m P_{R,Ls}^m \rho_{Ls}^m \exp\left(j\phi_{Ls}^m\right) + \ldots \\ &\left(\sigma_{Rs}^{l,m}\right)^2 P_{L,Rs}^m P_{R,Rs}^m \rho_{Rs}^m \exp\left(j\phi_{Rs}^m\right) + \ldots \\ &P_{L,L}^m P_{R,R}^m \sigma_L^{l,m} \sigma_R^{l,m} ICC_3 + \ldots \\ &P_{L,Ls}^m P_{R,Rs}^m \sigma_{Ls}^{l,m} \sigma_{Rs}^{l,m} ICC_2 + \ldots \\ &P_{L,Rs}^m P_{R,Ls}^m \sigma_{Ls}^{l,m} \sigma_{Rs}^{l,m} ICC_2 \rho_{Ls}^m \exp\left(j\left(\phi_{Rs}^m + \phi_{Ls}^m\right)\right) + \ldots \\ &P_{L,R}^m P_{R,L}^m \sigma_L^{l,m} \sigma_R^{l,m} ICC_3 \rho_L^m \exp\left(j\left(\phi_R^m + \phi_L^m\right)\right)\end{aligned}$$

for $0 \le m < M_{Proc}, 0 \le l < L$,

and where

$$ICC_3^{l,m} = \mathbf{D}_{ICC}(3,l,m), \quad ,0 \le m < M_{proc}, 0 \le l < L$$

$$ICC_2^{l,m} = \mathbf{D}_{ICC}(2,l,m), \quad ,0 \le m < M_{proc}, 0 \le l < L\,.$$

6.11.4.1.3.2 5152 configuration

Similar to the 5151 configuration, in order to define the (relative) power $\left(\sigma_{\mathrm{X}}^{l,m}\right)^2$ for X=L,R, i.e. the left and right 3D/binaural output signal, (with respect to the mono input signal), the transmitted CLD parameters are converted to (relative) signal powers $\left(\sigma_{\mathrm{X}}^{l,m}\right)^2$ for each virtual loudspeaker $X = \{L, R, C, Ls, Rs\}$, according to:

$$\left(\sigma_{\mathrm{L}}^{l,m}\right)^2 = r_1\left(CLD_0^{l,m}\right) r_1\left(CLD_1^{l,m}\right) r_1\left(CLD_3^{l,m}\right)$$

$$\left(\sigma_{\mathrm{R}}^{l,m}\right)^2 = r_1\left(CLD_0^{l,m}\right) r_2\left(CLD_1^{l,m}\right) r_1\left(CLD_4^{l,m}\right)$$

$$\left(\sigma_{\mathrm{C}}^{l,m}\right)^2 = r_2\left(CLD_0^{l,m}\right) \Big/ g_c^2$$

$$\left(\sigma_{\mathrm{Ls}}^{l,m}\right)^2 = r_1\left(CLD_0^{l,m}\right) r_1\left(CLD_1^{l,m}\right) r_2\left(CLD_3^{l,m}\right) \Big/ g_s^2$$

$$\left(\sigma_{\mathrm{Rs}}^{l,m}\right)^2 = r_1\left(CLD_0^{l,m}\right) r_2\left(CLD_1^{l,m}\right) r_2\left(CLD_4^{l,m}\right) \Big/ g_s^2$$

with

$$r_1(CLD) = \frac{10^{CLD/10}}{1+10^{CLD/10}}$$

and

$$r_2(CLD) = \frac{1}{1+10^{CLD/10}},$$

for $0 \le m < M_{proc}, 0 \le l < L$,

and where

$$CLD_X^{l,m} = \mathbf{D}_{CLD}(X,l,m), \quad 0 \le X < 5, 0 \le m < M_{proc}, 0 \le l < L\,.$$

The constants g_c and g_s represent the down-mix gain factors for centre and surround, respectively.

Given the above, and the HRTF parameters $P^{m}_{\mathrm{Y,X}}$, ϕ^{m}_{X} and ρ^{m}_{X}, the (relative) power $\left(\sigma^{l,m}_{\mathrm{L}}\right)^2$ and $\left(\sigma^{l,m}_{\mathrm{R}}\right)^2$ for the left and right 3D/binaural output signals (with respect to the mono input signal) is given by:

$$\begin{aligned}\left(\sigma^{l,m}_{\mathrm{X}}\right)^2 = &\left(P^{m}_{\mathrm{X,C}}\right)^2\left(\sigma^{l,m}_{\mathrm{C}}\right)^2 + \left(P^{m}_{\mathrm{X,L}}\right)^2\left(\sigma^{l,m}_{\mathrm{L}}\right)^2 + \left(P^{m}_{\mathrm{X,Ls}}\right)^2\left(\sigma^{l,m}_{\mathrm{Ls}}\right)^2 + \left(P^{m}_{\mathrm{X,R}}\right)^2\left(\sigma^{l,m}_{\mathrm{R}}\right)^2 + \left(P^{m}_{\mathrm{X,BR}}\right)^2\left(\sigma^{l,m}_{\mathrm{BR}}\right)^2 + \ldots \\ &P^{m}_{\mathrm{X,L}}P^{m}_{\mathrm{X,R}}\rho^{m}_{\mathrm{L}}\sigma^{l,m}_{\mathrm{L}}\sigma^{l,m}_{\mathrm{R}}ICC^{l,m}_{1}\cos\left(\phi^{m}_{\mathrm{L}}\right) + \ldots \\ &P^{m}_{\mathrm{X,L}}P^{m}_{\mathrm{X,R}}\rho^{m}_{\mathrm{R}}\sigma^{l,m}_{\mathrm{L}}\sigma^{l,m}_{\mathrm{R}}ICC^{l,m}_{1}\cos\left(\phi^{m}_{\mathrm{R}}\right) + \ldots \\ &P^{m}_{\mathrm{X,Ls}}P^{m}_{\mathrm{X,Rs}}\rho^{m}_{\mathrm{Ls}}\sigma^{l,m}_{\mathrm{Ls}}\sigma^{l,m}_{\mathrm{Rs}}ICC^{l,m}_{1}\cos\left(\phi^{m}_{\mathrm{Ls}}\right) + \ldots \\ &P^{m}_{\mathrm{X,Ls}}P^{m}_{\mathrm{X,Rs}}\rho^{m}_{\mathrm{Rs}}\sigma^{l,m}_{\mathrm{Ls}}\sigma^{l,m}_{\mathrm{Rs}}ICC^{l,m}_{1}\cos\left(\phi^{m}_{\mathrm{Rs}}\right)\end{aligned}$$

for $0 \le m < M_{\mathrm{Proc}}, 0 \le l < L$,

and where

$$ICC^{l,m}_{1} = \mathbf{D}_{\mathrm{ICC}}\left(1,l,m\right), \quad ,0 \le m < M_{\mathrm{Proc}}, 0 \le l < L$$

The complex cross-spectrum $\left\langle L_{\mathrm{B}}R^{*}_{\mathrm{B}}\right\rangle^{l,m}$ for each spatial parameter set is estimated for each parameter position and processing band as:

$$\begin{aligned}\left\langle L_{\mathrm{B}}R^{*}_{\mathrm{B}}\right\rangle^{l,m} = &\left(\sigma^{l,m}_{\mathrm{C}}\right)^2 P^{m}_{\mathrm{L,C}}P^{m}_{\mathrm{R,C}}\rho^{m}_{\mathrm{C}}\exp\left(j\phi^{m}_{\mathrm{C}}\right) + \ldots \\ &\left(\sigma^{l,m}_{\mathrm{L}}\right)^2 P^{m}_{\mathrm{L,L}}P^{m}_{\mathrm{R,L}}\rho^{m}_{\mathrm{L}}\exp\left(j\phi^{m}_{\mathrm{L}}\right) + \ldots \\ &\left(\sigma^{l,m}_{\mathrm{R}}\right)^2 P^{m}_{\mathrm{L,R}}P^{m}_{\mathrm{R,R}}\rho^{m}_{\mathrm{R}}\exp\left(j\phi^{m}_{\mathrm{R}}\right) + \ldots \\ &\left(\sigma^{l,m}_{\mathrm{Ls}}\right)^2 P^{m}_{\mathrm{L,Ls}}P^{m}_{\mathrm{R,Ls}}\rho^{m}_{\mathrm{Ls}}\exp\left(j\phi^{m}_{\mathrm{Ls}}\right) + \ldots \\ &\left(\sigma^{l,m}_{\mathrm{Rs}}\right)^2 P^{m}_{\mathrm{L,Rs}}P^{m}_{\mathrm{R,Rs}}\rho^{m}_{\mathrm{Rs}}\exp\left(j\phi^{m}_{\mathrm{Rs}}\right) + \ldots \\ &P^{m}_{\mathrm{L,L}}P^{m}_{\mathrm{R,R}}\sigma^{l,m}_{\mathrm{L}}\sigma^{l,m}_{\mathrm{R}}ICC_{1} + \ldots \\ &P^{m}_{\mathrm{L,Ls}}P^{m}_{\mathrm{R,Rs}}\sigma^{l,m}_{\mathrm{Ls}}\sigma^{l,m}_{\mathrm{Rs}}ICC_{1} + \ldots \\ &P^{m}_{\mathrm{L,Rs}}P^{m}_{\mathrm{R,Ls}}\sigma^{l,m}_{\mathrm{Ls}}\sigma^{l,m}_{\mathrm{Rs}}ICC_{1}\rho^{m}_{\mathrm{BL}}\exp\left(j\left(\phi^{m}_{\mathrm{Rs}} + \phi^{m}_{\mathrm{Ls}}\right)\right) + \ldots \\ &P^{m}_{\mathrm{L,R}}P^{m}_{\mathrm{R,L}}\sigma^{l,m}_{\mathrm{L}}\sigma^{l,m}_{\mathrm{R}}ICC_{1}\rho^{m}_{\mathrm{L}}\exp\left(j\left(\phi^{m}_{\mathrm{R}} + \phi^{m}_{\mathrm{L}}\right)\right)\end{aligned}$$

for $0 \le m < M_{\mathrm{Proc}}, 0 \le l < L$,

and where

$$ICC^{l,m}_{1} = \mathbf{D}_{\mathrm{ICC}}\left(1,l,m\right), \quad ,0 \le m < M_{\mathrm{Proc}}, 0 \le l < L.$$

6.11.4.2 Stereo-based configuration and synthesis

6.11.4.2.1 Introduction

The stereo based binaural decoder can be implemented by a single-slot parametric approach (similar to the mono configurations), or by a multi-slot convolution approach. The stereo based configuration requires a (conventional) stereo (MPEG surround) downmix as input signal $y_{L_0}^{n,k}$, $y_{R_0}^{n,k}$, combined with spatial parameters that enable a 5.1-channel reconstruction. The downmix signal can be a matrixed surround compatible stereo downmix.

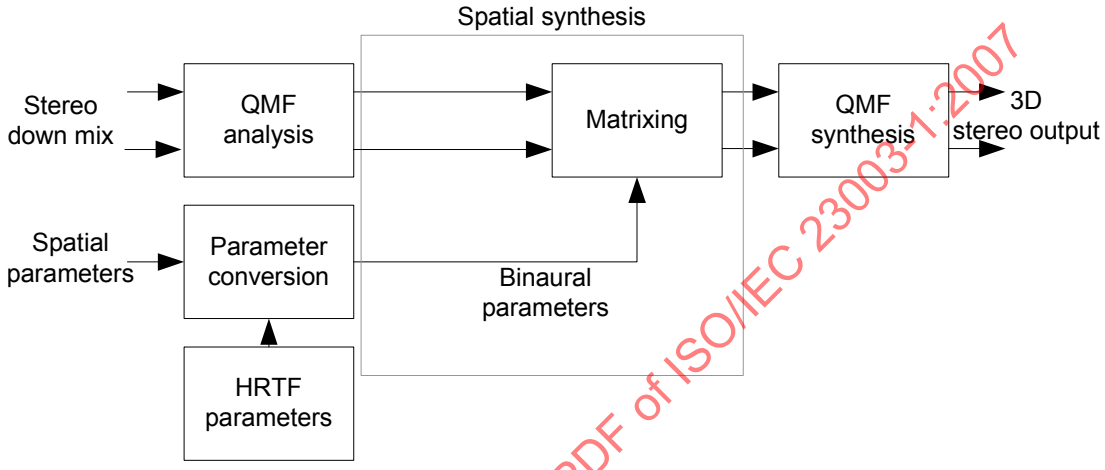

Figure 53 — 3D binaural decoder for '525' spatial configurations

For both configurations (normal downmix or matrixed surround compatible downmix), a spatial synthesis step performs the required signal modifications in the QMF domain, based on both spatial parameters as well as HRTF parameters. Two QMF synthesis filter banks reconstruct the binaural time-domain output signals. The spatial synthesis step comprises a mixing matrix that is applied on the stereo downmix pair according to:

$$\mathbf{y}_{\mathrm{B}}^{n,k} = \begin{bmatrix} y_{L_B}^{n,k} \\ y_{R_B}^{n,k} \end{bmatrix} = \sum_{i=0}^{N_q-1} \mathbf{H}_2^{n-i,k} \mathbf{y}_0^{n-i,k} = \sum_{i=0}^{N_q-1} \begin{bmatrix} \mathbf{h}_{11}^{n-i,k} & \mathbf{h}_{12}^{n-i,k} \\ \mathbf{h}_{21}^{n-i,k} & \mathbf{h}_{22}^{n-i,k} \end{bmatrix} \begin{bmatrix} y_{L_0}^{n-i,k} \\ y_{R_0}^{n-i,k} \end{bmatrix}, \quad 0 \le k < K \ ,$$

with $\mathbf{y}_0^{n,k}$ being the QMF-domain input channels and $\mathbf{y}_{\mathrm{B}}^{n,k}$ being the binaural output channels, k represents the hybrid QMF channel index, and i is the HRTF filter tap index, and n is the QMF slot index.

The mixing matrix $\mathbf{H}_2^{n,k}$ is defined for every time-slot and hybrid subband, as an interpolated version of the $\mathbf{H}_1^{l,k}$ matrix defined for every parameter set and hybrid subband in the following clauses, according to:

$$\mathbf{H}_2^{n,k} = \begin{cases} \mathbf{H}_{\mathrm{temp}}^{l,k}\alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{H}_{\mathrm{temp}}^{-1,k}, & 0 \le n \le \mathbf{t}(l), l = 0 \\ \mathbf{H}_{\mathrm{temp}}^{l,k}\alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{H}_{\mathrm{temp}}^{l-1,k}, & \mathbf{t}(l-1) < n \le \mathbf{t}(l), 1 \le l < L \end{cases}$$

for

$$0 \le k < K$$

where

$$\alpha(n,l)=\begin{cases}\dfrac{n}{\mathbf{t}(l)} & ,l=0\\ \dfrac{n-\mathbf{t}(l-1)}{\mathbf{t}(l)-\mathbf{t}(l-1)} & ,otherwise\end{cases}$$

and where $\mathbf{H}_{\text{temp}}^{l,k}$ can be processed by smoothing according to:

$$\mathbf{H}_{\text{temp}}^{l,k}=\begin{cases}\mathbf{s}_{delta}(l)\mathbf{H}_{1}^{l,k}+(1-\mathbf{s}_{delta}(l))\mathbf{H}_{\text{temp}}^{l-1,k} & ,\mathbf{S}_{proc}(l,\kappa(k))=1\\ \mathbf{H}_{1}^{l,k} & ,\mathbf{S}_{proc}(l,\kappa(k))=0\end{cases}$$

The vector $\mathbf{y}_{\text{B}}^{n,k}$ is subsequently synthesised to the time domain as outlined in subclause 6.3.

The elements of $\mathbf{H}_{1}^{l,k}$ are defined in the following subclauses, differently depending on if a single slot parametric approach is used or if a multi-slot convolution approach is used.

6.11.4.2.2 Single slot parametric low complexity approach

6.11.4.2.2.1 Introduction

If a one tap ($N_q=1$) parametric representation of the HRTFs is used, the mixing matrix $\mathbf{H}_{1}^{l,k}$ is computed on a processing band basis (i.e., 28 bands) instead of a full hybrid-band basis (71 bands). Furthermore, for one tap parametric representation of the HRTFs, the matrix entries are real-valued for processing bands from $M_{\text{proc}}^{\text{c}}$ onwards. The matrix elements are finally mapped to the hybrid band resolution. The HRTF parameters $P_{\text{Y,X}}^{m}$, ϕ_{X}^{m} and ρ_{X}^{m}, for $X\in\{\text{L,R,C,Ls,Rs}\}$ to target $Y\in\{\text{L,R}\}$ are derived from the time-domain HRTF filters, as outlined in e.g. Annex C.1.

6.11.4.2.2.2 Matrix elements

For the single-tap parametric approach six complex gain values are calculated for every parameter set and processing band. The process generates composite HRTF parameters for virtual loudspeaker channel L (being a combination of FL and BL), R (a combination of FR and BR) and C (a combination of centre and LFE). The subscript used in the following refers to the binaural output channel (L/R) followed by the virtual loudspeaker channel.

The relative powers of front vs. surround pairs (e.g., $\left(\sigma_{\text{L}}^{l,k}\right)^2/\left(\sigma_{\text{Ls}}^{l,k}\right)^2$ and $\left(\sigma_{\text{R}}^{l,k}\right)^2/\left(\sigma_{\text{Rs}}^{l,k}\right)^2$) are reconstructed on the basis of transmitted CLD parameters $\mathbf{D}_{\text{CLD}}(1,l,m)$ and $\mathbf{D}_{\text{CLD}}(2,l,m)$:

$$\left(\sigma_{\text{L}}^{l,m}\right)^2=r_1\left(CLD_1^{l,m}\right)\quad\left(\sigma_{\text{Ls}}^{l,m}\right)^2=r_2\left(CLD_1^{l,m}\right)/g_s^2$$

$$\left(\sigma_{\text{R}}^{l,m}\right)^2=r_1\left(CLD_2^{l,m}\right)\quad\left(\sigma_{\text{Rs}}^{l,m}\right)^2=r_2\left(CLD_2^{l,m}\right)/g_s^2$$

with g_s the down mix gain for surround channels, and where

$$r_1(CLD) = \frac{10^{CLD/10}}{1+10^{CLD/10}}, \quad r_2(CLD) = \frac{1}{1+10^{CLD/10}},$$

for $1 \le X < 3$, $0 \le m < M_{\text{Proc}}$, $0 \le l < L$ and where

$$CLD_X^{l,m} = \mathbf{D}_{\text{CLD}}(X,l,m),,$$

Given the relative powers calculated above, the matrix elements are defined according to:

$$h_{\text{L,C}}^{l,m} = P_{\text{L,C}}^{m} e^{+j\phi_{\text{C}}^{m}/2} \frac{1}{g_c}$$

$$h_{\text{R,C}}^{l,m} = P_{\text{R,C}}^{m} e^{+j\phi_{\text{C}}^{m}/2} \frac{1}{g_c}$$

$$h_{\text{L,L}}^{l,m} = \sqrt{\left(\sigma_{\text{L}}^{l,m}\right)^2 \left(P_{\text{L,L}}^{m}\right)^2 + \left(\sigma_{\text{Ls}}^{l,m}\right)^2 \left(P_{\text{L,Ls}}^{m}\right)^2},$$

$$h_{\text{R,L}}^{l,m} = e^{-j\left(w_{\text{L}}^{l,m}\phi_{\text{L}}^{m} + w_{\text{Ls}}^{l,m}\phi_{\text{Ls}}^{m}\right)} \sqrt{\left(\sigma_{\text{L}}^{l,m}\right)^2 \left(P_{\text{R,L}}^{m}\right)^2 + \left(\sigma_{\text{Ls}}^{l,m}\right)^2 \left(P_{\text{R,Ls}}^{m}\right)^2}$$

$$h_{\text{L,R}}^{l,m} = e^{j\left(w_{\text{R}}^{l,m}\phi_{\text{R}}^{m} + w_{\text{Rs}}^{l,m}\phi_{\text{Rs}}^{m}\right)} \sqrt{\left(\sigma_{\text{R}}^{l,m}\right)^2 \left(P_{\text{L,R}}^{m}\right)^2 + \left(\sigma_{\text{Rs}}^{l,m}\right)^2 \left(P_{\text{L,Rs}}^{m}\right)^2},$$

$$h_{\text{R,R}}^{l,m} = \sqrt{\left(\sigma_{\text{R}}^{l,m}\right)^2 \left(P_{\text{R,R}}^{m}\right)^2 + \left(\sigma_{\text{Rs}}^{l,m}\right)^2 \left(P_{\text{R,Rs}}^{m}\right)^2}$$

where $w_{\text{L}}^{l,m} = \frac{\left(\sigma_{\text{L}}^{l,m}\right)^2}{\left(\sigma_{\text{L}}^{l,m}\right)^2 + \left(\sigma_{\text{Ls}}^{l,m}\right)^2}$, $w_{\text{Ls}}^{l,m} = \frac{\left(\sigma_{\text{Ls}}^{l,m}\right)^2}{\left(\sigma_{\text{L}}^{l,m}\right)^2 + \left(\sigma_{\text{Ls}}^{l,m}\right)^2}$, $w_{\text{R}}^{l,m} = \frac{\left(\sigma_{\text{R}}^{l,m}\right)^2}{\left(\sigma_{\text{R}}^{l,m}\right)^2 + \left(\sigma_{\text{Rs}}^{l,m}\right)^2}$, $w_{\text{Rs}}^{l,m} = \frac{\left(\sigma_{\text{Rs}}^{l,m}\right)^2}{\left(\sigma_{\text{R}}^{l,m}\right)^2 + \left(\sigma_{\text{Rs}}^{l,m}\right)^2}$ and where g_c is the down mix gain for the centre channel.

In the equations above, the complex phase angles ϕ_{X}^{m} are only applied for low-frequency parameter bands ($0 \le m < M_{\text{proc}}^{\text{c}}$); for higher processing bands the phase angles are assumed to be zero.

Given the matrix elements above the $\mathbf{H}_1^{l,k}$ matrix is given by:

$$\mathbf{H}_1^{l,k} = \begin{bmatrix} \mathbf{h}_{11}^{l,k} & \mathbf{h}_{12}^{l,k} \\ \mathbf{h}_{21}^{l,k} & \mathbf{h}_{22}^{l,k} \end{bmatrix} = \begin{bmatrix} h_{\text{L,L}}^{l,\kappa(k)} & h_{\text{L,R}}^{l,\kappa(k)} & h_{\text{L,C}}^{l,\kappa(k)} \\ h_{\text{R,L}}^{l,\kappa(k)} & h_{\text{L,R}}^{l,\kappa(k)} & h_{\text{R,C}}^{l,\kappa(k)} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{W}_{\text{temp}}^{l,\kappa(k)}, \quad 0 \le k < K, 0 \le l < L,$$

where $\kappa(k)$ is given by Table A.31, and maps the hybrid band k to the corresponding processing band, and where $\mathbf{W}_{\text{temp}}$ is the upmix matrix as defined in subclause 6.5.3. Hence, only the first three rows of $\mathbf{W}_{\text{temp}}$ are used to generate $\mathbf{H}_1^{l,k}$.

6.11.4.2.3 Multi-slots HRTF convolution approach

6.11.4.2.3.1 Introduction

In the following the application of a multi-slot HRTF processing is defined. The processing requires a complex QMF representation of the time-domain HRTF filters as outlined in Annex B. The ten HRTF filters are combined into four HRTF filters, similar to the parametric HRTF processing. However, the filters are morphed as a function of the spatial parameters, phase compensation factors, and gain adjustment factors. The processing is carried out on the hybrid frequency resolution, and the filters are calculated for every parameter set and interpolated in time to the time-resolution of the QMF, i.e. every time-slot.

6.11.4.2.3.2 Mapping of filters to the hybrid subband resolution

The ten complex subband filters $\hat{\mathbf{v}}_{\mathrm{Y,X}}$ ($X \in \{L, R, C, Ls, Rs\}$ to target $Y \in \{L, R\}$) derived from the time-domain HRTF impulse responses as outlined in Annex B have components

$$\left(\hat{v}_{\mathrm{Y,X}}\right)_m (n)$$

for QMF subband $m = 0,1,\ldots,63$ and QMF time slot $n = 0,1,\ldots,N_q - 1$. The index mapping $q(m)$, from the hybrid band k to QMF band m is given by Table A.32, and thus the HRTF filters $\mathbf{v}^k_{\mathrm{Y,X}}$ in the hybrid band domain are defined by

$$\left(v_{\mathrm{Y,X}}\right)_k (n) = \left(\hat{v}_{\mathrm{Y,X}}\right)_{q(k)} (n) .$$

6.11.4.2.3.3 Estimation of HRTF parameters for filter combination

The morphing of the HRTF filters require a time-invariant phase compensated cross correlation $\left(ICCFB^{\phi}_{\mathrm{Y,X}}\right)_k$, and an average front/back level quotient $CFB^k_{\mathrm{Y,X}}$,in order not to get unwanted colorization during the morphing process. These are time-invariant and defined once for the selected HRTF filters.

The average front/back level quotient per hybrid band for the HRTF filters is defined for $Y \in \{L, R\}$ and $X \in \{L, R\}$ by

$$CFB^k_{\mathrm{Y,X}} = \sqrt{\frac{\sum_{n=0}^{N_q-1} \left|\left(v_{\mathrm{Y,FX}}\right)_k (n)\right|^2}{\sum_{n=0}^{N_q-1} \left|\left(v_{\mathrm{Y,BX}}\right)_k (n)\right|^2}}, \quad 0 \le k < K .$$

Furthermore, phase parameters $\phi^{\mathrm{L}}_{\mathrm{L,Ls}}, \phi^{\mathrm{L}}_{\mathrm{R,Rs}}, \phi^{\mathrm{R}}_{\mathrm{L,Ls}}, \phi^{\mathrm{R}}_{\mathrm{R,Rs}}$ are then defined for $Y \in \{L, R\}$ and $X \in \{L, R\}$ by

$$\left(CIC_{Y,X}\right)_k = \left|\left(CIC_{Y,X}\right)_k\right| \exp\left(j\left(\phi^Y_{X,Xs}\right)_k\right),$$

where the complex cross correlations $\left(CIC_{Y,X}\right)_k$ are defined by

$$\left(CIC_{Y,X}\right)_k = \frac{\sum_{l=0}^{L_q-1}\left(v_{Y,FX}\right)_k(l)\left(v_{Y,BX}\right)_k^*(l)}{\left(\sum_{l=0}^{L_q-1}\left|\left(v_{Y,FX}\right)_k(l)\right|^2\right)^{1/2}\left(\sum_{l=0}^{L_q-1}\left|\left(v_{Y,BX}\right)_k(l)\right|^2\right)^{1/2}}.$$

A phase unwrapping is applied to the phase parameters along the subband index k, such that the absolute value of the phase increment from subband k to subband $k+1$ is smaller or equal to π for $k=0,1,\ldots$. In cases where there are two choices, $\pm\pi$,for the increment, the sign of the increment for a phase measurement in the interval $]-\pi,\pi]$ is chosen.

Finally, normalized phase compensated cross correlations are defined for $Y\in\{\mathrm{L,R}\}$ and $X\in\{\mathrm{L,R}\}$ by

$$\left(ICCFB^{\phi}_{Y,X}\right)_k = \left|\left(CIC_{Y,X}\right)_k\right|.$$

6.11.4.2.3.4 Matrix elements

In the case where $N_q>1$, the six complex gains are replaced with six filters according to $\mathbf{h}_{Y,X}$ for $Y\in\{\mathrm{L,R}\}$ and $X\in\{\mathrm{L,R,C}\}$. These filters are derived from the ten filters $\mathbf{v}^k_{\mathrm{Y,X}}$ for $Y\in\{\mathrm{L,R}\}$ and $X\in\{\mathrm{L,R,C,Ls,Rs}\}$, which describe the given HRTF filter responses in the hybrid QMF domain.

The relative powers of front vs. surround pairs (e.g., $\left(\sigma^{l,k}_{\mathrm{L}}\right)^2\Big/\left(\sigma^{l,k}_{\mathrm{Ls}}\right)^2$ and $\left(\sigma^{l,k}_{\mathrm{R}}\right)^2\Big/\left(\sigma^{l,k}_{\mathrm{Rs}}\right)^2$) is similarly to the single slot case reconstructed on the basis of transmitted CLD parameters, $\mathbf{D}_{\mathrm{CLD}}(1,l,m)$ and $\mathbf{D}_{\mathrm{CLD}}(2,l,m)$, albeit mapped from the parameter band resolution to the hybrid filterbank resolution, according to:

$$\left(\sigma^{l,k}_{\mathrm{L}}\right)^2 = r_1\left(CLD_1^{l,\kappa(k)}\right) \qquad \left(\sigma^{l,k}_{\mathrm{Ls}}\right)^2 = r_2\left(CLD_1^{l,\kappa(k)}\right)\Big/g_s^2$$

$$\left(\sigma^{l,k}_{\mathrm{R}}\right)^2 = r_1\left(CLD_2^{l,\kappa(k)}\right) \qquad \left(\sigma^{l,k}_{\mathrm{Rs}}\right)^2 = r_2\left(CLD_2^{l,\kappa(k)}\right)\Big/g_s^2$$

with g_s the down mix gain for surround channels, and where

$$r_1(CLD) = \frac{10^{CLD/10}}{1+10^{CLD/10}}, \qquad r_2(CLD) = \frac{1}{1+10^{CLD/10}},$$

and where

$$CLD_X^{l,\kappa(k)} = \mathbf{D}_{\mathrm{CLD}}\left(X,l,\kappa(k)\right), \quad 1\le X<3, 0\le k<K, 0\le l<L,$$

and where $\kappa(k)$ is given by Table A.31, and maps the hybrid band k to the corresponding processing band.

The combination of the front and surround channel filters is performed with a complex linear combination based on the relative powers of front vs. surround pairs, phase compensation factors and gain compensation factors, according to

$$\mathbf{h}_{\mathrm{L,C}}^{k} = \mathbf{v}_{\mathrm{L,C}}^{k}$$

$$\mathbf{h}_{\mathrm{R,C}}^{k} = \mathbf{v}_{\mathrm{R,C}}^{k}$$

$$\mathbf{h}_{\mathrm{L,L}}^{l,k} = g_{\mathrm{L,L}}^{l,k}\sigma_{\mathrm{L}}^{l,k}\exp\left(-j\left(\phi_{\mathrm{L,Ls}}^{\mathrm{L}}\right)_{k}\left(\sigma_{\mathrm{Ls}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{L,L}}^{k} + g_{\mathrm{L,L}}^{l,k}\sigma_{\mathrm{Ls}}^{l,k}\exp\left(j\left(\phi_{\mathrm{L,Ls}}^{\mathrm{L}}\right)_{k}\left(\sigma_{\mathrm{L}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{L,Ls}}^{k}$$

$$\mathbf{h}_{\mathrm{L,R}}^{l,k} = g_{\mathrm{L,R}}^{l,k}\sigma_{\mathrm{R}}^{l,k}\exp\left(-j\left(\phi_{\mathrm{R,Rs}}^{\mathrm{L}}\right)_{k}\left(\sigma_{\mathrm{Rs}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{L,R}}^{k} + g_{\mathrm{L,R}}^{l,k}\sigma_{\mathrm{Rs}}^{l,k}\exp\left(j\left(\phi_{\mathrm{R,Rs}}^{\mathrm{L}}\right)_{k}\left(\sigma_{\mathrm{R}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{L,Rs}}^{l,k}$$

$$\mathbf{h}_{\mathrm{R,L}}^{l,k} = g_{\mathrm{R,L}}^{l,k}\sigma_{\mathrm{L}}^{l,k}\exp\left(-j\left(\phi_{\mathrm{L,Ls}}^{\mathrm{R}}\right)_{k}\left(\sigma_{\mathrm{Ls}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{R,L}}^{k} + g_{\mathrm{R,L}}^{l,k}\sigma_{\mathrm{Ls}}^{l,k}\exp\left(j\left(\phi_{\mathrm{L,Ls}}^{\mathrm{R}}\right)_{k}\left(\sigma_{\mathrm{L}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{R,Ls}}^{k}$$

$$\mathbf{h}_{\mathrm{R,R}}^{l,k} = g_{\mathrm{R,R}}^{l,k}\sigma_{\mathrm{R}}^{l,k}\exp\left(-j\left(\phi_{\mathrm{R,BR}}^{\mathrm{R}}\right)_{k}\left(\sigma_{\mathrm{BR}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{R,R}}^{k} + g_{\mathrm{L,R}}^{l,k}\sigma_{\mathrm{BR}}^{l,k}\exp\left(j\left(\phi_{\mathrm{R,Rs}}^{\mathrm{R}}\right)_{k}\left(\sigma_{\mathrm{R}}^{l,k}\right)^{2}\right)\mathbf{v}_{\mathrm{R,Rs}}^{k}$$

for $0 \le k < K$ and $0 \le l < L$.

The phase parameters $\left(\phi_{\mathrm{L,Ls}}^{\mathrm{L}}\right)_{k}, \left(\phi_{\mathrm{R,Rs}}^{\mathrm{L}}\right)_{k}, \left(\phi_{\mathrm{L,Ls}}^{\mathrm{R}}\right)_{k}, \left(\phi_{\mathrm{R,Rs}}^{\mathrm{R}}\right)_{k}$ are defined in the subclause above. The gain factors $g_{\mathrm{L,L}}^{l,k}, g_{\mathrm{L,R}}^{l,k}, g_{\mathrm{R,L}}^{l,k}, g_{\mathrm{R,R}}^{l,k}$ are determined by

$$g_{\mathrm{Y,X}}^{l,k} = \left(\frac{\left(\sigma_{\mathrm{X}}^{l,k}\right)^{2}\left(CFB_{\mathrm{Y,X}}^{k}\right)^{2} + \left(\sigma_{\mathrm{Xs}}^{l,k}\right)^{2}}{\left(\sigma_{\mathrm{X}}^{l,k}\right)^{2}\left(CFB_{\mathrm{Y,X}}^{k}\right)^{2} + \left(\sigma_{\mathrm{Xs}}^{l,k}\right)^{2} + 2\sigma_{\mathrm{X}}^{l,k}\sigma_{\mathrm{Xs}}^{l,k}CFB_{\mathrm{Y,X}}^{k}\left(ICCFB_{\mathrm{Y,X}}^{\phi}\right)_{k}}\right)^{1/2},$$

where the parameters $CFB_{\mathrm{Y,X}}^{k}$ and $\left(ICCFB_{\mathrm{Y,X}}^{\phi}\right)_{k}$ are defined in the subclause above.

Hence, the matrix $\mathbf{H}_{1}^{l,k}$ is three-dimensional, defined for every parameter-set l and hybrid subband k, where every element is a vector of length N_q, according to

$$\mathbf{H}_{1}^{l,k} = \begin{bmatrix} \mathbf{h}_{11}^{l,k} & \mathbf{h}_{12}^{l,k} \\ \mathbf{h}_{21}^{l,k} & \mathbf{h}_{22}^{l,k} \end{bmatrix} = \begin{bmatrix} g_{\mathrm{L}}^{l,k} & 0 \\ 0 & g_{\mathrm{R}}^{l,k} \end{bmatrix} \begin{bmatrix} \mathbf{h}_{\mathrm{L,L}}^{l,k} & \mathbf{h}_{\mathrm{L,R}}^{l,k} & \mathbf{h}_{\mathrm{L,C}}^{l,k} \\ \mathbf{h}_{\mathrm{R,L}}^{l,k} & \mathbf{h}_{\mathrm{R,R}}^{l,k} & \mathbf{h}_{\mathrm{R,C}}^{l,k} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{W}_{\mathrm{temp}}^{l,k}, \quad 0 \le k < K, 0 \le l < L$$

where $\mathbf{W}_{\mathrm{temp}}$ is the upmix matrix as defined in subclause 6.5.3. Hence, only the first three rows of $\mathbf{W}_{\mathrm{temp}}$ are used to generate $\mathbf{H}_{1}^{l,k}$. The gain factors $g_{\mathrm{R}}^{l,k}, g_{\mathrm{L}}^{l,k}$ are determined by the parameters $\alpha^{l,\kappa(k)}, \beta^{l,\kappa(k)}, \gamma^{l,\kappa(k)}$ of subclause 6.5.2.2.3.2 via the following steps. The parameters $\alpha^{l,\kappa(k)}, \beta^{l,\kappa(k)}, \gamma^{l,\kappa(k)}$ are given per processing band, and are hence mapped to the hybrid band frequency resolution by means of $\kappa(k)$ as given by Table A.31.

First, intermediate coefficients $a^{l,k}, b^{l,k}, c^{l,k}$ are defined,

$$a^{l,k} = (1 - \alpha^{l,\kappa(k)})/3, \quad b^{l,k} = (1 - \beta^{l,\kappa(k)})/3, \quad c^{l,k} = a^{l,k} + b^{l,k}.$$

Second, energy estimates are formed for $Y \in \{\mathrm{L},\mathrm{R}\}$ according to

$$E_{\mathrm{Y}}^{l,k} = b^{l,k}\left(1-c^{l,k}\right)\left\|\mathbf{h}_{\mathrm{Y,L}}^{l,k}\right\|^2 + a^{l,k}\left(1-c^{l,k}\right)\left\|\mathbf{h}_{\mathrm{Y,R}}^{l,k}\right\|^2 + a^{l,k}b^{l,k}\left\|\mathbf{h}_{\mathrm{Y,C}}^{l,k}\right\|^2,$$

$$\Delta E_{\mathrm{Y}}^{l,k} = a^{l,k}b^{l,k}\left(1-c^{l,k}\right)\left\|\mathbf{h}_{\mathrm{Y,L}}^{l,k} + \mathbf{h}_{\mathrm{Y,R}}^{l,k} - \mathbf{h}_{\mathrm{Y,C}}^{l,k}\right\|^2,$$

with the notation $\|\mathbf{h}\|^2 = \sum_{l=0}^{L_q-1}|h(l)|^2$. The gain factors are then given for $Y \in \{\mathrm{L},\mathrm{R}\}$ by

$$g_{\mathrm{Y}}^{l,k} = \begin{cases} \dfrac{1}{\gamma^{l,\kappa(k)}} \cdot \min\left\{2, \sqrt{\dfrac{E_{\mathrm{Y}}^{l,k}+\varepsilon}{E_{\mathrm{Y}}^{l,k}-\Delta E_{\mathrm{Y}}^{l,k}+\varepsilon}}\right\}, & \text{if } a^{l,k}>0, b^{l,k}>0, c^{l,k}<1; \\ 1, & \text{otherwise.} \end{cases}$$

6.11.5 3D decoder for 5.1 channel reconstruction

In the 3D decoder, the stereo (3D) down mix is processed by means of a '3D inverse' operation, which generates the conventional stereo down mix from the 3D down mix by means of 2x2 matrixing operation $\mathbf{Q}$. The processing structure is shown in Figure 54.

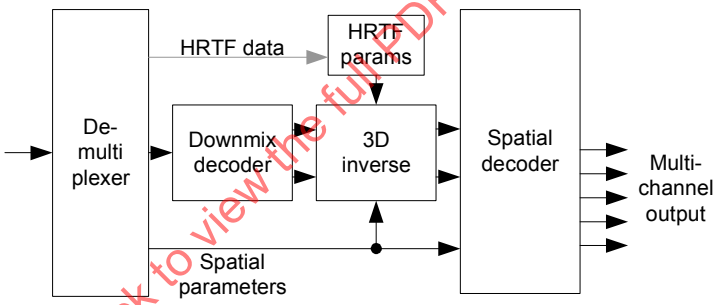

Figure 54 — 3D inversion (decoder)

The matrix operation $\mathbf{Q}$ should not be applied individually but can be merged in the subsequent matrix operations of the spatial decoder. Specifically, the signals are processed by a 3x2 matrix $\mathbf{M}_1$ when entering the spatial decoder. For 3D downmix inversion, the matrix $\mathbf{M}_1'$ of the spatial decoder that should be applied comprises the product of the 3D inversion matrix and the original matrix $\mathbf{M}_1$:

$$\mathbf{M}_1' = \mathbf{M}_1\mathbf{Q}.$$

The processing matrix $\mathbf{Q}$ is derived from the spatial parameters and the HRTF parameters. $\mathbf{Q}$ is the inverse matrix of the encoder processing matrix $\mathbf{H}$:

$$\mathbf{Q}^{n,k} = \mathbf{Q}_{tmp}^{l,k}\alpha(n,l) + \left(1-\alpha(n,l)\right)\mathbf{Q}_{tmp}^{l-1,k} \qquad , \ \mathbf{t}(l-1) < n < \mathbf{t}(l)$$

for $0 \le k < K$, $0 \le l < L$ and where $\mathbf{Q}_{tmp}^{-1,k}$ is the last datapoint from the previous frame. Furthermore:

$$\alpha(n,l) = \begin{cases} \dfrac{n}{\mathbf{t}(l)} & ,l = 0 \\ \dfrac{n - \mathbf{t}(l-1)}{\mathbf{t}(l) - \mathbf{t}(l-1)} & ,otherwise \end{cases}$$

and where the

$$\mathbf{Q}_{\text{tmp}}^{l,k} = \left(\mathbf{H}^{l,k}\right)^{-1} = \begin{bmatrix} q_{11}^{l,k} & q_{12}^{l,k} \\ q_{21}^{l,k} & q_{22}^{l,k} \end{bmatrix}$$

with

$$q_{11}^{l,k} = \frac{h_{22}^{l,k}}{N},\quad q_{22}^{l,k} = \frac{h_{11}^{l,k}}{N},\quad q_{12}^{l,k} = \frac{-h_{12}^{l,k}}{N},\quad q_{21}^{l,k} = \frac{-h_{21}^{l,k}}{N},$$

and

$$N = h_{11}^{l,k} h_{22}^{l,k} - h_{12}^{l,k} h_{21}^{l,k}.$$

The matrix element $h_{11}^{l,k}, h_{22}^{l,k}, h_{12}^{l,k}, h_{21}^{l,k}$ are defined as in subclause 6.11.4.2.2 with the exception that for the 3D decoder the HRTF parameters $P_{Y,X}^{m}$, ϕ_X^m and ρ_X^m can be transmitted in the bitstream and are hence defined dependant on the bitstream elements as:

$$P_{L,X}^{m} = 10^{\frac{\mathbf{bsHRTFlevelleft}\left(X, mapfunc\left(m, M_{\text{HRTF}}\right)\right) - 16}{20}}$$

$$P_{\text{R},X}^{m} = \begin{cases} P_{\text{L},i(X)}^{m} & ,if\ bsHRFTasymmetric = 0 \\ 10^{\frac{\mathbf{bsHRTFlevelright}\left(X, mapfunc\left(m, M_{\text{HRTF}}\right)\right) - 46}{20}} & ,if\ bsHRFTasymmetric = 1 \end{cases}$$

$$\phi_X^m = \begin{cases} \dfrac{\pi}{32} \cdot \mathbf{bsHRTFphaseLR}\left(X, mapfunc\left(m, M_{\text{HRTF}}\right)\right) & ,if\ \mathbf{bsHRTFphase}(X) = 1,\ m < 13 \\ \boldsymbol{\varphi}_{\text{default}}(X,m) & ,if\ \mathbf{bsHRTFphase}(X) = 0,\ m < 13 \\ 0 & ,m \ge 13 \end{cases}$$

$$\rho_X^m = \begin{cases} deq\left(\mathbf{bsHRTFiccLR}\left(X, mapfunc\left(m, M_{\text{HRTF}}\right)\right), ICC\right) & ,if\ \mathbf{bsHRTFicc}(X) = 1 \\ \boldsymbol{\rho}_{\text{default}}(X,m) & ,if\ \mathbf{bsHRTFicc}(X) = 0 \end{cases}$$

for $0 \le m < M_{\text{proc}}$, $M_{\text{HRTF}} = HRTFnumBands$ and where the channel order in the bitstream is $X \in \{\text{C,R,Rs,Ls,L}\}$ (clockwise starting with the centre channel) and $i(X)$ in defined in Table 106. The dequantization uses the fine ICC dequantization table and hence the function *deq*() is defined in subclause 6.1.8. The function *mapfunc*() maps from the transmitted frequency resolution M_{HRTF} to 28 processing bands. This function is defined in Table 86. The default tables $\boldsymbol{\varphi}_{\text{default}}(X,m)$ and $\boldsymbol{\rho}_{\text{default}}(X,m)$ are defined in Table A.34 and Table A.35.

Table 106 — Symmetric HRTF channel mapping $i(X)$

X	$i(X)$
C	C
R	L
Rs	Ls
Ls	Rs
L	R

6.11.6 3D decoder for personalized HRTFs

The 3D decoder using personalized HRTFs is shown in Figure 55. The down mix is first processed with an inverse 3D stage. The inversion process is controlled by the transmitted spatial parameters and the HRTF parameters used to create the 3D down mix (referred to as HRTF parameter set 'B'). Subsequently, the 3D-inverted down mix is processed by a 3D synthesis stage using (personalized) HRTF parameter set 'P'. Since both 3D inversion using HRTF parameters 'B' as well as 3D synthesis using HRTF parameters 'P' comprises a 2x2 matrix operation, the 3D inversion and the 3D synthesis process can be combined into a single 2x2 matrix operation $\mathbf{H}'_2$, which then replaces $\mathbf{H}_2$ in the synthesis stage of subclause 6.11.4.2 since, $\mathbf{H}'_2$ consists of the matrix product of the inversion and 3D synthesis matrices.

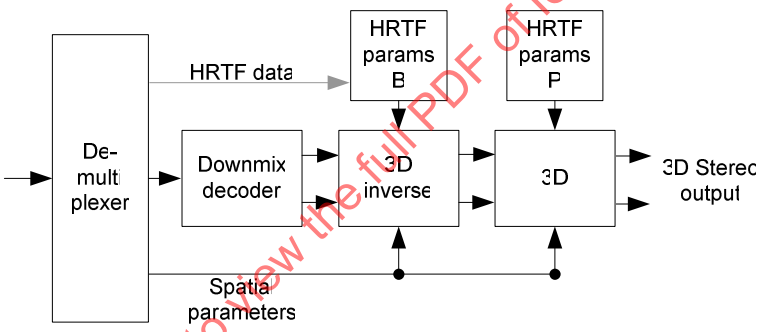

Figure 55 — 3D decoder for personalized HRTFs

If the 3D inverse matrix using HRTF parameter 'B' set is denoted by $\mathbf{Q}_B$ (comprising the inverse of the encoder-side 3D processing matrix $\mathbf{H}_B$), and the 3D synthesis matrix using HRTF parameter set 'P' is given by $\mathbf{H}_P$, the 2x2, 3D binaural decoder processing matrix $\mathbf{H}'_2$ is hence defined as:

$$\mathbf{H}'_2 = \mathbf{H}_{2P}\mathbf{Q}_B ,$$

Where $\mathbf{Q}_B$ as described in subclause 6.11.5. using the parameter set 'B' and where $\mathbf{H}_{2P}$ is defined in subclause 6.11.4.2 utilizing either a single slot parametric HRTF or multi-slots HRTF for the personalized HRTF set 'P'.

7 Transport of Spatial Audio Side Information

7.1 Overview

Due to its basic principle of operation, MPEG Surround technology requires two data streams for its operation, namely a stream of downmix data (the coded downmix audio signal) and as side information a stream of spatial data (the parametric spatial audio data guiding the MPEG Surround decoding process). Depending on the application scenario, both data streams may or may not be conveyed to the MPEG Surround decoder as part of a single stream, and may or may not be conveyed in a synchronized way. The subsequent subclauses define mechanisms for the transport of spatial data for MPEG Surround decoding for different environments employing MPEG Audio and Systems to convey a coded downmix audio signal.

7.2 Transport and Signalling in an MPEG Environment

7.2.1 Time alignment of spatial frames and downmix coder frames

The spatial frame length is preferred to be an integer multiple of the frame length of the underlying downmix coder. Asynchronous framing of spatial data and the downmix data (i.e., different frame lengths) is possible. However, in this case, additional buffering of the spatial data in the decoder might be needed.

In general spatial data is conveyed in such a manner that it is available to the MPEG Surround decoder in time when it is required to process the decoded downmix signals, assuming the most efficient connection of downmix decoder to the MPEG Surround decoder. This is a direct connection of HE-AAC and MPEG Surround in the QMF domain in case of MPEG Surround using normal operation (as opposed to upsampled or downsampled operation as defined in subclause 6.3.3), and a connection in the PCM time domain in all other cases. When HE-AAC and MPEG Surround are connected in the time domain even though the most efficient connection would have been in the QMF domain, the spatial parameters have to be delayed accordingly in order to maintain the time alignment between spatial data and downmix data. Information about this delay is given in subclause 6.4.1.

In the case that the spatial data is embedded in the downmix data stream (see subclause 7.2.2, 7.2.3, and 7.2.4), the temporal relationship between spatial frames and downmix frames is indicated by the value of sacTimeAlign (see subclause 7.2.5).

In the case that the downmix data and the spatial data are conveyed in separate streams, the temporal relationship between spatial frames and downmix frames is indicated by the time stamps of the corresponding streams. If the transport layer does not provide time stamps (as e.g. in case of LATM), the transport layer needs to define the temporal relationship between the data of these both streams by other means.

7.2.2 Transport and Signalling in an MPEG-4 Audio/Systems Environment

The transmission of spatial audio data requires a spatial elementary stream that depends on the elementary stream containing the related coded audio downmix data. The actual spatial data is either conveyed in the spatial elementary stream or multiplexed into the downmix data stored in the elementary stream upon which the spatial elementary stream depends. The latter is specified for MPEG-2/4 AAC payloads (see subclause 7.2.3) and for MPEG-1/2 Layer I/II/III payloads (see subclause 7.2.4).

Backwards compatibility with decoders that can decode the coded audio downmix data but not the spatial audio data is achieved in both scenarios.

If the downmix signal is encoded with the combination of a mono AAC downmix coder and SBR bandwidth extension, it is possible that both data for the MPEG-4 Parametric Stereo (PS) tool as well as data for MPEG Surround in a 5-1-5 configuration is present simultaneously in the bitstream conveying the downmix signal. Such a bitstream can be decoded into a 2 channel stereo signal in accordance with the MPEG-4 HE-AAC v2

Profile, whereby the MPEG Surround data is ignored. When such a bitstream is used in combination with an MPEG Surround decoder, the PS data in the bitstream is ignored and the downmix bitstream is decoded in accordance with the MPEG-4 HE-AAC profile. This provides a mono downmix signal in the QMF domain that is used as input to the subsequent MPEG Surround decoding process for a 5-1-5 configuration as described in subclause 6.4.2.

The interface to ISO/IEC 14496-1 is in line with the specification given in ISO/IEC 14496-3 subclause 1.6. An elementary stream carrying spatial audio data is identified by the Audio Object Type “MPEG Surround” (Object Type ID 30). The AudioSpecificConfig() for this object carries the SpatialSpecificConfig() data and a sacPayloadEmbedding flag that indicates whether the SpatialFrame() payload is conveyed as an elementary stream or embedded into the downmix data, as defined in ISO/IEC 14496-3 subclause 1.6.3.14.

7.2.3 Embedding spatial audio data in MPEG-2/4 AAC payloads

Spatial audio data can be conveyed in the AAC extension_payload() mechanism using extension_type EXT_SAC_DATA (“1100”), as defined in ISO/IEC 13818-7 subclause 8.8 and ISO/IEC 14496-3 subclause 4.5.2.9. The extension_payload() for type EXT_SAC_DATA is used to carry a SacDataFrame(), complete or split into several fragments, using the same syntax elements ancType, ancStart, and ancStop as defined in the next subclause.

This mechanism is backward compatible with existing AAC decoders and provides implicit signalling. Explicit signalling is possible in an MPEG-4 environment if the SpatialSpecificConfig() is conveyed out-of-band, as e.g. described in the previous subclause. In that case, any in-band SpatialSpecificConfig() shall be identical to the out-of-band one. There must be at least one extension_payload() element with extension_type==EXT_SAC_DATA in each AAC frame in order to enable immediate implicit signalling. An sac_extension_data() element can have an empty payload, i.e., cnt==1.

7.2.4 Embedding spatial audio data in MPEG-1/2 Layer I/II/III bitstreams

Spatial audio data can be inserted as ancillary data into an MPEG-1/2 Layer I/II/III bitstream as defined in ISO/IEC 11172-3 and ISO/IEC 13818-3. The AncDataElement() is conveyed in the ancillary data part of an MPEG-1/2 frame. It does not have to be located immediately after the audio_data of that frame. The first bit of the ancSyncword must be byte-aligned with respect to the first bit of the 0xFFF syncword of the MPEG-1/2 frame header. The AncDataElement() must be completely included in the ancillary data of a single MPEG-1/2 frame. There must be at least one AncDataElement() in the ancillary data of each MPEG-1/2 frame in order to enable immediate implicit signalling. An AncDataElement() can have an empty payload, i.e., ancLenBytes==0. If there is more than one AncDataElement() in the ancillary data of an MPEG-1/2 frame, all these AncDataElements() must directly follow each other.

Table 107 — Syntax of AncDataElement()

Syntax	No. of bits	Mnemonic
AncDataElement()		
{		
ancSyncword; /* 0x8E4 */	**12**	**bslbf**
ancType;	**2**	**uimsbf**
ancStart;	**1**	**uimsbf**
ancStop;	**1**	**uimsbf**
cnt = **ancLenBytes**;	**8**	**uimsbf**
if (cnt==255) {		
cnt += **ancLenBytesAdd**;	**16**	**uimsbf**
}		
ancCrcWord;	**8**	**uimsbf**
for (i=0; i<cnt; i++) {		
ancDataSegmentByte[i];	**8**	**bslbf**
}		
}		

ancSyncword Identification syncword. Shall be 0x8E4.

ancType Indicates type of ancillary data, see following table:

Table 108 — ancType

ancType	**Meaning**
0x0	SacDataFrame(0) (MPEG Surround frame)
0x1	SacDataFrame(1) (MPEG Surround header and MPEG Surround frame)
0x2..0x3	(reserved)

ancStart Indicates if data segment begins a data block.

ancStop Indicates if data segment ends a data block.

ancLenBytes Number of bytes in data segment.

ancLenBytesAdd Additional number of bytes in data segment, needed if the data segments contains 255 or more bytes.

ancCrcWord ancCrcWord is defined by the generator polynomial G(x) = x^8+x^2+x+1 and the initial value for the CRC calculation is 0xFF. The CRC covers all bits in the AncDataElement() excluding ancSyncword and the ancCrcWord itself.

ancDataSegmentByte The concatenation of all ancDataSegmentByte from consecutive AncDataElement(), starting from the AncDataElement with ancStart==1 up to and including the AncDataElement() with ancStop==1 forms one data block. In case a complete data block is contained in one AncDataElement(), it has ancStart==1 and ancStop==1. If ancType==0x0 or ancType==0x1 then this data block constitutes one SacDataFrame() syntax element, padded at the end to obtain an integer number of bytes.

7.2.5 SacDataFrame including optional header

The syntax of “SacDataFrame()” is given in the table below:

Table 109 — Syntax of SacDataFrame()

Syntax	No. of bits	Mnemonic

SacDataFrame(sacHeaderFlag)
{
  numSacBits = 0;
  if (sacHeaderFlag) {
  sacTimeAlignFlag;
	1	**uimsbf**

numSacBits += 1; cnt = **sacHeaderLen**;	**7**	**uimsbf**

numSacBits += 7;
  if (cnt==127) {
  cnt += **sacHeaderLenAdd**;
	16	**uimsbf**

numSacBits += 16;
  }
  tmpBits = SpatialSpecificConfig();
		Note 1

numFillBits = (8*cnt)-tmpBits; **bsFillBits**;	**numFillBits**	**bslbf**

numSacBits += tmpBits+numFillBits;
  if (sacTimeAlignFlag) {
  sacTimeAlign;
	16	**uimsbf**

numSacBits += 16;
  }
  }
  numSacBits += SpatialFrame();
		Note 1

return (numSacBits); }		
Note 1: SpatialSpecificConfig() and SpatialFrame() return the number of bits read.		

The following semantics apply:

sacHeaderFlag	Indicates the presence of a SpatialSpecificConfig() element.
sacHeaderLen	Indicates the length in bytes of SpatialSpecificConfig() plus the following fill bits.
sacHeaderLenAdd	Additional length in bytes of SpatialSpecificConfig() plus the following fill bits, needed if this length is 127 or more bytes.
bsFillBits	Fill bits, to be ignored.
sacTimeAlignFlag	Indicates the presence of a sacTimeAlign element. If sacTimeAlignFlag==0 then sacTimeAlign is set to the default value 0.
sacTimeAlign	Identifies the PCM sample in the output frame of the downmix decoder that corresponds to the beginning of the present SAC frame. The position of the first sample of the output frame is represented as 0. The present SAC frame is the first SAC frame that is completed (i.e., ancStop==1) in the present downmix decoder frame.

7.3 Transport of MPEG Surround over PCM channels

7.3.1 Introduction

MPEG Surround data can be conveyed over traditional PCM audio channels through adding a noise signal with specific characteristics to the PCM data. This noise signal, which actually is a randomized version of the MPEG Surround data, can be rendered inaudible by employing subtractive dithered noise shaping controlled by the masked threshold.

7.3.2 Syntax

Table 110 — Syntax of BuriedData()

Syntax	No. of bits	Mnemonic

BuriedData()
{
  BuriedDataHeader()
  BuriedDataFrame()
}		

Table 111 — Syntax of BuriedDataHeader()

Syntax	No. of bits	Mnemonic
BuriedDataHeader() {		
bsBDSyncword	**16**	**BsMsbf**, Note 1
bsBDChannels	**4**	**UiMsbf**
bsBDFramelength	**7**	**UiMsbf**
bsBDSubframes	**4**	**UiMsbf**
bsBDReserved	**1**	**BsMsbf**
bd_hbits=40+bd_subframes*bd_channels*4 /* header bits */		Note 2,3
bd_hsize=MAX(4,ceil(bd_hbits/(bd_channels*8)))		
B=0		
for (subframe=0; subframe<bd_subframes; subframe++) {		
for (channel=0; channel < bd_channels; channel++) {		
bsBDAlloc[channel][subframe]	**4**	**UiMsbf**
B+=bsBDAlloc[channel][subframe]*bd_framelength/(bd_subframes*8)		Note 4
}		
}		
bsBDHeaderCrc	**8**	**UiMsbf**
for (i=0; i<bd_hsize*bd_channels*8-bd_hbits; i++) {		
bsBDHeaderPadding	**1**	**BsMsbf**
}		
}		
Note 1: The bsBDSyncword has the value 0xAA95 Note 2: bd_subframes depends on bsBDSubframes and is restricted by bsBDChannels and bsBDFramelength. Note 3: bd_channels depends on bsBDChannels Note 4: bd_framelength depends on bsBDFramelength.		

Table 112 — Syntax of BuriedDataFrame()

Syntax	No. of bits	Mnemonic

BuriedDataFrame()
{
if (B>0) {
for (channel=0; channel < bd_channels; channel++) {
if (bsBDAlloc[channel][0]>0) {
B-=bs_hsize
}
}
}
while (B>2) {
BuriedDataElement()
}
bsBDFramePadding
}
	B*8	**BsMsbf**

Table 113 — Syntax of BuriedDataElement()

Syntax	No. of bits	Mnemonic
BuriedDataElement() {		
bsBDType	**3**	**UiMsbf**
bsBDID	**3**	**UiMsbf**
bsBDLengthIdx	**2**	**UiMsbf**
B-=1		
if (bsBDLengthIdx>0) {		
bsBDLength	**bsBDLengthIdx*8**	**UiMsbf**
for (b=0; b<bsBDLength; b++) {		
bsBDBytes[b]	**8**	**BsMsbf**
}		
B-=bsBDLengthIdx+ bsBDLength		
}		
bsBDDataCrc	**16**	**UiMsbf**
B-=2		
}		

7.3.3 Semantics

bsBDSyncword The 16 bit string 1010101010010101, i.e., 0xAA95. This is only used in the buried data channel.

bsBDChannels Indicates the number of channels according to bd_channels = bsBDChannels + 1.

bsBDFramelength Indicates the length of a frame according to bd_framelength = (bsBDFramelength+1) * 64.

bsBDSubframes Indicates the number and size of bd_subframes within a frame according to bd_subframes = bsBDSubframes + 1 and bd_subframelength = bd_framelength / bd_subframes. Only those values of bsBDSubframes which result in a bd_subframelength that is a multiple of 64 are allowed In addition bsBDSubframes shall be chosen such that bd_subframelength is not smaller than bd_hsize * 8.

bsBDReserved Reserved bit for possible future usage. The default value for these bits equals 0.

bsBDAlloc[channel][subframe]
Indicates the number of bits used to code the samples in channel “channel” of sub-frame “subframe”. The three bits constitute an unsigned integer with values 0…7.

bsBDHeaderCrc A 8 bit cyclic redundancy check word on the data starting with the first bit after bsBDSyncword up to bsBDHeaderCrc.

bsBDHeaderPadding Padding bits to fill the unused part of BuriedDataHeader().

bsBDFramePadding Padding bits to fill the unused part of BuriedDataFrame.

bsBDType Indicates the type of buried data. The types 1..5 are reserved for future MPEG defined usage, type 6 can be used for user specific applications and type 7 can be used for padding the unused part of BuriedDataFrame(). In the latter case the padding data shall contain zero-valued data bytes.

Table 114 — bsBDType

bsBDType	Type of data
0	MPEG Surround frame, i.e., SacDataFrame(0)
1	MPEG Surround header+frame, i.e., SacDataFrame(1)
2 ... 5	reserved
6	user specific
7	padding

bsBDID Identification of the data to support multiple streams. BuriedDataElements()'s with the same bsBDID value belong to the same stream.

bsBDLengthIdx Indicates the length of the bsBDLength field.

bsBDLength The number of buried data payload bytes in BuriedDataElement(). This number shall not exceed the value of B-bsLengthIdx-2.

bsBDBytes[b] One byte of the payload of BuriedDataElement(). The concatenation of all bsBDBytes from one BuriedDataElement() forms one data block. If bsBDType==0x0 or bsBDType==0x1 then this data block constitutes one SacDataFrame() syntax element, padded at the end to obtain an integer number of bytes.

bsBDDataCrc A 16 bit cyclic redundancy check word on the bytes contained in BuriedDataElement(), excluding bsBDDataCrc.

7.3.4 Decoding process

7.3.4.1 Introduction

The decoding process consists of two steps. In the first step the payload from the buried data channel, is decoded. In the second step the payload is converted into a complete MPEG-Surround audio bit-stream.

In the next subclauses the decoding of the payload is described.

7.3.4.2 Extraction of buried data payload (including header information)

For the extraction of the buried data payload, the n bit PCM samples contained in the PCM data need to be available.

The decoding process describes the extraction of the buried data payload contained in uniquely decode-able buried data frames of 'bd_framelength' PCM samples. A buried data frame is subdivided into bd_subframes buried data subframes of $bd\_framelength / bd\_subframes$ samples each. Each subframe for each channel

has an individual allocation which is denoted by bsBDAlloc[channel][subframe]. For the corresponding channel “channel” and subframe “subframe”, this allocation indicates the number of LSB’s of the PCM sample that is used to carry the buried data frame. The header information is always contained in the LSB of the PCM samples. The applied frame structure is depicted in Figure 56. In this example the allocation of the buried data bd_subframes is as given in Table 115.

Table 115 — subframe allocation.

bsBDAlloc[channel][subframe]	channel	
subframe	0	1
0	0	2
1	1	2

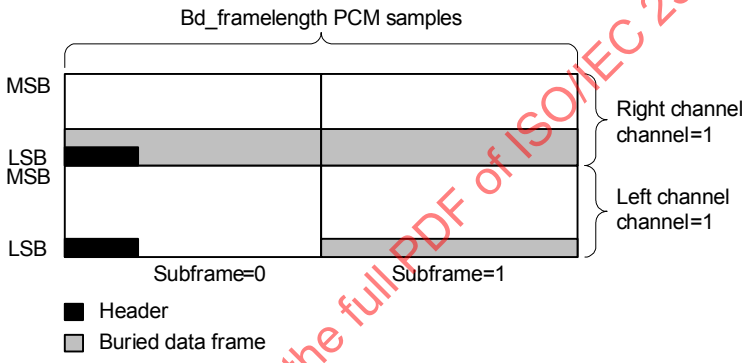

Figure 56 — Example buried data frame-structure with header.

In order to extract the correct number of LSB’s that are used to hold the buried data payload, the header needs to be read and interpreted first. After parsing the first 32 bits of the header in the LSB of the first channel the value of 'bd_hsize' can be calculated. Subsequently the remaining bits of BuriedDataHeader() are read from the LSB on a channel by channel basis.

Dependent on the allocation information in the header, the remaining LSB’s of the PCM samples that contain the header, may also hold buried data payload. The bits of BuriedDataFrame() are read on a sample by sample basis from the LSB's of the sample-interleaved channel data.

For perceptual control of the header information and the buried data payload, all the LSB’s contained in BuriedDataFrame(), except for bsBDSyncword, have to pass bit by bit through a de-randomization circuit prior to interpretation. The de-randomization circuit is illustrated in Figure 57. The blocks T represent shift registers. The additions represent “exclusive or gates”. At the start of every frame the shift registers are initialized to the value 1. For every new inserted input bit t_n, a new output bit s_n is generated.

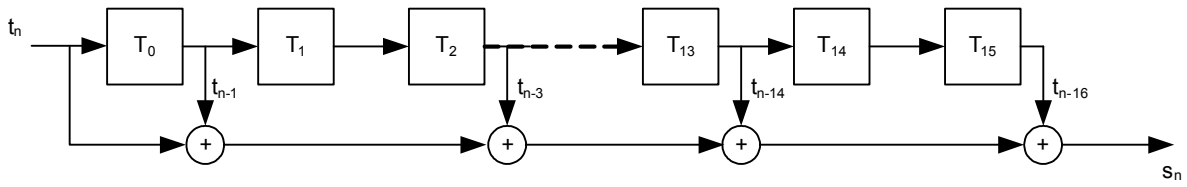

Figure 57 — The de-randomization circuit.

The following polynomial is applied:

$$s_n = t_n \oplus t_{n-1} \oplus t_{n-3} \oplus t_{n-14} \oplus t_{n-16}.$$

At the start of every frame all the states T_i are initialized to the value 1.

The bits have to be inserted into the de-randomization circuit in a specific order which is explained in Figure 58.

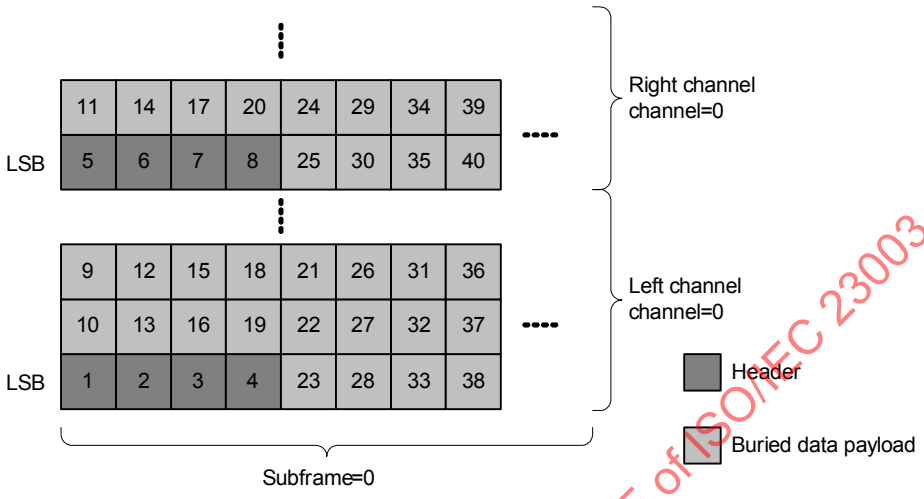

Figure 58 — The BuriedDataFrame()

The bits in BuriedDataFrame() need to be inserted into the de-randomization circuit in a specific order. In the figure this is explained by means of a simplified header and buried data payload. Assume that bsBDSyncword is only two bits and the remaining header is six bits. As illustrated in the figure, the allocation for the first subframe is 3 LSBs for the left channel and 2 bits for the right channel. The synchronization bits labelled “1” and “2” are read first and do not pass through the randomization circuit. The remaining bits are read in the indicated order. This order is “header first” where the bits are read channel by channel. After that, the bits are read MSB first in alternating channel order. All the bits labelled “3…” have to pass through the randomization circuit prior to interpretation.

7.3.4.3 Synchronization

The first action is synchronization of the decoder to the n bit PCM samples. The syncword is contained in the LSB of the PCM samples representing the first channel. The distance between two consecutive synchronisation words amounts to 'bd_framesize' PCM samples. In order to retrieve the syncword, a bit-stream is generated by successively concatenating the LSB of the PCM sample corresponding to the first channel. The last 16 bits of this bitstream are continuously compared to the syncword. If there is a match for all 16 bits, only then synchronization is achieved[1)].

1) Improved or more reliable synchronization can be achieved by also interpreting the bits up to the CRC-8 check. In the case the CRC-8 check on the appropriate data is false, either the data is in error or the synchronization was false. In both cases, the bit-stream needs to be re-synchronized.

7.3.4.4 Allocation

The information represented by alloc describes for each subframe the number of LSB's per channel that contain buried data payload. Header information is only present in the LSB of the first 8*bd_hsize PCM samples per channel residing in the first subframe. In the case bsBDAlloc[channel][0] is larger than 1, the LSB's up to bsBDAlloc[channel][0], except the LSB that hold the header information, contain buried data payload.

7.3.4.5 Retrieval of buried data payload

For all bd_subframes, a number of LSB's as described in the allocation, is extracted from the n bit PCM data in the MSB-first order and read in BuriedDataFrame(). For the first subframe, the bits contained in the header which is residing in the LSB of the PCM samples is omitted.

7.3.4.6 CRC check

Two CRC checks are performed. The error detection methods used are "CRC-8" and "CRC-16" which generator polynomials are

$$G(X) = X^8 + X^2 + X + 1 \text{ (CRC-8)}$$

$$G(X) = X^{16} + X^{15} + X^2 + 1 \quad \text{(CRC-16)}$$

The CRC method is depicted in the CRC-check diagram given in Figure 59. For CRC-8, the initial state of the shift register is 0xFF. For CRC-16, the initial state of the shift register is 0xFFFF. All bits included in the CRC check are input to the circuit shown in the figure. After each bit is input, the shift register is shifted by one bit. After the last shift operation, the outputs bn…b0 constitute a word to be compared with the CRC-check word in the stream. If the words are not identical, a transmission error has occurred in the field on which CRC-8 has been applied. To avoid annoying distortions, application of a concealment technique, such as muting of the actual frame or repetition of the previous frame is recommended.

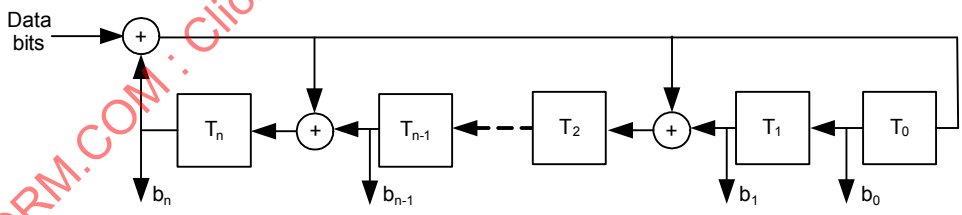

Figure 59 — CRC-check diagram. The addition blocks represent "exclusive or" gates.

7.3.4.7 MPEG Surround decoding

In the case that the buried data payload is of type MPEG Surround, the SpatialFrame data buried in one PCM frame shall be applied to the previously received PCM frame. Furthermore the first frame shall contain the SpatialSpecificConfig, i.e. bsBDType shall have the value 1, and it is recommended, for an encoder, to add the SpatialSpecificConfig to a buried_data_frame(), at regular time intervals, such that a decoder is also able to start decoding from another position in the stream.

Annex A
(normative)

Tables

A.1 Huffman Tables

The function *1Dhuff_dec()* is used as:

data = 1Dhuff_dec(hcod_huff, codeword),

where *hcod_huff* is the selected Huffman table and *codeword* is the word read from the bitstream. The returned value *data,* is a Huffman table index corresponding to a specific code word, with the Huffman table offset value subtracted. The offset value is specified for each Huffman table in Table A.1.

Similarly the function *2Dhuff_dec()* is used as:

(data0, data1) = 2Dhuff_dec(hcod_huff, codeword),

where *hcod_huff* is the selected Huffman table and *codeword* is the word read from the bitstream. The returned values *data0* and *data1* is the corresponding Huffman table index Idx0 and Idx1 corresponding to a specific code word with the Huffman table offset value subtracted. The offset value is specified for each Huffman table in Table A.1.

Table A.1 — Huffman table overview

Table name	Offset	LAV	Notes
hcodFirstBand_CLD	15	15	
hcodFirstBand_ICC	0	7	Note 1
hcodFirstBand_CPC	10	15	
hcod1D_CLD_YY	0	15	Note 1
hcod1D_ICC_YY	0	7	Note 1
hcod1D_CPC_YY	0	15	Note 1
hcond2D_CLD_YY_ZZ_LL_escape	N/A	N/A	
hcond2D_ICC_YY_ZZ_LL_escape	N/A	N/A	
hcond2D_CPC_YY_ZZ_LL_escape	N/A	N/A	
hcond2D_CLD_YY_ZZ_03	0	3	Note 1
hcond2D_CLD_YY_ZZ_05	0	5	Note 1
hcond2D_CLD_YY_ZZ_07	0	7	Note 1
hcond2D_CLD_YY_ZZ_09	0	9	Note 1
hcond2D_ICC_YY_ZZ_01	0	1	Note 1
hcond2D_ICC_YY_ZZ_03	0	3	Note 1
hcond2D_ICC_YY_ZZ_05	0	5	Note 1
hcond2D_ICC_YY_ZZ_07	0	7	Note 1
hcond2D_CPC_YY_ZZ_03	0	3	Note 1
hcond2D_CPC_YY_ZZ_06	0	6	Note 1
hcond2D_CPC_YY_ZZ_09	0	9	Note 1
hcond2D_CPC_YY_ZZ_12	0	12	Note 1
hcod1D_ICC_Diff	0	7	Note 1
hcodLavIdx	0	N/A	
hcod2D_EnvRes	0	N/A	
Note 1: Data can only have non-negative values for this table.			

Table A.2 — hcodFirstBand_CLD

Index	length	codeword (hexadecimal)	Index	length	codeword (hexadecimal)
0	8	0x52	16	4	0x06
1	9	0xae	17	4	0x08
2	9	0xaf	18	4	0x07
3	7	0x28	19	4	0x03
4	7	0x6e	20	4	0x01
5	6	0x36	21	5	0x1a
6	5	0x1e	22	5	0x13
7	4	0x0e	23	6	0x3e
8	4	0x0c	24	6	0x16
9	4	0x0a	25	6	0x17
10	4	0x02	26	7	0x6f
11	5	0x16	27	7	0x2a
12	5	0x12	28	8	0x56
13	5	0x17	29	8	0x53
14	4	0x00	30	6	0x3f
15	4	0x04			

Table A.3 — hcodFirstBand_ICC

Index	length	codeword (hexadecimal)	Index	length	codeword (hexadecimal)
0	5	0x1e	4	2	0x02
1	4	0x0e	5	2	0x01
2	3	0x06	6	6	0x3e
3	2	0x00	7	6	0x3f

Table A.4 — hcodFirstBand_CPC

Index	length	codeword (hexadecimal)	Index	length	codeword (hexadecimal)
0	8	0x0fe	13	3	0x004
1	8	0x076	14	3	0x006
2	7	0x03a	15	3	0x005
3	7	0x03e	16	5	0x01e
4	7	0x07e	17	6	0x01e
5	6	0x01c	18	7	0x03f
6	6	0x03e	19	8	0x077
7	4	0x002	20	10	0x3fe
8	4	0x003	21	10	0x3fc
9	4	0x006	22	11	0x7fe
10	4	0x00e	23	12	0xffe
11	3	0x000	24	12	0xfff
12	3	0x002	25	10	0x3fd

Table A.5 — hcod1D_CLD_YY

Index	DF		DT	
	length	codeword	length	codeword
0	1	0x000000	1	0x000000
1	2	0x000002	2	0x000002
2	3	0x000006	3	0x000006
3	4	0x00000e	4	0x00000e
4	5	0x00001e	5	0x00001e
5	6	0x00003e	6	0x00003e
6	7	0x00007e	7	0x00007e
7	8	0x0000fe	9	0x0001fe
8	9	0x0001fe	9	0x0001fc
9	10	0x0003fe	10	0x0003fe
10	11	0x0007fe	10	0x0003fa
11	12	0x000ffe	11	0x0007fe
12	13	0x001ffe	11	0x0007f6
13	15	0x007ffe	12	0x000ffe
14	15	0x007ffc	12	0x000fee
15	16	0x00fffe	13	0x001ffe
16	16	0x00fffa	13	0x001fde
17	17	0x01fffe	14	0x003ffe
18	17	0x01fff6	14	0x003fbe
19	18	0x03fffe	14	0x003fbf
20	18	0x03ffff	15	0x007ffe
21	19	0x07ffde	16	0x00fffe
22	18	0x03ffee	17	0x01fffe
23	20	0x0fffbe	19	0x07fffe
24	21	0x1fff7e	19	0x07fffc
25	24	0xfffbfc	20	0x0ffffa
26	24	0xfffbfd	21	0x1ffffc
27	24	0xfffbfe	21	0x1ffffd
28	24	0xfffbff	21	0x1ffffe
29	23	0x7ffdfc	21	0x1fffff
30	23	0x7ffdfd	20	0x0ffffb

Table A.6 — hcod1D_ICC_YY

Index	DF		DT	
	length	codeword	length	codeword
0	1	0x00	1	0x00
1	2	0x02	2	0x02
2	3	0x06	3	0x06
3	4	0x0e	4	0x0e
4	5	0x1e	5	0x1e
5	6	0x3e	6	0x3e
6	7	0x7e	7	0x7e
7	7	0x7f	7	0x7f

Table A.7 — hcod1D_CPC_YY

Index	DF		DT	
	length	codeword	length	codeword
0	2	0x00002	1	0x00000
1	2	0x00003	2	0x00002
2	2	0x00000	3	0x00006
3	3	0x00002	4	0x0000e
4	5	0x0000e	5	0x0001e
5	5	0x0000c	6	0x0003e
6	6	0x0001e	7	0x0007e
7	6	0x0001a	9	0x001fe
8	7	0x0003e	9	0x001fc
9	7	0x00036	10	0x003fe
10	8	0x0007e	10	0x003fa
11	8	0x0006e	11	0x007f6
12	9	0x000fe	12	0x00ffe
13	9	0x000de	11	0x007f7
14	10	0x001fe	12	0x00ffc
15	10	0x001be	13	0x01ffe
16	11	0x003fe	13	0x01ffa
17	11	0x0037e	14	0x03ffe
18	12	0x007fe	14	0x03ff6
19	12	0x006fe	15	0x07ffe
20	13	0x00ffe	16	0x0fffe
21	13	0x00dfe	16	0x0ffff
22	14	0x01ffe	16	0x0ffde
23	15	0x03ffe	17	0x1ffbe
24	15	0x03fff	17	0x1ffbf
25	13	0x00dff	15	0x07fee

Table A.8 — hcod2D_CLD_YY_ZZ_LL_escape

LL	DF/FP		DF/TP		DT/FP		DT/TP	
	length	codeword	length	codeword	length	codeword	length	codeword
03		N/A		N/A		N/A		N/A
05		N/A		N/A		N/A		N/A
07		N/A	21	0x17feff		N/A		N/A
09		N/A	21	0x17fef7	21	0x0b7dff		N/A

Table A.9 — hcod2D_ICC_YY_ZZ_LL_escape

LL	DF/FP		DF/TP		DT/FP		DT/TP	
	length	codeword	length	codeword	length	codeword	length	codeword
01		N/A		N/A		N/A		N/A
03		N/A		N/A		N/A		N/A
05	16	0x0ddff		N/A	18	0x36fff		N/A
07	12	0x0087b	16	0x0ffff	15	0x079ff	16	0x03dff

Table A.10 — hcod2D_CPC_YY_ZZ_LL_escape

LL	DF/FP		DF/TP		DT/FP		DT/TP	
	length	codeword	length	codeword	length	codeword	length	codeword
03		N/A		N/A		N/A		N/A
06		N/A		N/A		N/A		N/A
09		N/A		N/A		N/A		N/A
12	18	0x4fff		N/A		N/A		N/A

Table A.11 — hcod2D_CLD_YY_ZZ_03

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	2	0x002	1	0x000	1	0x000	1	0x000
0	1	3	0x002	6	0x03e	5	0x01e	4	0x00e
0	2	5	0x004	11	0x76e	10	0x3be	8	0x0fa
0	3	8	0x03e	12	0xede	12	0xefe	11	0x7de
1	0	4	0x006	3	0x006	3	0x006	4	0x00c
1	1	4	0x007	6	0x03f	5	0x01c	5	0x01e
1	2	6	0x00e	10	0x3b6	9	0x1de	8	0x0fe
1	3	10	0x0fe	6	0x03a	8	0x0ea	9	0x1f6
2	0	9	0x07e	5	0x01c	7	0x074	8	0x0ff
2	1	7	0x01e	8	0x0ee	8	0x0ee	7	0x07c
2	2	6	0x00c	9	0x1da	8	0x0eb	7	0x07e
2	3	5	0x005	5	0x01e	5	0x01f	5	0x01a
3	0	10	0x0ff	8	0x0ef	11	0x77e	11	0x7df
3	1	6	0x00d	12	0xedf	12	0xeff	10	0x3ee
3	2	3	0x000	8	0x0ec	7	0x076	5	0x01b
3	3	2	0x003	2	0x002	2	0x002	2	0x002

Table A.12 — hcod2D_CLD_YY_ZZ_05

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	3	0x0002	3	0x0006	2	0x0000	3	0x0006
0	1	3	0x0003	5	0x001c	4	0x0006	4	0x000e
0	2	5	0x0010	8	0x007e	7	0x0024	7	0x007c
0	3	7	0x007c	12	0x0efc	11	0x025e	10	0x03fe
0	4	8	0x00d6	16	0xeffe	14	0x3cfe	12	0x0fbe
0	5	10	0x03ee	17	0x1dffe	15	0x79fe	14	0x3efe
1	0	4	0x000a	3	0x0004	3	0x0006	3	0x0000
1	1	4	0x000c	4	0x000a	4	0x0007	3	0x0001
1	2	5	0x0016	7	0x003e	7	0x0078	6	0x003c
1	3	6	0x0034	12	0x0efe	10	0x03ce	8	0x005e
1	4	8	0x00fe	15	0x77fe	13	0x1e7e	11	0x07de
1	5	13	0x1f7e	7	0x0076	9	0x00be	11	0x07be
2	0	7	0x007e	4	0x0006	5	0x0008	6	0x001e
2	1	6	0x0036	5	0x0016	6	0x003e	5	0x000a
2	2	6	0x0026	8	0x00be	7	0x0026	6	0x001f
2	3	7	0x0046	12	0x0efd	10	0x012e	8	0x005f
2	4	9	0x011e	8	0x00ee	9	0x00bf	9	0x01ee
2	5	9	0x01f6	5	0x000e	7	0x002e	9	0x01f6
3	0	9	0x011f	6	0x003e	7	0x0027	9	0x01fe
3	1	8	0x00d7	6	0x002e	7	0x007a	8	0x00fe
3	2	8	0x008e	9	0x01de	9	0x01e4	8	0x00f6
3	3	8	0x00ff	10	0x03be	9	0x0096	8	0x00fa
3	4	7	0x006a	7	0x007e	7	0x007b	7	0x007e
3	5	7	0x004e	5	0x001e	6	0x003f	6	0x0016
4	0	12	0x0fbe	7	0x007f	9	0x01e6	11	0x07bf
4	1	11	0x07de	7	0x005e	9	0x01e5	10	0x03de
4	2	7	0x004f	14	0x3bfe	12	0x0f3e	10	0x03ee
4	3	6	0x0037	9	0x00fe	8	0x005e	7	0x007a
4	4	5	0x0017	6	0x001e	6	0x0016	5	0x000e
4	5	5	0x001e	3	0x0002	4	0x000e	4	0x0006
5	0	13	0x1f7f	8	0x00bf	11	0x079e	14	0x3eff
5	1	8	0x00fa	17	0x1dfff	15	0x79ff	13	0x1f7e
5	2	6	0x0022	13	0x1dfe	11	0x025f	10	0x03ff
5	3	5	0x0012	9	0x00ff	8	0x004a	7	0x002e
5	4	4	0x000e	6	0x003a	5	0x000a	4	0x0004
5	5	2	0x0000	2	0x0000	2	0x0002	2	0x0002

Table A.13 — hcod2D_CLD_YY_ZZ_07

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	4	0x00000e	3	0x000002	2	0x000000	3	0x000002
0	1	4	0x00000a	5	0x00001c	4	0x000006	4	0x00000a
0	2	5	0x00000a	8	0x0000bc	8	0x0000de	6	0x00001a
0	3	7	0x00007c	11	0x0005fc	11	0x00069e	9	0x0001be
0	4	8	0x0000be	15	0x005ffe	14	0x0034fe	11	0x0006e6
0	5	9	0x00017a	18	0x02ffde	17	0x01a7fe	12	0x00067a
0	6	9	0x0000ee	20	0x0bff7e	19	0x069ff6	13	0x000cf2
0	7	11	0x0007b6	21	0x17feff	19	0x069ff7	15	0x0033de
1	0	4	0x000006	3	0x000004	3	0x000002	4	0x00000c
1	1	4	0x00000c	4	0x00000a	4	0x00000c	4	0x00000e
1	2	5	0x000016	7	0x00000e	7	0x00006a	5	0x00000e
1	3	6	0x000026	10	0x0002fa	10	0x00034e	8	0x0000de
1	4	7	0x00003e	13	0x0001fe	13	0x001fde	10	0x000372
1	5	7	0x00002e	16	0x00bff2	15	0x0069fe	11	0x0003d6
1	6	9	0x0001ec	19	0x05ffbe	17	0x01a7fc	12	0x000678
1	7	15	0x0047ce	8	0x0000ee	10	0x000372	13	0x000cf6
2	0	6	0x000016	4	0x000002	6	0x00003e	6	0x000036
2	1	6	0x00003c	5	0x000016	6	0x00003c	5	0x000012
2	2	6	0x000022	8	0x0000f6	8	0x0000df	6	0x00003e
2	3	7	0x00004e	11	0x0005fe	10	0x0001ee	7	0x00003c
2	4	7	0x00003f	13	0x0001ff	12	0x000dde	9	0x0001b8
2	5	8	0x00005e	16	0x00bff6	15	0x0069fa	11	0x0003d4
2	6	12	0x0008fa	9	0x0001de	10	0x000373	11	0x00033e
2	7	12	0x0008fb	7	0x00007e	8	0x00007a	11	0x00033f
3	0	8	0x00005f	5	0x000000	7	0x00003e	8	0x00007e
3	1	8	0x0000fa	6	0x00003c	7	0x000068	7	0x00006a
3	2	8	0x0000bf	8	0x00000e	9	0x0001ba	7	0x00004e
3	3	7	0x00003a	10	0x00003e	10	0x0003f6	7	0x00007e
3	4	9	0x0001f6	14	0x002ffe	12	0x000d3e	9	0x0001ba
3	5	10	0x0001de	10	0x0002fb	10	0x00034c	9	0x0000ce
3	6	10	0x0003da	8	0x0000f7	9	0x0001fa	9	0x0000f6
3	7	11	0x0007b7	6	0x00002e	8	0x0000d2	10	0x0001ee
4	0	10	0x0001df	6	0x000006	8	0x00007e	10	0x0001ef
4	1	10	0x0003ee	7	0x00007a	8	0x00007f	9	0x00013e
4	2	9	0x00017b	8	0x00000a	9	0x0001f8	8	0x00007f
4	3	10	0x0003ef	11	0x00007e	11	0x0006ee	8	0x000066
4	4	9	0x0001ee	12	0x0000fe	11	0x0003de	8	0x0000d6
4	5	8	0x00008e	9	0x000016	9	0x0001b8	7	0x00003e
4	6	9	0x0001ef	7	0x000006	9	0x0001fc	8	0x0000d7
4	7	9	0x0001fe	5	0x000002	7	0x00006b	8	0x00009e
5	0	12	0x0008f8	7	0x00000f	9	0x0000f6	12	0x0007ae
5	1	11	0x00047e	7	0x000076	9	0x0001fe	10	0x0001e8
5	2	11	0x00047f	9	0x000017	10	0x00034d	10	0x0001e9
5	3	8	0x000076	15	0x005ff8	14	0x003fbe	10	0x00027e
5	4	7	0x00003c	12	0x000bfe	11	0x0007f6	7	0x000032
5	5	7	0x000046	9	0x00001e	10	0x0003fa	6	0x000018
5	6	7	0x00007a	7	0x00007f	7	0x00003c	6	0x000026
5	7	7	0x00007e	4	0x000003	6	0x00003d	6	0x000034
6	0	14	0x0023e6	7	0x000004	10	0x0003f7	13	0x000cf3
6	1	13	0x0011f2	8	0x0000bd	10	0x000376	12	0x0007aa
6	2	9	0x0001ff	16	0x00bff3	17	0x01a7ff	12	0x0007ab
6	3	7	0x00003d	15	0x005fff	14	0x003fbf	10	0x00027f

6	4	7	0x00004f	12	0x000bfa	12	0x000ddf	9	0x0001bf
6	5	6	0x00002e	9	0x00017c	9	0x0001f9	6	0x00001b
6	6	5	0x000012	6	0x00003a	6	0x000036	5	0x00001e
6	7	4	0x000004	3	0x000003	4	0x00000e	4	0x00000b
7	0	15	0x0047cf	9	0x00017e	11	0x0003df	15	0x0033df
7	1	9	0x00011e	21	0x17fefe	18	0x034ffa	14	0x0019ee
7	2	8	0x0000bc	17	0x017fee	15	0x0069fb	12	0x0007af
7	3	8	0x0000fe	15	0x005ffa	14	0x0034fc	11	0x0006e7
7	4	6	0x00001c	12	0x000bfb	12	0x000fee	9	0x0001bb
7	5	5	0x000010	9	0x0001df	9	0x0001ff	7	0x00007f
7	6	4	0x00000d	6	0x00003e	5	0x00000e	4	0x000008
7	7	2	0x000000	3	0x000006	2	0x000002	2	0x000000

Table A.14 — hcod2D_CLD_YY_ZZ_09

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	4	0x000006	4	0x00000e	3	0x000006	4	0x00000e
0	1	4	0x000007	5	0x000014	4	0x000004	4	0x000008
0	2	5	0x000006	8	0x00008e	7	0x000012	7	0x00007e
0	3	7	0x00007e	11	0x0004fe	11	0x0007fe	9	0x0001fe
0	4	7	0x00000a	14	0x0023fe	13	0x001f7e	10	0x0001ba
0	5	8	0x00001e	16	0x008ffe	16	0x00fbfe	12	0x000dbe
0	6	9	0x00008a	19	0x05ffbc	17	0x01f7fe	13	0x000d7e
0	7	10	0x00004e	21	0x17fef7	21	0x0b7dfe	14	0x001af6
0	8	10	0x000276	21	0x17fef7	21	0x0b7dff	15	0x007fec
0	9	11	0x0002e2	21	0x17fef7	21	0x0b7dff	17	0x01ffb6
1	0	4	0x000000	3	0x000002	3	0x000000	4	0x00000a
1	1	4	0x00000a	4	0x000002	4	0x000006	4	0x00000c
1	2	5	0x000016	7	0x000044	7	0x00007c	5	0x00000c
1	3	6	0x000026	10	0x00027e	9	0x000046	7	0x000036
1	4	7	0x000076	13	0x0017fc	12	0x0007d0	9	0x0000de
1	5	8	0x0000f2	16	0x00bff6	14	0x001f4e	11	0x0005fe
1	6	8	0x000012	19	0x05ffbe	17	0x00b7fe	12	0x0006be
1	7	8	0x00005e	17	0x011ff8	16	0x005bee	13	0x001b7e
1	8	9	0x00008b	20	0x0bff7a	18	0x016fbe	15	0x007fee
1	9	15	0x002e76	8	0x0000bc	10	0x0003ee	15	0x006dfe
2	0	6	0x000012	4	0x000006	5	0x000006	6	0x00001e
2	1	5	0x000007	5	0x000016	5	0x00000a	5	0x00000e
2	2	6	0x000038	7	0x00001a	7	0x00002e	5	0x00000a
2	3	7	0x00007c	10	0x0000fe	10	0x0003fe	7	0x00006a
2	4	7	0x000008	13	0x0011f6	12	0x0007d2	9	0x0001ae
2	5	8	0x000046	16	0x00bffe	14	0x001f4f	11	0x0006fe
2	6	8	0x0000f6	17	0x011ff9	15	0x002dfe	11	0x000376
2	7	9	0x0001ca	21	0x17fef6	17	0x00b7de	13	0x000dfe
2	8	14	0x00173a	9	0x00011e	10	0x0001fe	13	0x000dff
2	9	14	0x001738	7	0x000056	8	0x00002e	13	0x000d7f
3	0	8	0x00009e	5	0x000010	7	0x00007a	8	0x0000b6
3	1	7	0x00004a	6	0x00003e	7	0x00007e	7	0x00005e
3	2	7	0x000026	8	0x00009e	8	0x00007a	7	0x00007c
3	3	7	0x00000c	11	0x0007fe	10	0x0001fa	7	0x00006e
3	4	8	0x00004e	13	0x0011f7	12	0x0007fe	8	0x00006a
3	5	8	0x0000f7	15	0x005ff8	13	0x001f7c	9	0x00016a
3	6	9	0x00013a	17	0x017fee	14	0x0016fa	12	0x000ffe

3	7	11	0x00009e	11	0x0007ff	10	0x00009e	12	0x000dfe
3	8	12	0x0009fe	8	0x0000ae	8	0x000020	12	0x000ffc
3	9	12	0x00013e	7	0x00001e	8	0x000021	13	0x001bfe
4	0	9	0x000026	6	0x000026	8	0x0000fe	10	0x00035e
4	1	8	0x00001a	6	0x00000e	7	0x000016	9	0x0001b6
4	2	9	0x0001e6	9	0x0001ee	9	0x0000fe	8	0x00005e
4	3	9	0x0001e2	11	0x00047e	10	0x00016e	8	0x0000b4
4	4	8	0x0000ee	12	0x000bfc	10	0x00009f	7	0x00006c
4	5	9	0x0001ce	16	0x00bfff	13	0x000b7c	9	0x00017e
4	6	10	0x000277	12	0x0008fa	11	0x0003de	10	0x00036e
4	7	10	0x0003ce	9	0x00006e	9	0x0000b6	10	0x0003ee
4	8	11	0x0002e6	9	0x0001ef	9	0x0000be	11	0x00037e
4	9	11	0x0004fc	7	0x00007e	8	0x00007c	11	0x000377
5	0	11	0x0002e3	7	0x00007a	8	0x00005a	12	0x000fff
5	1	10	0x000170	7	0x00004e	8	0x000078	10	0x0001ae
5	2	10	0x000172	9	0x00007e	9	0x000047	10	0x0001be
5	3	9	0x0000ba	10	0x0000de	9	0x000044	9	0x0001f6
5	4	9	0x00003e	13	0x0011fe	12	0x0007ff	9	0x0001be
5	5	9	0x0001e3	14	0x002ffe	12	0x0007d1	8	0x0000da
5	6	8	0x00001b	11	0x0004ff	10	0x0001f6	8	0x0000fe
5	7	9	0x00003f	10	0x0000ff	10	0x0001f7	9	0x00016b
5	8	9	0x00009e	8	0x0000bd	8	0x00002f	9	0x0000d6
5	9	9	0x00009f	6	0x00002e	7	0x00002c	10	0x00037e
6	0	13	0x000b9e	8	0x0000fe	9	0x0000fc	13	0x0017fe
6	1	12	0x0009ff	8	0x0000af	9	0x0001f6	12	0x000bfe
6	2	11	0x0004fd	9	0x0001ec	9	0x0000f6	11	0x0007de
6	3	11	0x0004fe	11	0x0001be	11	0x0007ff	11	0x0006de
6	4	9	0x0001cf	17	0x011ffe	14	0x0016fe	10	0x0001b8
6	5	8	0x0000ef	14	0x002ffa	11	0x0002de	8	0x0000d6
6	6	8	0x000044	12	0x0008fe	11	0x0003ea	7	0x00002e
6	7	8	0x00005f	10	0x0003fe	9	0x0000bf	7	0x000034
6	8	8	0x0000e4	7	0x000046	8	0x0000fa	8	0x0000de
6	9	8	0x0000f0	5	0x000012	6	0x00000a	8	0x0000be
7	0	15	0x002e72	8	0x00003e	9	0x00004e	15	0x007fef
7	1	12	0x00013f	7	0x000045	8	0x000026	12	0x0006bc
7	2	13	0x000b9f	10	0x0002fe	10	0x0001ee	13	0x001bff
7	3	9	0x00013e	20	0x0bff7e	16	0x005bfe	13	0x001ffa
7	4	8	0x0000fe	15	0x005ff9	14	0x003efe	10	0x0001b9
7	5	8	0x000047	15	0x005ffa	13	0x000b7e	10	0x0003fe
7	6	7	0x00000e	12	0x000bfd	11	0x0003eb	8	0x0000fa
7	7	7	0x00007d	9	0x00013e	9	0x0001fe	6	0x00002e
7	8	6	0x000010	6	0x00000c	7	0x00007b	6	0x000034
7	9	6	0x000024	4	0x000007	5	0x000007	6	0x00001f
8	0	15	0x002e77	8	0x0000be	10	0x0001fb	15	0x006dff
8	1	16	0x005ce6	8	0x000036	9	0x000045	14	0x001af7
8	2	9	0x0000bb	20	0x0bff7f	18	0x016ffe	14	0x0036fe
8	3	8	0x0000e6	18	0x023ffe	17	0x01f7ff	12	0x0006fe
8	4	8	0x000016	17	0x011ffa	15	0x002df6	12	0x000fbe
8	5	8	0x0000ff	15	0x005ffe	13	0x001f7d	10	0x00035f
8	6	7	0x00007a	11	0x0001bf	11	0x0003fe	8	0x0000b7
8	7	6	0x00003a	9	0x0001ed	8	0x00005e	6	0x00002c
8	8	5	0x000017	6	0x00002a	6	0x00003c	5	0x00001e
8	9	4	0x000002	3	0x000000	4	0x00000e	4	0x000009
9	0	16	0x005ce7	9	0x00017e	11	0x0003df	17	0x01ffb7
9	1	10	0x0003cf	21	0x17fef7	20	0x05befe	16	0x00ffda

9	2	8	0x000017	19	0x047ffe	19	0x02df7e	13	0x000d7a
9	3	9	0x0001cb	19	0x047fff	18	0x016fff	13	0x0017ff
9	4	8	0x00009c	17	0x011ffb	15	0x007dfe	12	0x000fbf
9	5	7	0x00004b	14	0x002ffb	13	0x000fa6	10	0x0002fe
9	6	6	0x000016	11	0x00047c	11	0x0007de	8	0x00005f
9	7	5	0x00000a	9	0x0001fe	8	0x000079	6	0x000016
9	8	4	0x000008	6	0x00003c	5	0x00000e	4	0x000004
9	9	3	0x000006	3	0x000006	2	0x000002	2	0x000000

Table A.15 — hcod2D_ICC_YY_ZZ_01

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	1	0x0	1	0x0	1	0x0	1	0x0
0	1	3	0x6	3	0x6	3	0x6	3	0x6
1	0	3	0x7	3	0x7	3	0x7	3	0x7
1	1	2	0x2	2	0x2	2	0x2	2	0x2

Table A.16 — hcod2D_ICC_YY_ZZ_03

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	2	0x002	2	0x002	2	0x002	2	0x002
0	1	2	0x000	4	0x004	4	0x00e	4	0x00e
0	2	5	0x00a	10	0x17e	10	0x37e	8	0x0fc
0	3	8	0x07e	11	0x2fe	12	0xdfe	12	0xfde
1	0	5	0x00e	2	0x000	4	0x00f	4	0x00c
1	1	4	0x004	5	0x00e	4	0x00c	4	0x00d
1	2	6	0x016	9	0x0be	9	0x1ba	9	0x1fe
1	3	11	0x3fe	6	0x016	9	0x1bb	11	0x7ee
2	0	10	0x1fe	5	0x00f	8	0x0de	9	0x1fa
2	1	9	0x0fe	6	0x014	8	0x0dc	9	0x1ff
2	2	7	0x03e	8	0x05e	9	0x1be	8	0x0fe
2	3	6	0x01e	4	0x006	5	0x01a	6	0x03e
3	0	11	0x3ff	7	0x02e	11	0x6fe	12	0xfdf
3	1	6	0x017	11	0x2ff	12	0xdff	10	0x3f6
3	2	4	0x006	6	0x015	6	0x036	5	0x01e
3	3	2	0x003	2	0x003	1	0x000	1	0x000

Table A.17 — hcod2D_ICC_YY_ZZ_05

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	2	0x00000	2	0x00000	1	0x00000	2	0x00000
0	1	3	0x00002	5	0x0001e	4	0x0000c	4	0x0000e
0	2	5	0x0000c	10	0x003fc	9	0x001b6	7	0x0003a
0	3	7	0x0006a	16	0x0fffa	13	0x01b7c	11	0x00676
0	4	8	0x000dc	20	0xfff9e	16	0x0dbfe	13	0x019fe
0	5	11	0x006ee	20	0xfff9f	18	0x36fff	16	0x0cebe
1	0	5	0x0001e	3	0x00006	4	0x0000e	4	0x0000f
1	1	4	0x0000c	4	0x00004	5	0x0001e	3	0x00002
1	2	5	0x0000d	9	0x000be	9	0x001be	6	0x0001e
1	3	6	0x0001e	15	0x07ffe	12	0x00dfe	9	0x000fe
1	4	9	0x001ae	19	0x7ffce	18	0x36ffe	13	0x019d6
1	5	16	0x0ddff	8	0x000fe	10	0x0036e	15	0x0675e
2	0	8	0x000de	4	0x00006	7	0x0006e	7	0x0003e
2	1	7	0x0007e	6	0x0001e	8	0x000fe	6	0x00032
2	2	6	0x0001f	10	0x003fd	8	0x000d8	5	0x00018
2	3	9	0x001be	16	0x0fffb	14	0x036fe	10	0x0033e
2	4	15	0x06efe	12	0x00ffe	11	0x006de	12	0x00cfe
2	5	16	0x0ddfe	6	0x0003e	8	0x000de	11	0x00677
3	0	14	0x0377e	5	0x0000a	9	0x001fa	11	0x00674
3	1	13	0x01bbe	7	0x0007e	8	0x000da	9	0x0019c
3	2	12	0x00dde	13	0x01ffe	12	0x00dff	9	0x000ff
3	3	9	0x001bf	15	0x07fff	13	0x01b7e	7	0x0003b
3	4	8	0x000d6	8	0x0005e	8	0x000d9	6	0x0001c
3	5	10	0x00376	5	0x0000e	8	0x000ff	8	0x0007e
4	0	16	0x0ddff	6	0x0001f	10	0x003f6	14	0x033fe
4	1	16	0x0ddff	10	0x003fe	11	0x006fe	14	0x033ff
4	2	9	0x001ba	17	0x1fff2	15	0x06dfe	12	0x00cea
4	3	6	0x00034	12	0x00ffc	10	0x0037e	7	0x00066
4	4	6	0x0003e	7	0x0002e	8	0x000fc	5	0x0001a
4	5	5	0x0000e	4	0x0000e	5	0x0001a	4	0x00006
5	0	16	0x0ddff	9	0x000bf	11	0x007ee	16	0x0cebf
5	1	9	0x001af	18	0x3ffe6	17	0x1b7fe	14	0x033ae
5	2	7	0x0007f	16	0x0fff8	13	0x01b7d	11	0x0067e
5	3	6	0x00036	12	0x00ffd	11	0x007ef	9	0x0019e
5	4	4	0x0000e	6	0x00016	6	0x0003e	5	0x0001b
5	5	2	0x00002	2	0x00002	2	0x00002	2	0x00002

Table A.18 — hcod2D_ICC_YY_ZZ_07

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	2	0x0000	2	0x0002	1	0x0000	2	0x0002
0	1	4	0x000c	6	0x001e	4	0x000c	4	0x0002
0	2	6	0x002e	12	0x0ffe	11	0x07ee	9	0x00fe
0	3	7	0x0044	16	0xffff	13	0x1e7e	12	0x07be
0	4	8	0x0086	16	0xfffe	14	0x3cfe	13	0x0ffc
0	5	11	0x069e	16	0xffff	15	0x79ff	13	0x0ffd
0	6	11	0x043e	16	0xffff	15	0x79ff	15	0x1efe
0	7	12	0x087a	16	0xffff	15	0x79ff	16	0x3dfe
1	0	5	0x001e	3	0x0006	4	0x000e	4	0x0004
1	1	4	0x000e	5	0x0008	5	0x001a	3	0x0000
1	2	6	0x002a	11	0x07fe	9	0x01e6	7	0x003c
1	3	7	0x0046	16	0xffff	13	0x1fbe	10	0x00f6
1	4	9	0x015e	16	0xffff	15	0x79fe	11	0x01da
1	5	7	0x0047	16	0xffff	15	0x79ff	12	0x03fe
1	6	10	0x034a	16	0xffff	15	0x79ff	15	0x3dfe
1	7	12	0x087b	8	0x005a	11	0x06fc	16	0x3dff
2	0	8	0x00d6	4	0x0006	7	0x006c	8	0x003c
2	1	6	0x0026	7	0x007a	8	0x00f6	7	0x003e
2	2	6	0x002f	10	0x0164	9	0x01ba	5	0x000a
2	3	8	0x00d7	15	0x7ffa	12	0x0dfc	8	0x003a
2	4	7	0x006a	16	0xffff	12	0x0dfd	11	0x03de
2	5	10	0x034e	16	0xffff	15	0x79ff	13	0x07be
2	6	12	0x087b	13	0x1fee	12	0x0f3e	14	0x0f7e
2	7	12	0x087b	6	0x003c	9	0x01bb	14	0x1efe
3	0	10	0x02be	5	0x000e	8	0x00dc	11	0x01de
3	1	9	0x01a6	8	0x00fe	9	0x01fe	10	0x00ec
3	2	9	0x01be	11	0x02ce	10	0x036e	9	0x007e
3	3	5	0x0012	11	0x02cf	10	0x03fe	5	0x000c
3	4	9	0x01bf	15	0x7ffb	15	0x79ff	10	0x01ee
3	5	12	0x087b	13	0x1fec	12	0x0fde	13	0x0f7e
3	6	12	0x087b	9	0x00b0	9	0x01ee	12	0x07fc
3	7	12	0x087b	7	0x002e	8	0x00f2	15	0x3dff
4	0	12	0x087b	6	0x003e	9	0x01fa	16	0x7ffe
4	1	12	0x087b	10	0x03fe	10	0x03f6	12	0x03be
4	2	12	0x087b	10	0x0165	9	0x01be	10	0x00fe
4	3	12	0x087b	15	0x7ffc	15	0x79ff	10	0x01fe
4	4	6	0x0036	13	0x1fef	13	0x1fbf	6	0x001a
4	5	8	0x00d0	11	0x07fa	10	0x03ce	7	0x001c
4	6	11	0x043c	11	0x07f8	10	0x03ff	12	0x07fd
4	7	11	0x043f	6	0x001f	8	0x00de	13	0x0ffe
5	0	12	0x087b	7	0x002f	7	0x0078	16	0x3dff
5	1	12	0x087b	8	0x00f6	8	0x00da	12	0x03bf
5	2	12	0x087b	13	0x1fed	15	0x79ff	14	0x1ffe
5	3	10	0x034b	16	0xffff	15	0x79ff	12	0x03ff
5	4	6	0x0027	15	0x7ffd	11	0x06fd	8	0x003e
5	5	6	0x0020	12	0x0ff2	10	0x036c	6	0x001b
5	6	7	0x0042	9	0x00b1	9	0x01ef	8	0x007e
5	7	8	0x00d1	5	0x000a	8	0x00fe	9	0x00f6
6	0	12	0x087b	5	0x0009	10	0x036f	16	0x7fff
6	1	12	0x087b	10	0x0166	12	0x0dfe	16	0x3dff
6	2	10	0x02bf	16	0xffff	15	0x79ff	15	0x3ffe
6	3	8	0x00de	16	0xffff	15	0x79ff	11	0x01db

6	4	8	0x00ae	15	0x7ffe	15	0x79ff	10	0x00ee
6	5	7	0x0056	14	0x3ffc	10	0x036d	8	0x007a
6	6	5	0x0016	8	0x005b	8	0x00fc	5	0x000e
6	7	5	0x0014	4	0x000e	6	0x003e	5	0x000b
7	0	12	0x087b	7	0x007e	12	0x0dff	16	0x3dff
7	1	11	0x069f	16	0xffff	15	0x79ff	16	0x3dff
7	2	9	0x01a4	16	0xffff	15	0x79ff	12	0x03de
7	3	9	0x010e	16	0xffff	15	0x79ff	11	0x01fe
7	4	7	0x0045	16	0xffff	15	0x79ff	11	0x01ee
7	5	7	0x006e	12	0x0ff3	11	0x079e	9	0x007a
7	6	5	0x001f	8	0x00f7	7	0x007a	5	0x0006
7	7	2	0x0001	2	0x0000	2	0x0002	2	0x0003

Table A.19 — hcod2D_CPC_YY_ZZ_03

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	2	0x002	2	0x002	1	0x000	1	0x000
0	1	3	0x002	5	0x00e	4	0x00c	4	0x00e
0	2	5	0x006	10	0x1fe	8	0x0d2	8	0x0fc
0	3	8	0x076	11	0x3fe	10	0x34e	11	0x7fe
1	0	4	0x002	2	0x000	4	0x00e	4	0x00c
1	1	4	0x006	4	0x004	5	0x01e	5	0x01e
1	2	6	0x01e	9	0x0fe	7	0x068	8	0x0fe
1	3	10	0x1de	6	0x01e	7	0x06a	9	0x1fe
2	0	9	0x0ee	5	0x00a	7	0x06e	8	0x0fd
2	1	7	0x03a	6	0x016	7	0x06f	7	0x07c
2	2	6	0x01c	7	0x03e	7	0x06b	7	0x07d
2	3	5	0x007	4	0x006	5	0x01f	5	0x01a
3	0	10	0x1df	8	0x07e	9	0x1a6	11	0x7ff
3	1	6	0x01f	11	0x3ff	10	0x34f	10	0x3fe
3	2	3	0x000	6	0x017	6	0x036	5	0x01b
3	3	2	0x003	2	0x003	2	0x002	2	0x002

Table A.20 — hcod2D_CPC_YY_ZZ_06

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	3	0x0006	3	0x0006	2	0x0000	2	0x0000
0	1	3	0x0002	5	0x001e	4	0x000c	4	0x000e
0	2	5	0x000e	8	0x00be	7	0x003a	7	0x007a
0	3	7	0x005e	11	0x077e	10	0x03ae	9	0x01ee
0	4	8	0x00ba	13	0x17fe	12	0x0efe	11	0x07fe
0	5	9	0x00fe	16	0xbffe	15	0x77fe	13	0x1ffe
0	6	10	0x02ee	16	0xbfff	16	0xeffe	14	0x3ffe
1	0	4	0x0006	3	0x0004	3	0x0002	4	0x000c
1	1	4	0x000a	4	0x000a	4	0x0006	4	0x000d
1	2	5	0x0016	7	0x005e	7	0x007e	6	0x001e
1	3	6	0x0024	9	0x00d2	9	0x01de	8	0x00f6
1	4	8	0x00fa	12	0x0bfe	11	0x075e	10	0x03fe
1	5	9	0x017e	14	0x2ffe	14	0x3bfe	11	0x03ba
1	6	15	0x5dfe	8	0x006a	11	0x075f	12	0x0f7e
2	0	7	0x005c	5	0x001f	6	0x003c	6	0x001c
2	1	6	0x0026	5	0x000e	6	0x003e	6	0x003e
2	2	6	0x001e	7	0x0036	7	0x0074	6	0x0016
2	3	8	0x00fe	10	0x03be	9	0x01d4	8	0x0074
2	4	9	0x01f2	12	0x07fe	11	0x03be	10	0x01de
2	5	12	0x0bbe	9	0x01de	10	0x01fe	10	0x017a
2	6	12	0x0fbe	7	0x003e	9	0x00fe	10	0x01df
3	0	9	0x00ff	6	0x002c	8	0x00ee	9	0x01fe
3	1	8	0x007e	6	0x002d	7	0x003e	7	0x002e
3	2	9	0x01fe	8	0x007e	8	0x007e	8	0x0076
3	3	9	0x017f	10	0x01fe	9	0x00e6	9	0x00fe
3	4	10	0x03ee	9	0x00d3	9	0x00e7	9	0x00be
3	5	10	0x03fe	8	0x00ee	8	0x0072	9	0x01fc
3	6	10	0x03ff	6	0x002e	7	0x0038	8	0x0075
4	0	12	0x0fbf	7	0x003c	9	0x00ee	10	0x01dc
4	1	11	0x07de	7	0x003d	9	0x01d6	9	0x00bc
4	2	11	0x05de	9	0x00fe	10	0x03be	10	0x03de
4	3	9	0x01f3	11	0x03fe	10	0x01ff	9	0x00bf
4	4	8	0x00f8	9	0x017e	8	0x0076	8	0x007e
4	5	7	0x004a	7	0x0076	7	0x0076	7	0x0078
4	6	7	0x004b	5	0x001c	6	0x001e	7	0x007e
5	0	14	0x2efe	8	0x006b	11	0x077e	12	0x0f7f
5	1	13	0x177e	8	0x0068	10	0x01de	11	0x03bb
5	2	9	0x01f6	13	0x0ffe	12	0x077e	11	0x07be
5	3	7	0x003e	11	0x05fe	10	0x03aa	9	0x00ff
5	4	6	0x0027	8	0x006e	8	0x00fe	7	0x003e
5	5	5	0x0010	6	0x003a	6	0x003d	5	0x000a
5	6	5	0x001e	3	0x0002	4	0x000d	4	0x0006
6	0	15	0x5dff	10	0x02fe	12	0x077f	14	0x3fff
6	1	9	0x0176	15	0x5ffe	16	0xefff	12	0x0ffe
6	2	8	0x00be	13	0x0fff	13	0x1dfe	10	0x017b
6	3	7	0x007e	11	0x077f	10	0x03ab	9	0x01fd
6	4	5	0x0011	8	0x006f	8	0x00ff	7	0x0079
6	5	4	0x000e	5	0x000c	5	0x001c	4	0x0004
6	6	2	0x0000	2	0x0000	2	0x0002	2	0x0002

Table A.21 — hcod2D_CPC_YY_ZZ_09

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	3	0x00000	3	0x00004	3	0x00006	3	0x00004
0	1	4	0x0000a	5	0x00014	4	0x00004	4	0x0000c
0	2	5	0x0000a	8	0x000f6	7	0x0003a	6	0x0001e
0	3	6	0x00010	10	0x003d2	9	0x000be	8	0x0007e
0	4	7	0x00026	12	0x00ffe	11	0x003b6	10	0x003ce
0	5	8	0x0003e	13	0x00f5e	13	0x00ece	11	0x0057e
0	6	9	0x0009e	15	0x066fe	14	0x009fe	13	0x01ffe
0	7	9	0x00064	16	0x0cdfe	15	0x013fe	13	0x01eb6
0	8	10	0x000ce	18	0x3dffe	18	0x1d9fe	14	0x03dfe
0	9	11	0x0019e	19	0x7bffe	18	0x1d9ff	15	0x05ede
1	0	4	0x00002	3	0x00002	3	0x00000	4	0x0000d
1	1	4	0x00008	4	0x00006	4	0x0000e	4	0x0000e
1	2	5	0x00012	7	0x0006e	6	0x0001c	6	0x0003e
1	3	6	0x00026	9	0x0017e	8	0x0003e	7	0x0002e
1	4	7	0x0004e	11	0x007be	10	0x001d8	9	0x001e2
1	5	8	0x000e8	12	0x007ae	12	0x00772	10	0x0015e
1	6	8	0x0007e	14	0x03dfe	13	0x00f7e	11	0x007ae
1	7	9	0x000fe	15	0x03d6e	14	0x00efe	12	0x00f5a
1	8	10	0x0027e	17	0x1effe	16	0x0767e	13	0x017b6
1	9	16	0x06ffe	10	0x003de	12	0x0027e	13	0x00fbe
2	0	6	0x00011	5	0x00018	6	0x0003e	6	0x0002c
2	1	6	0x0003e	5	0x00015	5	0x0000a	5	0x0000e
2	2	6	0x0001e	7	0x0005a	6	0x00008	6	0x0002a
2	3	7	0x0007e	9	0x00178	8	0x0003a	8	0x000fe
2	4	7	0x0001a	11	0x0066e	10	0x0016e	9	0x0017a
2	5	8	0x000ee	12	0x00b7e	11	0x003b2	10	0x003de
2	6	9	0x001de	14	0x0337e	13	0x00ffe	11	0x0079e
2	7	10	0x003fe	15	0x03d6f	14	0x00eff	12	0x007fe
2	8	14	0x013ba	10	0x003d6	12	0x0077e	13	0x01efe
2	9	14	0x01bfe	9	0x001ea	11	0x001de	13	0x01fff
3	0	8	0x0005c	6	0x0002e	7	0x0001c	8	0x000be
3	1	7	0x0002c	6	0x0002c	7	0x0003c	7	0x0005c
3	2	7	0x0002d	7	0x0003a	8	0x0007e	7	0x0002f
3	3	7	0x00018	9	0x000f6	9	0x000fa	8	0x000f2
3	4	8	0x000e6	10	0x001e6	9	0x0007e	8	0x00054
3	5	9	0x001df	12	0x007ac	11	0x003b8	10	0x003c6
3	6	9	0x0007a	14	0x02dfe	13	0x0077e	11	0x003fe
3	7	13	0x00dfe	10	0x001ea	11	0x002de	12	0x00ff6
3	8	12	0x0033e	9	0x0017a	10	0x0009e	11	0x003ee
3	9	12	0x0033f	8	0x000cc	10	0x001ee	11	0x0057f
4	0	10	0x003ae	7	0x00064	9	0x000f8	10	0x003fe
4	1	9	0x0013e	7	0x0006f	8	0x0005a	9	0x001ee
4	2	9	0x000ff	8	0x0007e	9	0x000f6	9	0x001ec
4	3	8	0x0003c	9	0x000f7	8	0x0002c	8	0x0007c
4	4	9	0x001d6	11	0x005e6	9	0x0004c	9	0x001e6
4	5	10	0x003ff	13	0x019be	11	0x002df	10	0x001f6
4	6	11	0x002fe	11	0x005e7	11	0x003fe	10	0x0015f
4	7	11	0x0037e	9	0x000f2	10	0x001da	10	0x001fe
4	8	11	0x004fe	8	0x000b6	9	0x0005c	10	0x003c7
4	9	11	0x002ff	7	0x0003b	9	0x0004e	11	0x007be
5	0	11	0x00276	8	0x000ce	10	0x001de	11	0x005ee
5	1	10	0x0013a	8	0x000fe	9	0x0005e	10	0x002f2

5	2	10	0x001be	8	0x00078	9	0x000bc	9	0x000fa
5	3	10	0x001bc	10	0x002fe	10	0x001fe	10	0x003da
5	4	10	0x0017e	11	0x003d2	10	0x000ee	10	0x002f3
5	5	9	0x0009c	12	0x00cde	10	0x000be	9	0x000aa
5	6	9	0x0007e	10	0x001e8	9	0x0005d	9	0x0015e
5	7	9	0x0009f	9	0x001e8	9	0x000f9	9	0x001ea
5	8	9	0x00065	8	0x000ca	9	0x000bf	9	0x000ae
5	9	9	0x0007f	7	0x0007e	8	0x0002d	9	0x000fe
6	0	13	0x009de	9	0x001fe	11	0x003be	12	0x007de
6	1	12	0x009fe	8	0x000cb	9	0x0004d	11	0x007fe
6	2	12	0x009ff	9	0x0016e	10	0x000bf	11	0x007af
6	3	12	0x006fe	11	0x007fe	11	0x003ba	11	0x005ef
6	4	9	0x0007b	13	0x00f5a	12	0x0077f	11	0x007ae
6	5	9	0x001ce	12	0x00fff	10	0x0016c	9	0x000ab
6	6	8	0x000ea	10	0x003d7	9	0x000fe	8	0x000bf
6	7	8	0x0005e	9	0x001ee	9	0x000fb	8	0x000ac
6	8	8	0x0006e	7	0x0003e	8	0x0007a	8	0x000f0
6	9	8	0x0007c	6	0x0003c	7	0x0002c	8	0x000f4
7	0	14	0x013bb	9	0x0017b	11	0x0013e	13	0x01eb7
7	1	13	0x009dc	9	0x0019a	11	0x003bb	12	0x00bda
7	2	13	0x009df	10	0x002de	12	0x007fe	12	0x007ff
7	3	9	0x00066	15	0x07bfe	14	0x01d9e	12	0x00f7e
7	4	9	0x001fe	13	0x00f5f	12	0x00766	11	0x007fa
7	5	8	0x000e9	11	0x005be	10	0x0016d	9	0x00178
7	6	7	0x0001b	10	0x002f2	9	0x00076	8	0x00056
7	7	7	0x00072	8	0x0007f	7	0x00012	7	0x0005d
7	8	6	0x0000e	6	0x0001c	6	0x0000a	6	0x00014
7	9	6	0x00012	5	0x0001a	6	0x0003f	6	0x0002d
8	0	17	0x0dffe	10	0x002ff	12	0x003be	14	0x02f6e
8	1	15	0x037fe	10	0x003fe	12	0x007be	13	0x00fbf
8	2	10	0x003af	16	0x0cdff	15	0x013ff	13	0x01ffc
8	3	9	0x001cf	14	0x01eb6	13	0x004fe	12	0x00ff7
8	4	8	0x0009e	13	0x01efe	12	0x00773	10	0x002be
8	5	8	0x000fe	12	0x00f7e	11	0x003b7	10	0x003db
8	6	7	0x00076	10	0x00336	9	0x0007f	8	0x000ad
8	7	6	0x00038	8	0x000be	7	0x0001e	7	0x0007e
8	8	5	0x0000e	6	0x00036	5	0x00006	5	0x00014
8	9	5	0x0001e	4	0x0000e	4	0x00006	4	0x00006
9	0	17	0x0dfff	10	0x001e7	14	0x01ffe	15	0x05edf
9	1	10	0x001bd	19	0x7bfff	17	0x0ecfe	14	0x03dff
9	2	9	0x000be	16	0x0f7fe	15	0x03b3e	13	0x01ffd
9	3	8	0x0005d	14	0x02dff	14	0x01fff	11	0x005ec
9	4	8	0x0007d	13	0x016fe	13	0x00f7f	11	0x0079f
9	5	7	0x00036	11	0x003d3	11	0x003de	10	0x003fc
9	6	6	0x0001a	10	0x003d3	9	0x000bd	8	0x000ae
9	7	5	0x0000c	8	0x000cf	7	0x0002e	6	0x00016
9	8	4	0x0000b	6	0x0003e	5	0x0001e	4	0x00004
9	9	3	0x00006	2	0x00000	2	0x00002	2	0x00000

Table A.22 — hcod2D_CPC_YY_ZZ_12

Idx0	Idx1	DF/FP		DF/TP		DT/FP		DT/TP	
		length	codeword	length	codeword	length	codeword	length	codeword
0	0	4	0x0000e	3	0x00002	3	0x00006	3	0x00002
0	1	4	0x00004	5	0x00014	4	0x00008	4	0x00008
0	2	5	0x00004	8	0x000fe	7	0x00078	7	0x0007e
0	3	6	0x00002	10	0x003fe	9	0x0017a	8	0x0009a
0	4	7	0x0000a	11	0x0057a	11	0x005fe	10	0x003be
0	5	8	0x00032	12	0x0077c	12	0x006be	11	0x006fe
0	6	9	0x000be	14	0x03eee	13	0x00d7e	12	0x009e6
0	7	10	0x00362	15	0x05dfe	15	0x03d5e	13	0x017be
0	8	10	0x00042	15	0x057be	17	0x0a3fe	14	0x02ffa
0	9	11	0x003de	17	0x177fe	17	0x17f7e	15	0x04f7e
0	10	10	0x0007e	19	0x7defe	17	0x17f4e	14	0x0387e
0	11	11	0x0062e	20	0xfbdfe	17	0x0f57e	15	0x04f7f
0	12	12	0x00d8e	20	0xfbdff	18	0x1eafe	16	0x07ffe
1	0	5	0x0001a	3	0x00000	4	0x0000e	4	0x0000a
1	1	4	0x00006	4	0x00006	4	0x0000a	4	0x0000c
1	2	5	0x00016	7	0x0007c	6	0x0002e	6	0x0003a
1	3	6	0x0002e	9	0x001f6	8	0x000bc	7	0x0004e
1	4	7	0x00066	10	0x002b2	10	0x003d6	9	0x001d8
1	5	8	0x000fe	12	0x00fbe	11	0x0037a	10	0x0037e
1	6	8	0x00028	13	0x015ee	13	0x017f6	11	0x005be
1	7	9	0x000f6	14	0x02ebe	14	0x01ffe	12	0x009ee
1	8	10	0x0033e	16	0x0fbde	16	0x0bfbc	13	0x017bf
1	9	9	0x0007e	16	0x0dbfe	15	0x028fe	13	0x01fde
1	10	10	0x0036e	17	0x15ef6	16	0x0bfae	13	0x013de
1	11	10	0x000ce	18	0x3ef7c	19	0x3d5fe	14	0x027be
1	12	17	0x027fe	10	0x003b0	13	0x00ffe	15	0x077fe
2	0	6	0x00003	5	0x0001a	6	0x0003e	6	0x0002c
2	1	5	0x00000	5	0x00016	5	0x0000e	5	0x0000e
2	2	6	0x00030	7	0x0006c	6	0x00016	6	0x00036
2	3	7	0x0007e	9	0x001f4	8	0x000b2	7	0x0004a
2	4	7	0x00016	10	0x001de	10	0x003de	9	0x001d9
2	5	8	0x0005e	12	0x00fba	11	0x0037e	10	0x002fe
2	6	9	0x0018e	13	0x01b7e	13	0x00fce	11	0x005bf
2	7	9	0x00024	14	0x02eba	14	0x02fea	12	0x00b7a
2	8	9	0x0019e	15	0x07dee	13	0x00d7f	12	0x00fae
2	9	10	0x0037e	16	0x077de	15	0x05fd2	13	0x01f4e
2	10	10	0x000de	17	0x15ef7	17	0x17f4f	14	0x03bfe
2	11	16	0x07ffe	10	0x001ce	12	0x0056e	14	0x03bfc
2	12	16	0x013fe	9	0x00174	12	0x00bea	14	0x0387f
3	0	8	0x0007c	6	0x00038	7	0x0003c	8	0x000e0
3	1	7	0x0003c	6	0x00039	7	0x00079	7	0x0006e
3	2	7	0x0002e	7	0x00038	7	0x0002a	7	0x0005a
3	3	7	0x0001a	9	0x001b6	8	0x0006c	7	0x0003e
3	4	8	0x0007e	10	0x000ee	9	0x000a0	9	0x001de
3	5	9	0x001f6	12	0x00dbe	11	0x0035a	10	0x002f6
3	6	9	0x00062	13	0x0175c	12	0x0051e	11	0x004ce
3	7	9	0x001bc	13	0x007ba	12	0x0056f	11	0x0073e
3	8	9	0x00022	15	0x06dfc	14	0x01eae	12	0x00ede
3	9	10	0x000f2	16	0x0af7a	16	0x0bfa6	13	0x01f4f
3	10	15	0x02ffe	11	0x00766	12	0x007be	13	0x0133e
3	11	15	0x02fff	9	0x00076	12	0x007fa	13	0x017fe
3	12	14	0x004fc	8	0x0003e	11	0x005be	13	0x01db8

4	0	9	0x0006e	6	0x0000a	9	0x001ee	9	0x0012e
4	1	8	0x00030	6	0x0000b	8	0x000b6	8	0x0007e
4	2	8	0x00029	8	0x000da	8	0x00052	9	0x001fe
4	3	8	0x00016	9	0x00158	9	0x000fe	9	0x001fc
4	4	8	0x0001e	10	0x000ef	10	0x002f6	9	0x001f6
4	5	9	0x0007a	12	0x00bbc	11	0x003d6	10	0x0038e
4	6	9	0x001b6	12	0x00bbd	11	0x005bf	10	0x0039e
4	7	9	0x00026	13	0x007bb	12	0x006fa	11	0x005fe
4	8	10	0x000fa	15	0x05d76	14	0x01fff	12	0x00e1e
4	9	13	0x007be	11	0x005d6	12	0x007bf	12	0x009e4
4	10	14	0x01ffe	10	0x002f6	11	0x0037c	12	0x00b7b
4	11	14	0x0363e	9	0x001da	11	0x005ff	12	0x00e76
4	12	14	0x00f7e	9	0x001be	11	0x00286	13	0x0133f
5	0	10	0x0004e	7	0x0003a	10	0x003d2	11	0x007d2
5	1	9	0x00025	7	0x0003e	9	0x00166	10	0x003ea
5	2	10	0x003fe	8	0x0003f	10	0x003d7	10	0x003b4
5	3	9	0x00023	10	0x003b2	10	0x001ee	10	0x003b5
5	4	9	0x00020	11	0x005fe	10	0x00144	10	0x003b6
5	5	8	0x0002a	11	0x00566	9	0x000ae	8	0x00098
5	6	9	0x00056	12	0x003dc	11	0x007a8	10	0x0038f
5	7	10	0x000f6	14	0x03ef6	12	0x007ee	11	0x004f6
5	8	12	0x005fe	12	0x00dba	11	0x0035e	11	0x004ca
5	9	12	0x007fe	10	0x000f6	11	0x003f6	11	0x005ee
5	10	12	0x007fc	9	0x0015a	10	0x0015e	11	0x006ff
5	11	13	0x018be	9	0x000fe	11	0x002be	12	0x0099e
5	12	12	0x0013e	8	0x0007e	11	0x005f4	12	0x007fe
6	0	12	0x00c5e	8	0x000bc	10	0x0014c	12	0x00efe
6	1	11	0x006c6	8	0x000de	10	0x002f7	11	0x0073f
6	2	11	0x007de	9	0x0017e	10	0x0015a	11	0x0070e
6	3	11	0x006fe	10	0x002bc	11	0x003fc	11	0x00738
6	4	10	0x001fe	10	0x001cf	10	0x003df	9	0x000fe
6	5	10	0x00316	12	0x00fbc	10	0x001e8	10	0x003e8
6	6	10	0x0036f	13	0x01f76	11	0x003fe	10	0x003fe
6	7	11	0x006ff	12	0x00af6	11	0x005f6	10	0x00264
6	8	11	0x007df	11	0x00567	10	0x001ae	10	0x0027a
6	9	10	0x00043	10	0x002fe	10	0x002ce	10	0x00386
6	10	11	0x003df	10	0x003b6	11	0x0037b	11	0x003fe
6	11	11	0x0009e	9	0x001f5	10	0x00146	12	0x00fea
6	12	11	0x001ee	8	0x000ee	10	0x001f8	11	0x003fc
7	0	13	0x00bfe	9	0x001fa	11	0x003d7	13	0x01db9
7	1	12	0x007fd	8	0x0003c	11	0x007a9	12	0x00e72
7	2	12	0x003de	10	0x003fc	11	0x003de	12	0x00bde
7	3	11	0x002fe	9	0x00074	10	0x001bc	11	0x007f4
7	4	11	0x0015e	11	0x00767	11	0x005f7	11	0x0077e
7	5	11	0x0015f	12	0x00bbe	11	0x003f2	11	0x0073a
7	6	10	0x001f6	13	0x00efa	11	0x002bf	11	0x007f6
7	7	9	0x0007f	12	0x00dbb	10	0x001e9	9	0x001c6
7	8	9	0x0007c	11	0x007dc	9	0x000ac	9	0x0017e
7	9	10	0x000fb	10	0x000ea	11	0x005fc	10	0x0025a
7	10	10	0x001ee	9	0x000ee	10	0x001ac	10	0x00278
7	11	10	0x001f7	8	0x000ae	10	0x003d3	10	0x0026e
7	12	10	0x0017e	7	0x0005c	9	0x000da	10	0x00266
8	0	14	0x017fe	9	0x0015b	12	0x007aa	14	0x02ffb
8	1	14	0x0363f	9	0x00176	11	0x0037f	13	0x01dba
8	2	13	0x01b1e	9	0x0017a	10	0x00142	12	0x00feb

8	3	13	0x00ffe	10	0x002be	11	0x003d4	12	0x00faf
8	4	13	0x018bf	11	0x00396	11	0x0028e	12	0x00e73
8	5	10	0x000df	14	0x01df6	13	0x00d6e	12	0x00fa6
8	6	9	0x0003e	13	0x0177e	12	0x00b7a	10	0x0025b
8	7	9	0x001bd	12	0x00fbf	10	0x001fa	9	0x0013e
8	8	9	0x0018f	11	0x00397	10	0x00145	9	0x001c2
8	9	9	0x00188	10	0x002bf	10	0x002cf	9	0x0016e
8	10	9	0x000fe	9	0x001bf	9	0x000db	9	0x0013f
8	11	9	0x000fa	7	0x0001c	9	0x001e8	9	0x0017a
8	12	9	0x00189	6	0x0001e	8	0x0006e	9	0x001be
9	0	16	0x07ffc	10	0x003ff	13	0x01eae	15	0x07f76
9	1	14	0x004fd	9	0x001fb	11	0x005bc	12	0x007fa
9	2	14	0x004fe	10	0x003b1	11	0x002b6	13	0x01fdf
9	3	14	0x00f7f	11	0x006de	12	0x007fb	13	0x01dbb
9	4	10	0x000f3	15	0x03bee	14	0x01ade	13	0x01f7e
9	5	9	0x0002e	14	0x02ebf	13	0x00f56	11	0x004cb
9	6	9	0x001b2	13	0x0175e	11	0x00287	10	0x0026f
9	7	9	0x00078	12	0x003de	12	0x00beb	11	0x007de
9	8	9	0x001be	11	0x005d4	10	0x0014d	9	0x00136
9	9	8	0x000be	10	0x003b7	9	0x000f6	8	0x000be
9	10	8	0x000ce	8	0x00076	8	0x0006a	8	0x000e2
9	11	8	0x000da	7	0x0006e	8	0x000f6	8	0x000e6
9	12	8	0x000fa	5	0x00004	7	0x0003e	8	0x000ee
10	0	17	0x0fffa	9	0x000ff	12	0x007ef	15	0x077ff
10	1	16	0x07fff	9	0x000e6	12	0x00f56	14	0x03fba
10	2	15	0x009fe	10	0x000eb	12	0x006b6	13	0x00ffe
10	3	10	0x000ae	16	0x0bbfe	16	0x0bfbe	13	0x017fc
10	4	10	0x003ee	15	0x057bc	14	0x02fe8	12	0x009e7
10	5	9	0x001b0	14	0x02efe	13	0x00fcf	11	0x005bc
10	6	9	0x0002f	14	0x03ef4	13	0x00df6	12	0x00e77
10	7	9	0x001b3	12	0x0077e	12	0x007fe	11	0x007d6
10	8	8	0x0007a	11	0x005d5	10	0x0014e	9	0x0012c
10	9	7	0x00017	9	0x0007a	9	0x0016e	8	0x000bc
10	10	7	0x0006e	8	0x000be	7	0x00034	7	0x00072
10	11	6	0x00006	6	0x0002a	6	0x00017	6	0x00024
10	12	6	0x0000e	5	0x0001e	6	0x0003f	6	0x0002e
11	0	18	0x04ffe	10	0x002f7	13	0x00df7	15	0x03ffe
11	1	17	0x0fffb	10	0x002ee	13	0x01eaf	14	0x01ffe
11	2	10	0x000cf	18	0x3ef7e	17	0x0a3ff	14	0x03bfd
11	3	10	0x0033f	16	0x0dbff	16	0x0bfaf	13	0x017ff
11	4	9	0x0018a	15	0x057bf	14	0x0147e	12	0x009e5
11	5	10	0x003ff	15	0x06dfd	15	0x05fd6	12	0x007fb
11	6	9	0x0017e	14	0x03ef5	13	0x00a3e	12	0x00edf
11	7	8	0x00036	12	0x003df	12	0x007e6	11	0x007dc
11	8	7	0x0000e	11	0x005ff	10	0x0014f	10	0x003ff
11	9	7	0x0007c	10	0x003fd	9	0x0017c	8	0x000b6
11	10	6	0x00032	8	0x000ef	7	0x00058	7	0x0007c
11	11	5	0x0000e	6	0x0003a	5	0x0000c	5	0x0001a
11	12	5	0x0001e	4	0x0000c	4	0x00009	4	0x00006
12	0	18	0x04fff	10	0x001ca	14	0x01adf	16	0x07fff
12	1	10	0x0007f	18	0x3ef7d	19	0x3d5ff	15	0x07f77
12	2	10	0x002fe	17	0x177ff	16	0x051fe	14	0x03efe
12	3	9	0x00066	16	0x077df	16	0x07abe	13	0x01c3e
12	4	10	0x002ff	15	0x05d77	17	0x17f7f	14	0x03eff
12	5	9	0x00063	15	0x06dfe	16	0x0bfbd	13	0x01fdc

12	6	9	0x001fe	14	0x03eef	14	0x02fee	12	0x00fbe
12	7	8	0x000c6	12	0x0077f	12	0x00b7b	11	0x007dd
12	8	7	0x0005e	11	0x006dc	11	0x007aa	9	0x0012f
12	9	6	0x00016	9	0x000e4	9	0x0017e	8	0x000de
12	10	5	0x0000a	8	0x000fc	7	0x0005a	6	0x0001e
12	11	4	0x0000a	5	0x00006	4	0x00004	5	0x0001e
12	12	3	0x00004	3	0x00004	2	0x00000	2	0x00000

Table A.23 — hcod1D_ICC_Diff

Index	length	codeword (hexadecimal)	Index	length	codeword (hexadecimal)
0	1	0x01	4	5	0x01
1	2	0x01	5	6	0x01
2	3	0x01	6	7	0x01
3	4	0x01	7	7	0x00

Table A.24 — hcodLavIdx

Index	length	codeword (hexadecimal)
0	1	0x0
1	2	0x2
2	3	0x6
3	3	0x7

Table A.25 — hcod2D_EnvRes

erVal	erLen	length	codeword	erVal	erLen	length	codeword
-2	1	5	0x006	0	5	5	0x007
-2	2	6	0x01e	0	6	6	0x03a
-2	3	8	0x0f8	0	7	6	0x01f
-2	4	9	0x1f2	0	8	5	0x01e
-2	5	10	0x3e6	1	1	3	0x002
-2	6	11	0x7ce	1	2	4	0x006
-2	7	11	0x17e	1	3	5	0x004
-2	8	10	0x3ee	1	4	7	0x07e
-1	1	3	0x004	1	5	8	0x0fa
-1	2	4	0x00a	1	6	9	0x1f6
-1	3	5	0x00e	1	7	10	0x3ff
-1	4	6	0x00a	1	8	9	0x05e
-1	5	7	0x016	2	1	6	0x03b
-1	6	9	0x1fe	2	2	8	0x0fe
-1	7	10	0x3fe	2	3	9	0x05d
-1	8	9	0x05c	2	4	10	0x0be
0	1	3	0x006	2	5	11	0x17f
0	2	3	0x000	2	6	12	0xf9e
0	3	4	0x00b	2	7	12	0xf9f
0	4	5	0x01c	2	8	10	0x3ef