INTERNATIONAL STANDARD

**ISO/IEC 23008-3**

First edition
2015-10-15

Corrected version
2016-03-01

# Information technology — High efficiency coding and media delivery in heterogeneous environments —

## Part 3: 3D audio

*Technologies de l'information — Codage à haute efficacité et livraison des medias dans des environnements hétérogènes —*

*Partie 3: Audio 3D*

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

Reference number
ISO/IEC 23008-3:2015(E)

© ISO/IEC 2015

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2015, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org



© ISO/IEC 2015 – All rights reserved

# Contents

Page

Foreword ..... viii
Introduction ..... x
1 Scope ..... 1
2 Normative references ..... 1
3 Terms, definitions and mnemonics ..... 1
3.1 Terms and Definitions ..... 1
3.2 Mnemonics ..... 1
4 Technical Overview ..... 2
4.1 Decoder block diagram ..... 2
4.2 Overview over the codec building blocks ..... 3
4.3 Efficient combination of decoder processing blocks in time domain and QMF domain ..... 4
4.4 Rule set for determining processing domains ..... 5
4.4.1 Audio Core Codec, Processing Domain ..... 5
4.4.2 Mixing ..... 6
4.4.3 Audio Core Codec, Interface Domain to Rendering ..... 6
4.4.4 Rendering Context ..... 6
4.4.5 Post-Processing Context ..... 6
4.4.6 End-of-Chain Context ..... 7
5 MPEG-H 3D Audio Core decoder ..... 7
5.1 Terms and Definitions ..... 7
5.1.1 Joint Stereo ..... 7
5.1.2 MPEG Surround based stereo (MPS 212) ..... 7
5.2 Syntax ..... 7
5.2.1 General ..... 7
5.2.2 Decoder configuration ..... 7
5.2.3 MPEG-H 3D Audio Core bitstream payloads ..... 22
5.3 Data Structure ..... 30
5.3.1 General ..... 30
5.3.2 General Configuration Data Elements ..... 30
5.3.3 Loudspeaker Configuration Data Elements ..... 32
5.3.4 Core Decoder Configuration Data Elements ..... 34
5.3.5 Downmix Matrix Data Elements ..... 37
5.3.6 HOA Rendering Matrix Data Elements ..... 40
5.4 Configuration Element Descriptions ..... 42
5.4.1 General ..... 42
5.4.2 Downmix configuration ..... 43
5.4.3 HOA rendering matrix configuration ..... 47
5.5 Tool Descriptions ..... 51
5.5.1 General ..... 51
5.5.2 Quad Channel Element ..... 52
5.5.3 Transform Splitting ..... 53
5.5.4 MPEG Surround for Mono to Stereo upmixing ..... 60
5.5.5 Enhanced Noise Filling ..... 62
5.5.6 Audio Pre-Roll ..... 82
5.6 Buffer requirements ..... 86
5.6.1 Minimum decoder input buffer ..... 86
5.6.2 Bit reservoir ..... 86
5.6.3 Maximum bit rate ..... 87
5.7 Stream Access Point requirements and inter-frame dependency ..... 87
6 Dynamic Range Control and Loudness Processing ..... 88

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

6.1 Introduction ..... 88
6.2 Description ..... 88
6.3 Syntax ..... 88
6.3.1 Loudness Metadata ..... 88
6.3.2 Dynamic Range Control Metadata ..... 89
6.3.3 Data Elements ..... 90
6.4 Decoding Process ..... 91
6.4.1 General ..... 91
6.4.2 Dynamic Range Control ..... 93
6.4.3 Usage of downmixId in MPEG-H ..... 93
6.4.4 DRC Set Selection Process ..... 94
6.4.5 DRC-1 for SAOC 3D Content ..... 95
6.4.6 DRC-1 for HOA Content ..... 96
6.4.7 Loudness Normalization ..... 98
6.4.8 Peak Limiter ..... 98
6.4.9 Time-Synchronization of DRC gains ..... 98

7 Object Metadata Decoding ..... 98
7.1 Introduction ..... 98
7.2 Description ..... 98
7.3 Syntax ..... 99
7.3.1 Object Metadata Configuration ..... 99
7.3.2 Top level object metadata syntax ..... 100
7.3.3 Subsidiary payloads for efficient object metadata decoding ..... 100
7.3.4 Subsidiary payloads for object metadata decoding with low delay ..... 104
7.4 Data Structure ..... 108
7.4.1 Definition of ObjectMetadataConfig() payloads ..... 108
7.4.2 Efficient Object Metadata Decoding ..... 108
7.4.3 Object Metadata Decoding with Low Delay ..... 113

8 Object Rendering ..... 117
8.1 Description ..... 117
8.2 Terms and Definitions ..... 117
8.3 Input data ..... 117
8.4 Processing ..... 119
8.4.1 Imaginary Loudspeakers ..... 119
8.4.2 Dividing the Loudspeaker Setup into a Triangle Mesh ..... 120
8.4.3 Rendering Algorithm ..... 121

9 SAOC 3D ..... 126
9.1 Description ..... 126
9.2 Definitions ..... 126
9.3 Delay and synchronization ..... 127
9.4 Syntax ..... 127
9.4.1 Payloads for SAOC 3D ..... 127
9.4.2 Definition of SAOC 3D payloads ..... 131
9.5 SAOC 3D processing ..... 133
9.5.1 Compressed data stream decoding and dequantization of SAOC 3D data ..... 133
9.5.2 Time/frequency tranforms ..... 133
9.5.3 Signals and parameters ..... 133
9.5.4 SAOC 3D decoding ..... 135
9.5.5 Dual mode ..... 140

10 Generic Loudspeaker Rendering/Format Conversion ..... 141
10.1 Description ..... 141
10.2 Definitions ..... 142
10.2.1 General remarks ..... 142
10.2.2 Variable definitions ..... 142
10.3 Processing ..... 143
10.3.1 Application of transmitted downmix matrices ..... 143
10.3.2 Application of transmitted equalizer settings ..... 148
10.3.3 Downmix processing involving multiple channel groups ..... 148

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

 © ISO/IEC 2015 – All rights reserved

10.3.4 Initialization of the format converter ..... 149
10.3.5 Audio signal processing ..... 165

11 Immersive Loudspeaker Rendering / Format Conversion ..... 171
11.1 Description ..... 171
11.2 Syntax ..... 172
11.3 Definitions ..... 173
11.3.1 General remarks ..... 173
11.3.2 Variable definitions ..... 173

12 Higher Order Ambisonics (HOA) ..... 221
12.1 Technical Overview ..... 221
12.1.1 Block Diagram ..... 221
12.1.2 Overview of the decoder tools ..... 222
12.2 Syntax ..... 223
12.2.1 Configuration of HOA elements ..... 223
12.2.2 Payloads of HOA elements ..... 224
12.3 Data Structure ..... 229
12.3.1 Definitions of HOA Config ..... 229
12.3.2 Definitions of HOA payload ..... 231
12.4 HOA Tool Description ..... 234
12.4.1 HOA Frame Converter ..... 234
12.4.2 Spatial HOA decoding ..... 243
12.4.3 HOA Renderer ..... 255

13 Binaural Renderer ..... 263
13.1 Introduction ..... 263
13.2 Frequency-Domain Binaural Renderer ..... 264
13.2.1 Introduction ..... 264
13.2.2 Definitions ..... 266
13.2.3 Parameterization of Binaural Room Impulse Responses ..... 270
13.2.4 Frequency-Domain Binaural Processing ..... 282
13.3 Time-Domain Binaural Renderer ..... 289
13.3.1 Introduction ..... 289
13.3.2 Definitions ..... 290
13.3.3 Parameterization of Binaural Room Impulse Responses ..... 291
13.3.4 Time-Domain Binaural Processing ..... 296

14 MPEG-H 3D audio stream (MHAS) ..... 297
14.1 Overview ..... 297
14.2 Syntax ..... 297
14.2.1 Main MHAS syntax elements ..... 297
14.2.2 Subsidiary MHAS syntax elements ..... 299
14.3 Semantics ..... 299
14.3.1 mpeghAudioStreamPacket() ..... 299
14.3.2 MHASPacketPayload() ..... 300
14.4 Description of MHASPacketTypes ..... 300
14.4.1 PACTYP_FILLDATA ..... 300
14.4.2 PACTYP_MPEGH3DACFG ..... 300
14.4.3 PACTYP_MPEGH3DAFRAME ..... 301
14.4.4 PACTYP_SYNC ..... 301
14.4.5 PACTYP_SYNCGAP ..... 301
14.4.6 PACTYP_MARKER ..... 301
14.4.7 PACTYP_CRC16 and PACTYP_CRC32 ..... 302
14.4.8 PACTYP_DESCRIPTOR ..... 302
14.4.9 PACTYP_USERINTERACTION ..... 302
14.4.10 PACTYP_LOUDNESS_DRC ..... 302
14.4.11 PACTYP_BUFFERINFO ..... 303
14.5 Application Examples ..... 303
14.5.1 Light-weighted broadcast ..... 303
14.5.2 MPEG-2 Transport Stream ..... 303
14.6 Multi-Stream Delivery and Interface ..... 304

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

15 Metadata Audio Elements (MAE) ..... 306
15.1 Introduction ..... 306
15.2 Syntax ..... 307
15.3 Semantics ..... 311
15.4 Definition of mae_metaDataElementIDs ..... 319

16 Loudspeaker Distance Compensation ..... 319

17 Interfaces to the MPEG-H 3D audio decoder ..... 320
17.1 General ..... 320
17.2 Interface for local setup information ..... 321
17.2.1 General ..... 321
17.2.2 WIRE output ..... 321
17.2.3 Syntax for local setup information ..... 321
17.2.4 Semantics for local setup information ..... 321
17.3 Interface for local loudspeaker setup and rendering ..... 322
17.3.1 General ..... 322
17.3.2 Syntax for local loudspeaker signaling ..... 322
17.3.3 Semantics for local loudspeaker signaling ..... 323
17.4 Interface for binaural room impulse responses (BRIRs) ..... 324
17.4.1 Introduction ..... 324
17.4.2 Syntax of Binaural Renderer Interface ..... 324
17.4.3 Semantics ..... 327
17.5 Interface for local screen size information ..... 332
17.5.1 General ..... 332
17.5.2 Syntax ..... 332
17.5.3 Semantics ..... 332
17.6 Interface for signaling of local zoom area ..... 333
17.6.1 General ..... 333
17.6.2 Syntax ..... 333
17.6.3 Semantics ..... 334
17.7 Interface for user interaction ..... 334
17.7.1 Introduction ..... 334
17.7.2 Definition of User Interaction Categories ..... 334
17.7.3 Definition of an Interface for User Interaction ..... 335
17.7.4 Syntax of interaction interface ..... 335
17.7.5 Semantics of interaction interface ..... 336
17.8 Interface for loudness normalization and dynamic range control (DRC) ..... 338

18 Application and processing of local setup information and interaction data ..... 338
18.1 *Element Metadata* Preprocessing ..... 338
18.2 Interactivity Limitations and Restrictions ..... 341
18.2.1 General Information ..... 341
18.2.2 WIRE Interactivity ..... 341
18.2.3 Position Interactivity ..... 342
18.2.4 Screen-Related Element Remapping and Object Remapping for Zooming ..... 342
18.2.5 Closest Speaker Playback ..... 342
18.3 Screen-Related Element Remapping ..... 342
18.4 Object Remapping for Zooming ..... 344
18.5 Determination of the Closest Speaker ..... 345

Annex A (normative) Tables for arithmetic decoding of IGF information ..... 347
A.1 cf_se01[27] ..... 347
A.2 cf_se10[27] ..... 347
A.3 cf_se02[ 7][27] ..... 347
A.4 short cf_se20[ 7][27] ..... 347
A.5 short cf_se11[ 7][ 7][27] ..... 348
A.6 cf_off_se01 ..... 349
A.7 cf_off_se10 ..... 349
A.8 cf_off_se02[ 7] ..... 349
A.9 short cf_off_se20[ 7] ..... 350
A.10 cf_off_se11[ 7][ 7] ..... 350


IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

 © ISO/IEC 2015 – All rights reserved

**Annex B** (normative) **SAOC 3D Decorrelator pre-mixing matrices** ..... 351
B.1 Premixing matrix for output configurations with small number of output channels ..... 351
B.2 Premixing matrix for 22.2 output configuration ..... 351
B.3 Algorithm for generating pre-mixing matrices ..... 352
B.3.1 Input to the algorithm and representations ..... 352
B.3.2 Algorithm steps ..... 352

**Annex C** (informative) **Encoder Tools** ..... 356
C.1 General Overview ..... 356
C.1.1 Encoder block diagram ..... 356
C.1.2 Overview of the encoder and decoder building blocks ..... 356
C.2 Core Encoder Tools ..... 357
C.2.1 Quad Channel Element ..... 357
C.2.2 Transform Splitting ..... 358
C.2.3 Calculation of Residual Signal for MPEG Surround with Hybrid Residual Coding ..... 359
C.2.4 Enhanced Noise Filling ..... 359
C.3 Object Metadata Encoding ..... 360
C.3.1 Pre-Processing of the Object Metadata ..... 360
C.3.2 Efficient Object Metadata Encoding ..... 361
C.3.3 Object Metadata Encoding with Low Delay ..... 361
C.3.4 Spatially skipping objects ..... 361
C.4 SAOC 3D Encoder ..... 361
C.4.1 Overview ..... 361
C.4.2 Calculation of the SAOC 3D parameters ..... 361
C.4.3 Time/frequency transform ..... 362
C.4.4 Framing ..... 362
C.4.5 Parameter quantization and coding ..... 362

**Annex D** (informative) **Peak limiter for unguided clipping prevention** ..... 384

**Annex E** (normative) **Compact Template Downmix Matrices** ..... 385

**Annex F** (normative) **HOA Tables** ..... 386
F.6 32 Uniformly Distributed Positions in Spherical Coordinates ..... 390
F.12 Table of 256x8 weighting values, WeightValCdbk ..... 404

**Annex G** (informative) **Low Complexity HOA Rendering** ..... 421
G.1 Tool Description ..... 421
G.2 Predominant Sound Rendering ..... 421
G.3 Ambient Sound Rendering ..... 422
G.4 Output Signal Composition ..... 422

**Annex H** (informative) **Information on delay and complexity of Time-Domain binauralization** ..... 423
H.1 Complexity and latency ..... 423
H.1.1 Algorithm description ..... 423
H.1.2 Complexity ..... 423
H.1.3 Latency ..... 424
H.2 Experimental results ..... 425
H.3 Alternative low-delay implementations ..... 426

**Bibliography** ..... 428

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

# 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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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

Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23008 consists of the following parts, under the general title *Information technology — High efficiency coding and media delivery in heterogeneous environments*:

— *Part 1:* MPEG media transport (MMT)

— *Part 2:* High efficiency video coding

— *Part 3: 3D audio*

— *Part 4: MMT Reference and Conformance Software*

— *Part 5: Reference software for high efficiency video coding*

— *Part 8: HEVC conformance testing*

— *Part 10: MPEG media transport forward error correction (FEC) codes*

— *Part 11: MPEG media transport composition information*

— *Part 12: Image file format*

— *Part 13: MMT Implementation Guidelines*

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

 © ISO/IEC 2015 – All rights reserved

This corrected version of ISO 23008-3:2015 incorporates the following corrections: unreadable equations have been improved.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

# Introduction

3D sound systems are able to realize a significantly enhanced sound experience relative to current widespread 5.1 channel audio programs and playback systems. These systems demand high quality audio coding and error-free transmission in order to keep the timbre, sound localization and sound envelopment of the original audio program. Presentation over headphones with suitable spatialization are also considered.

This part of ISO/IEC 23008-3 “High Efficiency Coding and Media Delivery in Heterogeneous Environments — Part 3: 3D Audio” provides means for all scenarios where there is a need to compress a multi-channel audio program (e.g. 22.2 channel program) and to render it to the native target number of loudspeakers. In order to reach a wide market, a 3D Audio program is able to be downmixed to a lower hierarchy of loudspeakers, for example 10.1 or 8.1 channels. In addition, all scenarios support a level of random access to facilitate broadcast break-in, and “trick modes” such as fast forward when playing from stored media.

The main focus of this specification are applications such as audio for Home Theatres where the audio presentation is immersive, involving many loudspeakers (e.g. from 10 to more than 20) surrounding the listener and placed below, at and above ear-level. Moreover applications as Personal TV, TV for SmartPhones and Multi-channel Audio-only Programs are envisioned. These require that 3D Audio encoding/decoding systems are able to operate at low bitrates appropriate for efficient transmission over a cellular channel. At the same time the sense of envelopment and accurate sonic localization even for systems having a tablet-sized visual displays with speakers built into the device and headphone listening are maintained.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

# Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio

## 1 Scope

This part of ISO/IEC 23008-3 specifies technology which supports the efficient transmission of 3D audio signals and flexible rendering for the playback of 3D audio in a wide variety of listening scenarios. These include 3D home theater setups, 22.2 loudspeaker systems, automotive entertainment systems and playback over headphones connected to a tablet or smartphone.

## 2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensible for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 13818-1:2013, *Information technology — Generic Coding of moving pictures and associated audio information: Systems*

ISO/IEC 14496-3:2009, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 14496-11, *Information technology — Coding of audio-visual objects — Part 11: Scene description and application engine*

ISO/IEC 23001-8:2013, *Information technology — MPEG systems technologies — Part 8: Coding-independent code-points*

ISO/IEC 23001-8:2013/Amd.1, *Information technology — MPEG systems technologies — Part 8: Coding-independent code-points, AMENDMENT 1: New audio code points*

ISO/IEC 23003-1:2007, *Information technology — MPEG audio technologies — Part 1: MPEG Surround*

ISO/IEC 23003-2:2010, *Information technology — MPEG audio technologies — Part 2: Spatial Audio Object Coding (SAOC)*

ISO/IEC 23003-3:2012, *Information technology — MPEG audio technologies — Part 3: Unified speech and audio coding*

ISO/IEC 23003-4:2015, *Information technology — MPEG audio technologies — Part 4: Dynamic range control*

## 3 Terms, definitions and mnemonics

### 3.1 Terms and Definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14496-3:2009, 1.3 (Terms and definitions), in ISO/IEC 14496-3:2009, 1.4 (Symbols and abbreviations) and in ISO/IEC 23003-3:2012, 3.1 (Terms and definitions) apply.

### 3.2 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bitstream payload.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| bslbf | Bit string, left bit first, where “left” is the order in which bit strings are written in ISO/IEC 14496. Bit strings are written as a string of 1s and 0s within single quote marks, for example '1000 0001'. Blanks within a bit string are for ease of reading and have no significance. |
| uimsbf | Unsigned integer, most significant bit first. |
| vlclbf | Variable length code, left bit first, where “left” refers to the order in which the variable length codes are written. |
| tcimsbf | Two's complement integer, most significant (sign) bit first. |

# 4 Technical Overview

## 4.1 Decoder block diagram

The 3D Audio Codec System consists of an MPEG-H 3D Audio Core Codec for coding of channel, object and Higher Order Ambisonics (HOA) signals. The core codec is based on the MPEG-D USAC codec. To increase the efficiency for coding a large amount of objects, MPEG SAOC technology has been adopted. Several types of renderers perform the tasks of rendering objects to channels, rendering channels to a different loudspeaker setup, rendering HOA signals to the loudspeaker setup or rendering virtual loudspeaker channels or HOA components to headphones.

When object signals are explicitly transmitted or parametrically encoded using SAOC, the corresponding Object Metadata information is compressed and multiplexed into the 3D-Audio bitstream.

Figure 1 shows the different algorithmic blocks of the 3D-Audio system.

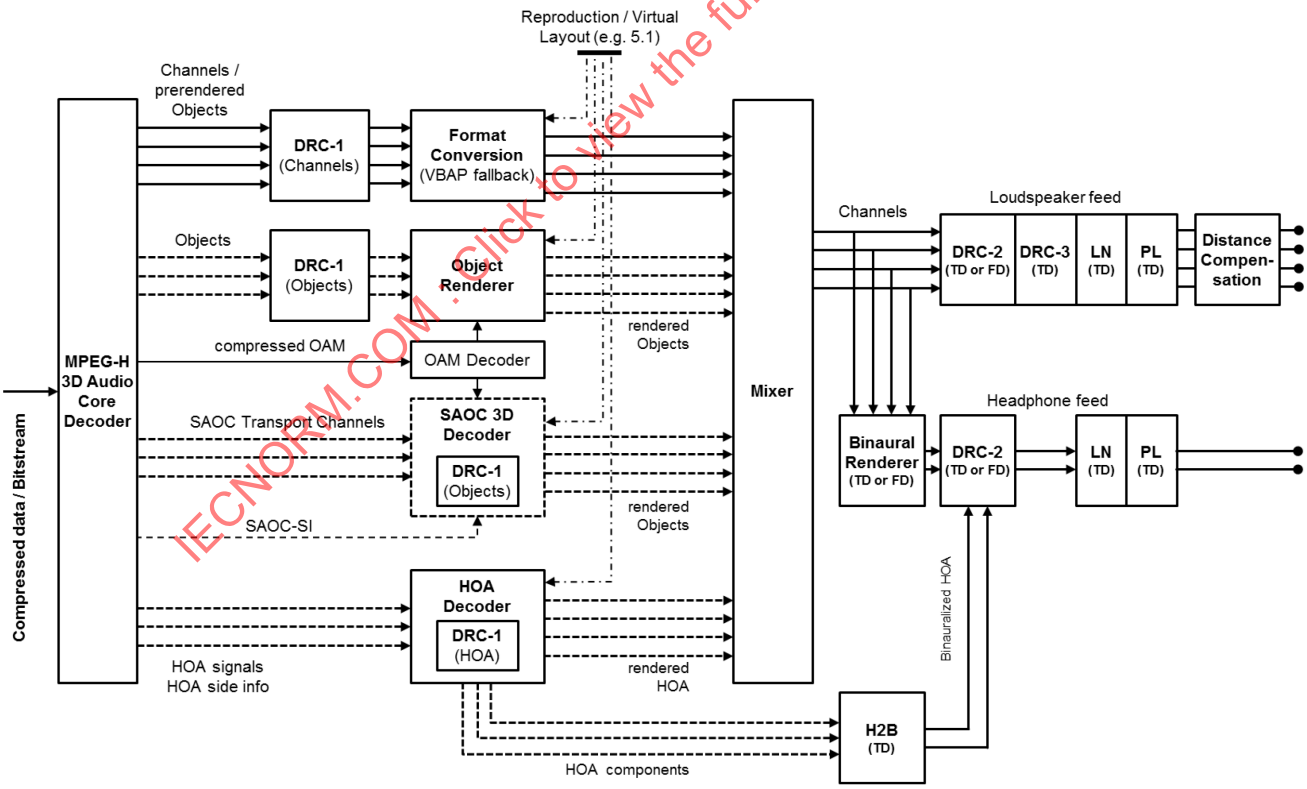



**Figure 1 — Block diagram of the 3D-Audio decoder.**
**(DRC: Dynamc Range Control, SAOC: Spatial Audio Object Coding, HOA: Higher Order Ambisonics, LN: Loudness Normalization, PL: Peak Limiter)**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

## 4.2 Overview over the codec building blocks

The MPEG-H 3DA Core Codec for loudspeaker-channel signals, discrete object signals, object downmix signals and pre-rendered signals is based on MPEG-D USAC technology. It handles the coding of the multitude of signals by creating channel- and object-mapping information based on the geometric and semantic information of the input's channel and object assignment. This mapping information describes how input channels and objects are mapped to channel elements (CPEs, SCEs, LFEs) and the corresponding information is transmitted to the decoder.

The coding of objects is possible in different ways, depending on the rate/distortion requirements and the interactivity requirements for the renderer. The following object coding variants are possible:

— Prerendered objects: Object signals are pre-rendered and mixed to multi-channel or HOA signals before encoding, as appropriate. The subsequent coding chain then operates on multi-channel or HOA signals.

— Discrete object waveforms: Objects are supplied as monophonic waveforms to the encoder. The encoder uses single channel elements SCEs to transmit the objects in addition to the channel signals. The decoded objects are rendered and mixed at the receiver side. Compressed object metadata information is transmitted to the receiver/renderer alongside.

— Parametric object waveforms: Object properties and their relation to each other are described by means of SAOC parameters. The downmix of the object signals is coded with the MPEG-H 3D Audio Core codec. The parametric information is transmitted alongside. The number of downmix channels is chosen depending on the number of objects and the overall data rate. Compressed object metadata information is transmitted to the SAOC renderer.

The SAOC Encoder and Decoder for object signals are based on MPEG SAOC technology. The system is capable of recreating, modifying and rendering a number of audio objects based on a smaller number of transmitted channels and additional parametric data (OLDs, IOCs, DMGs).

The SAOC decoder reconstructs the object/channel signals from the decoded SAOC transport channels and parametric information, and generates the output audio scene based on the reproduction layout, the decompressed object metadata information and optionally on the user interaction information.

The Object Metadata Codec efficiently codes the associated metadata that specifies the geometrical position and volume of each object in 3D space by quantization of the object properties in time and space. The compressed object metadata is transmitted to the receiver as side information.

The Object Renderer utilizes the compressed object metadata to generate object waveforms according to the given reproduction format. Each object is rendered to certain output channels according to its metadata. The output of this block results from the sum of the partial results.

The Loudspeaker Renderer converts between the transmitted channel configuration and the desired reproduction format. It is thus called 'format converter'. In case of conversions to lower numbers of output channels it creates downmixes. The system automatically generates optimized downmix matrices for the given combination of input and output formats and applies these matrices in a downmix process. The format converter allows for standard loudspeaker configurations as well as for random configurations with non-standard loudspeaker positions.

The Higher Order Ambisonics (HOA) Decoder/Renderer reconstructs the HOA coefficient signals based on the HOA transport channels decoded by the 3D Audio Core Decoder and the HOA specific side information. The coding principle is based on a separate transmission of so-called predominant sounds and ambient sound scene components. Subsequently the HOA renderer generates the loudspeaker channel feeds based on the reproduction layout.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

If two or more groups of channel based content, discrete/parametric objects or HOA based content are decoded, the corresponding waveforms are delay-aligned and sample-wise added by the Mixer before providing the resulting waveforms (or before feeding them to a postprocessor module such as the binaural renderer, DRC-2, DRC-3, the peak limiter PL, or the loudspeaker distance compensation).

The Binaural Renderer module produces a binaural downmix of the multichannel audio material, such that each input channel is represented by a virtual sound source. The processing is conducted frame-wise in the QMF domain or in the time domain. The binauralization is based on measured binaural room impulse responses.

Virtual layout information fed from an application shall be consistent with the corresponding BRIR set provided (expressed with MeasurementSetup, see in Table 157), in the sense that the set of positions corresponding to the virtual layout is a subset of the set of positions corresponding to the BRIRs.

## 4.3 Efficient combination of decoder processing blocks in time domain and QMF domain

There are numerous processing blocks in the MPEG-H 3DA decoding, rendering and processing framework. In general these blocks can be classified into different classes:

— Block operates in time domain (**TD**)

— Block operates in frequency domain (**FD**) – also called QMF domain

— Block is "**neutral**" and can operate in FD or TD.
Signal-processing-wise the same operation is carried out in either FD or TD. Operation in a different domain may cause no difference or only a small but perceptually negligible difference to the output signal

Table 1 lists all blocks with corresponding processing domain. For the sake of clarification the functional blocks are grouped into semantically differentiable contexts, which follow the MPEG-H 3D Audio signal processing as shown in Figure 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Table 1 — MPEG-H 3DA functional blocks and internal processing domain**

| Processing Context | Functional Block | Processing Domain |
|---|---|---|
| Audio Core | MPEG-H 3D Audio Core Coder | **FD** or **TD**[1)] |
| Rendering | DRC-1 | if multiband: **FD**<br>else: **neutral** |
| | Format Converter (FC) | **FD** |
| | Object Renderer | **neutral** |
| | SAOC 3D Decoder | **FD** |
| | HOA Decoder | **TD** |
| Mixing | Mixer | **neutral** |
| Post-processing | DRC-2 | if multiband: **FD**<br>else: **neutral** |
| | FD Binauralizer | **FD** |
| | TD Binauralizer | **TD** |
| End of chain | DRC-3 (only singleband) | **TD** |
| | Loudness Normalization | **TD** |
| | Peak Limiter | **TD** |
| | LS Distance Compensation | **TD** |

[1)] depends on bit stream configuration


© ISO/IEC 2015 – All rights reserved

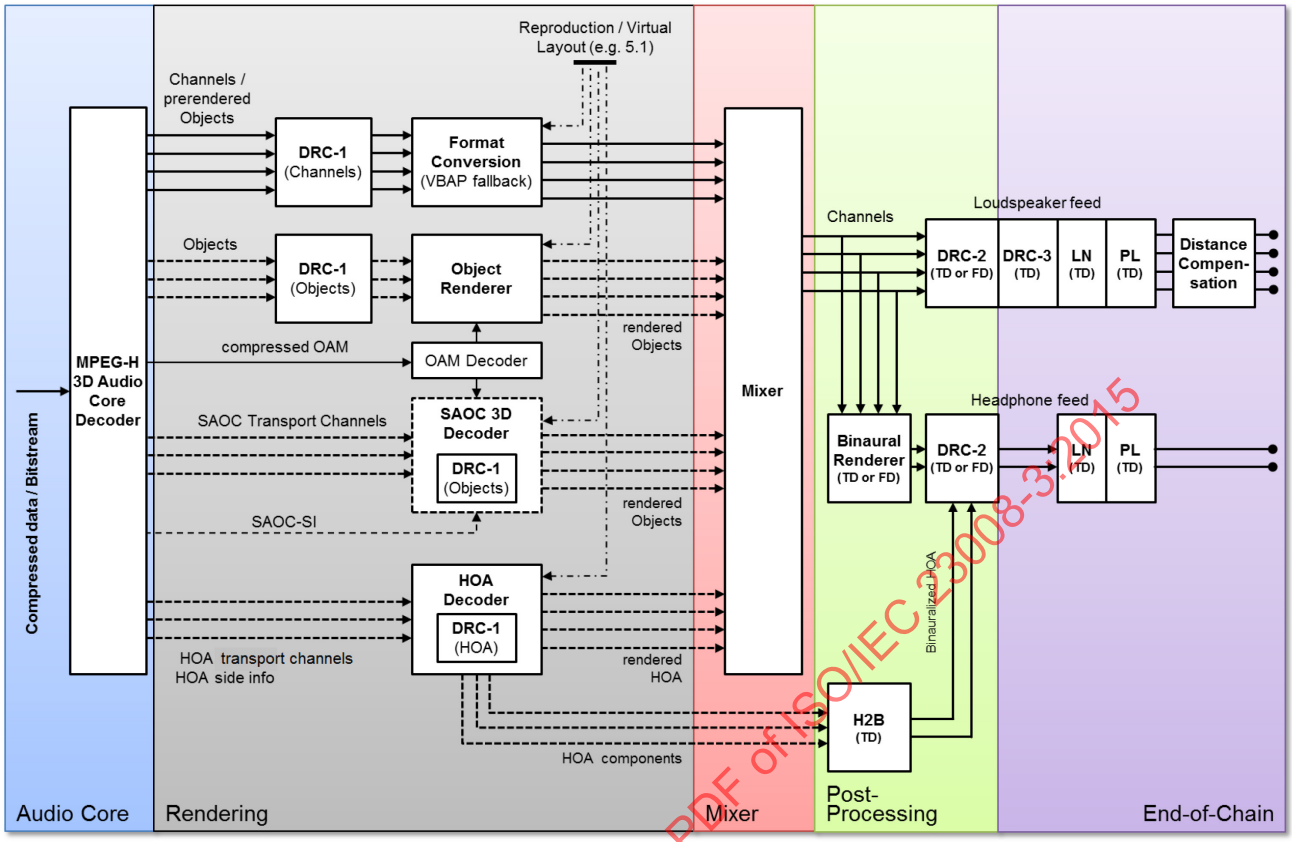



**Figure 2 — MPEG-H 3D audio decoder overview with signal processing context**

Operating blocks that are directly connected in the signal chain and which operate in the *same* domain can interface in that common domain.

Operating blocks that are directly connected in the signal chain and which operate in *different* domains require a transformation block which turns one signal representation into another, i.e. a QMF Analysis (TD to FD) or a QMF Synthesis (FD to TD). This transform causes additional delay and hence needs to be taken into account when determining the overall signal processing delay.

The following subclause describes formal generic rules which determine how the various functional can be connected.

## 4.4 Rule set for determining processing domains

### 4.4.1 Audio Core Codec, Processing Domain

The incoming bit stream configuration determines which core decoder tools are active and whether parts of the core codec operate in QMF domain / FD.

- If last decoding stage is in QMF domain (SBR and/or MPS212 active) (typical for mid to low bit rate coding)
  - formal Audio Core Codec processing domain is **FD**

- If last decoding stage is **not** in QMF domain (neither SBR nor MPS212 active) (typical for high rate coding)
  - formal Audio Core Codec processing domain is **TD**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

#### 4.4.2 Mixing

— If the Audio Core Codec processing domain is **TD**

  — If any block in Rendering Context requires operation in **FD**
  AND any block in Post-processing Context requires operation in **FD**
    — then: Mixer shall operate in **FD**

  — else:
    — Mixer shall operate in **TD**

— If the Audio Core Codec processing domain is **FD**

  — If any block in Rendering Context requires operation in **TD**
  AND **no** blocks in Post-processing Context require operation in **FD**
    — then: Mixer shall operate in **TD**

  — else:
    — Mixer shall operate in **FD**

#### 4.4.3 Audio Core Codec, Interface Domain to Rendering

— If the Audio Core Codec processing domain is **TD**,

  — If subsequent rendering block requires operation in **FD**
    — apply QMF analysis and output **FD** for this rendering block

  — else
    — output **TD**

— If the Audio Core Codec processing domain is **FD**,

  — If subsequent rendering block requires operation in **TD**
    — apply QMF synthesis and output **TD** for this rendering block

  — else
    — output **FD**

#### 4.4.4 Rendering Context

Rendering paths are considered individually:

— If Rendering block operates in **FD** and Mixer operates in **TD**
  — apply QMF synthesis after renderer and output **TD**

— If Rendering block operates in **TD** and Mixer operates in **FD**
  — apply QMF analysis after renderer and output **FD**

— else
  — keep processing domain

#### 4.4.5 Post-Processing Context

— If Mixer operates **TD**

  — If **any** block in Post-processing Context requires operation in **FD**
    — convert to **FD** immediately after Mixer;
    — after the **FD** operation convert to **TD**, i.e. prior to End-of-Chain Context

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

- else
  - stay in **TD**

- If Mixer operates in **FD**

  - If **no** block in Post-processing Context requires operation in **FD**
    - convert to **TD** prior to DRC-2

  - else
    - convert to **TD** prior to End-of-Chain Context

#### 4.4.6 End-of-Chain Context

All blocks shall operate in **TD**.

# 5 MPEG-H 3D Audio Core decoder

## 5.1 Terms and Definitions

### 5.1.1 Joint Stereo

The MDCT domain based joint stereo coding tool with the possibility of Complex Stereo Prediction as defined in ISO/IEC 23003-3:2012, subclause 7.7.

### 5.1.2 MPEG Surround based stereo (MPS 212)

The MPEG Surround 2-1-2 based stereo tool working in QMF domain as defined in ISO/IEC 23003-3:2012, subclause 7.11, with the possibility of using residual coding (Unified Stereo) as specified in ISO/IEC 23003-3:2012, Annex B.21.

## 5.2 Syntax

### 5.2.1 General

The bit stream syntax is based on ISO/IEC 23003-3:2012, clause 5.

Modifications and amendments to the existing bit stream syntax are listed below.

In environments that require byte alignment, MPEG-H 3D Audio configuration elements or payload elements that are not an integer number of bytes in length are padded at the end to achieve an integer byte count.

### 5.2.2 Decoder configuration

#### 5.2.2.1 General Configuration Syntax

**Table 2 — Syntax of mpegh3daConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daConfig() | | |
| { | | |
| **mpegh3daProfileLevelIndication** | **8** | **uimsbf** |
| **usacSamplingFrequencyIndex;** | **5** | **bslbf** |
| if (usacSamplingFrequencyIndex == 0x1f) { | | |
| **usacSamplingFrequency;** | **24** | **uimsbf** |
| } | | |
| **coreSbrFrameLengthIndex;** | **3** | **uimsbf** |
| **coreQmfDelayCompensation;** | **1** | **uimsbf** |
| **receiverDelayCompensation;** | **1** | **uimsbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| referenceLayout = SpeakerConfig3d(); | | |
| FrameworkConfig3d(); | | |
| mpegh3daDecoderConfig(); | | |
| if (**usacConfigExtensionPresent**) { | **1** | **uimsbf** |
| mpegh3daConfigExtension(); | | |
| } | | |
| } | | |

**Table 3 — Syntax of FrameworkConfig3d()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| FrameworkConfig3d() | | |
| { | | |
| Signals3d(); | | |
| } | | |

**Table 4 — Syntax of Signals3d()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| Signals3d() | | |
| { | | |
| numAudioChannels = 0; | | |
| numAudioObjects = 0; | | |
| numSAOCTransportChannels = 0; | | |
| numHOATransportChannels = 0; | | |
| | | |
| **bsNumSignalGroups;** | **5** | **uimsbf** |
| for ( grp = 0; grp < bsNumSignalGroups + 1 ; grp++ ) { | | |
| signal_groupID[grp] = grp; | | |
| differsFromReferenceLayout[grp] = 0; | | |
| **signalGroupType[grp];** | **3** | **bslbf** |
| bsNumberOfSignals[grp] = escapedValue(5, 8, 16); | | |
| if ( SignalGroupType[grp] == SignalGroupTypeChannels ) { | | |
| numAudioChannels += bsNumberOfSignals[grp] + 1; | | |
| **differsFromReferenceLayout[grp]**; | **1** | **bslbf** |
| if(differsFromReferenceLayout[grp]) { | | |
| audioChannelLayout[grp] = SpeakerConfig3d(); | | |
| } | | |
| else { | | |
| audioChannelLayout[grp] = referenceLayout; | | |
| } | | |
| } | | |
| if ( SignalGroupType[grp] == SignalGroupTypeObject ) { | | |
| numAudioObjects += bsNumberOfSignals[grp] + 1; | | |
| } | | |
| if ( SignalGroupType[grp] == SignalGroupTypeSAOC ) { | | |
| numSAOCTransportChannels += bsNumberOfSignals[grp] + 1; | | |
| } | | |
| if ( SignalGroupType[grp] == SignalGroupTypeHOA ) { | | |
| numHOATransportChannels += bsNumberOfSignals[grp] + 1; | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
}
```

#### 5.2.2.2 Loudspeaker Configuration Syntax

**Table 5 — Syntax of SpeakerConfig3d()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| SpeakerConfig3d() | | |
| { | | |
| **speakerLayoutType;** | **2** | **uimsbf** |
| if (speakerLayoutType == 0 ) { | | |
| **CICPspeakerLayoutIdx;** | **6** | **uimsbf** |
| } | | |
| else { | | |
| **numSpeakers** = escapedValue(5, 8, 16) + 1; | | |
| if (speakerLayoutType == 1 ) { | | |
| for (i = 0; i < numSpeakers; i++) { | | |
| **CICPspeakerIdx;** | **7** | **uimsbf** |
| } | | |
| } | | |
| if (speakerLayoutType == 2 ) { | | |
| mpegh3daFlexibleSpeakerConfig(numSpeakers); | | |
| } | | |
| } | | |
| } | | |

**Table 6 — Syntax of mpegh3daFlexibleSpeakerConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daFlexibleSpeakerConfig(numSpeakers) | | |
| { | | |
| **angularPrecision**; | **1** | **uimsbf** |
| for (i = 0; i < numSpeakers; i++) { | | |
| mpegh3daSpeakerDescription(); | | |
| if ((AzimuthAngle != 0°) && (AzimuthAngle != 180°)) { | | NOTE |
| **alsoAddSymmetricPair**; | **1** | **uimsbf** |
| if (alsoAddSymmetricPair) { | | |
| (*also add the speaker with the opposite AzimuthDirection*); | | |
| i++; | | |
| } | | |
| } | | |
| } | | |
| } | | |
| NOTE: The value of AzimuthAngle can be calculated using Table 7. | | |

**Table 7 — Syntax of mpegh3daSpeakerDescription()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daSpeakerDescription() | | |
| { | | |
| **isCICPspeakerIdx** | **1** | **uimsbf** |
| if (isCICPspeakerIdx) { | | |
| **CICPspeakerIdx**; | **7** | **uimsbf** |
| } | | |
| else { | | |
| **ElevationClass**; | **2** | **uimsbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | | |
|---|---|---|---|
| if (ElevationClass == 3) { | | | |
| **ElevationAngleIdx**; | **5, 7** | **uimsbf** |
| if (ElevationAngle != 0°) { | | NOTE |
| **ElevationDirection**; | **1** | **uimsbf** |
| } | | |
| } | | |
| **AzimuthAngleIdx**; | **6, 8** | **uimsbf** |
| if ((AzimuthAngle != 0°) && (AzimuthAngle != 180°)) { | | NOTE |
| **AzimuthDirection**; | **1** | **uimsbf** |
| } | | |
| **isLFE;** | **1** | **uimsbf** |
| } | | |
| } | | |

NOTE: The number of bits for ElevationAngleIdx and AzimuthAngleIdx depends on the value of angularPrecision according to Table 42. The value of ElevationAngle and AzimuthAngle can be derived from Table 44 and Table 46, respectively. In case isCICPspeakerIdx is one, ElevationAngle and AzimuthAngle are signaled by means of a LoudspeakerGeometry as defined in ISO/MPEG 23001-8.

#### 5.2.2.3 Core Decoder Configuration

**Table 8 — Syntax of mpegh3daDecoderConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daDecoderConfig() | | |
| { | | |
| numElements = escapedValue(4,8,16) + 1**;** | | |
| **elementLengthPresent** | **1** | |
| | | |
| for (elemIdx=0; elemIdx<numElements; ++elemIdx) { | | |
| **usacElementType[**elemIdx**]** | **2** | **uimsbf** |
| switch (usacElementType[elemIdx]) { | | |
| case ID_USAC_SCE: | | |
| mpegh3daSingleChannelElementConfig(sbrRatioIndex); | | |
| break; | | |
| case ID_USAC_CPE: | | |
| mpegh3daChannelPairElementConfig(sbrRatioIndex); | | |
| break; | | |
| case ID_USAC_LFE: | | |
| mpegh3daLfeElementConfig(); | | |
| break; | | |
| case ID_USAC_EXT: | | |
| mpegh3daExtElementConfig(); | | |
| break; | | |
| } | | |
| } | | |

NOTE: mpegh3daSingleChannelElementConfig(), mpegh3daChannelPairElementConfig(), mpegh3daLfeElementConfig() and mpegh3daExtElementConfig() signaled at position elemIdx refer to the corresponding elements in mpegh3daFrame() at the respective position elemIdx.

**Table 9 — Syntax of mpegh3daSingleChannelElementConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daSingleChannelElementConfig(sbrRatioIndex) | | |
| { | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
    mpegh3daCoreConfig();
    if (sbrRatioIndex > 0) {
        SbrConfig();
    }
}
```

**Table 10 — Syntax of mpegh3daChannelPairElementConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daChannelPairElementConfig(sbrRatioIndex) | | |
| { | | |
| mpegh3daCoreConfig(); | | |
| if (enhancedNoiseFilling) { | | |
| **igfIndependentTiling;** | **1** | **bslbf** |
| } | | |
| if (sbrRatioIndex > 0) { | | |
| SbrConfig(); | | |
| **stereoConfigIndex;** | **2** | **uimsbf** |
| } else { | | |
| stereoConfigIndex = 0; | | |
| } | | |
| if (stereoConfigIndex > 0) { | | |
| Mps212Config(stereoConfigIndex); | | |
| } | | |
| **qceIndex;** | **2** | **uimsbf** |
| If(qceIndex > 0) { | | |
| **shiftIndex0;** | **1** | **uimsbf** |
| if(shiftIndex0 > 0) { | | |
| **shiftChannel0;** | **nBits**[1)] | |
| } | | |
| } | | |
| **shiftIndex1;** | **1** | **uimsbf** |
| if(shiftIndex1 > 0) { | | |
| **shiftChannel1;** | **nBits**[1)] | |
| } | | |
| } | | |

[1)] **nBits** = floor(log2(numAudioChannels + numAudioObjects + numHOATransportChannels + numSAOCTransportChannels - 1)) + 1

**Table 11 — Syntax of mpegh3daCoreConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daCoreConfig() | | |
| { | | |
| **tw_mdct;** | **1** | **bslbf** |
| **noiseFilling;** | **1** | **bslbf** |
| if (**enhancedNoiseFilling**) { | **1** | **bslbf** |
| **igfUseEnf;** | **1** | **bslbf** |
| **igfUseWhitening;** | **1** | **bslbf** |
| **igfAfterTnsSynth;** | **1** | **bslbf** |
| **igfStartIndex;** | **4** | **uimsbf** |
| **igfUseHighRes;** | **1** | **bslbf** |
| **igfStopIndex;** | **3** | **uimsbf** |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 12 — Syntax of mpegh3daLfeElementConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daLfeElementConfig()<br>{<br>  tw_mdct = 0;<br>  noiseFilling = 0;<br>  enhancedNoiseFilling = 0;<br>} | | |

**Table 13 — Syntax of mpegh3daExtElementConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daExtElementConfig()<br>{<br>  usacExtElementType = escapedValue(4, 8, 16);<br>  usacExtElementConfigLength = escapedValue(4, 8, 16);<br><br>  if (**usacExtElementDefaultLengthPresent**) {<br>  usacExtElementDefaultLength = escapedValue(8, 16, 0) + 1;<br>  } else {<br>  usacExtElementDefaultLength = 0;<br>  }<br><br>  **usacExtElementPayloadFrag;**<br><br>  switch (usacExtElementType) {<br>  case ID_EXT_ELE_FILL:<br>  /* No configuration element */<br>  break;<br>  case ID_EXT_ELE_MPEGS:<br>  SpatialSpecificConfig();<br>  break;<br>  case ID_EXT_ELE_SAOC:<br>  SAOCSpecificConfig();<br>  break;<br>  case ID_EXT_ELE_AUDIOPREROLL:<br>  /* No configuration element */<br>  break;<br>  case ID_EXT_ELE_UNI_DRC:<br>  mpegh3daUniDrcConfig();<br>  break;<br>  case ID_EXT_ELE_OBJ_METADATA:<br>  ObjectMetadataConfig();<br>  break;<br>  case ID_EXT_ELE_SAOC_3D:<br>  SAOC3DSpecificConfig();<br>  break;<br>  case ID_EXT_ELE_HOA:<br>  HOAConfig();<br>  break;<br>  case ID_EXT_ELE_FMT_CNVRTR<br>  /* No configuration element */<br>  break;<br>  default:<br>  while (usacExtElementConfigLength--) {<br>  **tmp;**<br>  } | <br><br><br><br>**1**<br><br><br><br>**1**<br><br><br>

```
            break;
        }
    }
```

NOTE: The default entry for the usacExtElementType is used for unknown extElementTypes so that legacy decoders can cope with future extensions.

**Table 14 — Syntax of mpegh3daConfigExtension()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daConfigExtension()<br>{<br>  numConfigExtensions = escapedValue(2,4,8) + 1;<br><br>  for (confExtIdx=0; confExtIdx<numConfigExtensions; confExtIdx++) {<br>  usacConfigExtType[confExtIdx] = escapedValue(4,8,16);<br>  usacConfigExtLength[confExtIdx] = escapedValue(4,8,16);<br><br>  switch (usacConfigExtType[confExtIdx]) {<br>  case ID_CONFIG_EXT_FILL:<br>  while (usacConfigExtLength[confExtIdx]--) {<br>  **fill_byte[i];** /* should be '10100101' */ | **8** | **uimsbf** |
| }<br>  break;<br>  case ID_CONFIG_EXT_DOWNMIX:<br>  downmixConfig();<br>  break;<br>  case ID_CONFIG_EXT_LOUDNESS_INFO:<br>  mpegh3daLoudnessInfoSet();<br>  break;<br>  case ID_CONFIG_EXT_AUDIOSCENE_INFO:<br>  mae_AudioSceneInfo();<br>  break;<br>  case ID_CONFIG_EXT_HOA_MATRIX:<br>  HoaRenderingMatrixSet();<br>  break;<br>  default:<br>  while (usacConfigExtLength[confExtIdx]--) {<br>  **tmp;** | **8** | **uimsbf** |
| }<br>  break;<br>  }<br>  }<br>} | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Table 15 — References to USAC configuration syntactic elements**

| **Syntax of** | **defined in** |
|---|---|
| escapedValue() | ISO/IEC 23003-3:2012, 5.2, Table 16 |
| SbrConfig() | ISO/IEC 23003-3:2012, 5.2, Table 11 |
| Mps212Config() | ISO/IEC 23003-3:2012, 5.2, Table 13 |
| SAOCSpecificConfig() | ISO/IEC 23003-2:2010, 6.1, Table 5 |
| SpatialSpecificConfig() | ISO/IEC 23003-1:2007, 5.1, Table 5 |

© ISO/IEC 2015 – All rights reserved

#### 5.2.2.4 Syntax of downmix matrix elements

**Table 16 — Syntax of downmixConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| downmixConfig () | | |
| { | | |
| **downmixConfigType**; | **2** | **uimsbf** |
| if (downmixConfigType == 0 \|\| downmixConfigType == 2) { | | |
| **passiveDownmixFlag** | **1** | **uimsbf** |
| if (passiveDownmixFlag ==0) { | | |
| **phaseAlignStrength** | **3** | **uimsbf** |
| } | | |
| **immersiveDownmixFlag** | **1** | **uimsbf** |
| } | | |
| if (downmixConfigType == 1 \|\| downmixConfigType == 2) { | | |
| DownmixMatrixSet() | | |
| } | | |
| } | | |

**Table 17 — Syntax of DownmixMatrixSet()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| DownmixMatrixSet() | | |
| { | | |
| **downmixIdCount**; | **5** | **uimsbf** |
| for (k=0; k< downmixIdCount; ++k) { | | |
| **downmixId;** | **7** | **uimsbf** |
| **downmixType**; | **2** | **uimsbf** |
| if (downmixType == 0) { | | |
| **CICPspeakerLayoutIdx;** | **6** | **uimsbf** |
| } else if (downmixType == 1) { | | |
| **CICPspeakerLayoutIdx;** | **6** | **uimsbf** |
| **bsDownmixMatrixCount[k]** = escapedValue(1,3,0); | **1..4** | |
| for (l=0; l< bsDownmixMatrixCount[k]+1; ++l) { | | |
| **bsNumAssignedGroupIDs[k][l]** = escapedValue(1,4,4); | **1..9** | |
| for (m=0; m<bsNumAssignedGroupIDs[k][l]+1; ++m) { | | |
| **signal_groupID[k][l][m];** | **5** | **uimsbf** |
| } | | |
| **DmxMatrixLenBits** = escapedValue(8,8,12); | **8..28** | |
| DownmixMatrix(inputConfig(audioChannelConfig), inputCount(audioChannelConfig), outputConfig(CICPspeakerLayoutIdx), outputCount(CICPspeakerLayoutIdx) ); | **DmxMatrixLenBits** | |
| } | | |
| } | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 18 — Syntax of DownmixMatrix()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| DownmixMatrix(inputConfig, inputCount, outputConfig, outputCount) | | |
| { | | |
| **equalizerPresent**; | **1** | **uimsbf** |
| if (equalizerPresent) { | | |
| EqualizerConfig (inputConfig, inputCount); | | |
| } | | |
| **precisionLevel**; | **2** | **uimsbf** |
| maxGain = escapedValue(3, 4, 0); | | |
| minGain = (-1) × (escapedValue(4, 5, 0) + 1); | | |
| (compactInputConfig, compactInputCount) = ConvertToCompactConfig(inputConfig, inputCount); | | |
| (compactOutputConfig, compactOutputCount) = ConvertToCompactConfig(outputConfig, outputCount); | | |
| **isAllSeparable**; | **1** | **uimsbf** |
| if (!isAllSeparable) { | | |
| for (i = 0; i < compactOutputCount; i++) { | | |
| if (compactOutputConfig[i].pairType == SYMMETRIC) { | | |
| **isSeparable[i]**; | **1** | **uimsbf** |
| } | | |
| } | | |
| } else { | | |
| for (i = 0; i < compactOutputCount; i++) { | | |
| if (compactOutputConfig[i].pairType == SYMMETRIC) { | | |
| isSeparable[i] = 1; | | |
| } | | |
| } | | |
| } | | |
| **isAllSymmetric**; | **1** | **uimsbf** |
| if (!isAllSymmetric) { | | |
| for (i = 0; i < compactOutputCount; i++) { | | |
| **isSymmetric[i]**; | **1** | **uimsbf** |
| } | | |
| } else { | | |
| for (i = 0; i < compactOutputCount; i++) { | | |
| isSymmetric[i] = 1; | | |
| } | | |
| } | | |
| **mixLFEOnlyToLFE**; | **1** | **uimsbf** |
| **rawCodingCompactMatrix**; | **1** | **uimsbf** |
| if (rawCodingCompactMatrix) { | | |
| for (i = 0; i < compactInputCount; i++) { | | |
| for (j = 0; j < compactOutputCount; j++) { | | |
| if (!mixLFEOnlyToLFE \|\| (compactInputConfig[i].isLFE == compactOutputConfig[j].isLFE)) { | | |
| **compactDownmixMatrix[i][j]**; | **1** | **uimsbf** |
| } else { | | |
| compactDownmixMatrix[i][j] = 0; | | |
| } | | |
| } | | |
| } | | |
| } else { | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
        totalCount = CalculateTotalCount();
        useCompactTemplate;                                          1          uimsbf
        nBits = 3; if (totalCount >= 256) nBits = 4;
        runLGRParam;                                                 nBits      uimsbf
        count = 0;
        while (count < totalCount) {
          zeroRunLength;                                             varies     bslbf
           /* limited Golomb-Rice using p = runLGRparam  and N = totalCount+1*/
          flatCompactMatrix[count .. count + zeroRunLength] = {0, ..., 0, 1};
          count += zeroRunLength + 1;
        }

        count = 0;
        compactTemplate =
                        FindCompactTemplate(inputConfig, inputCount, outputConfig, outputCount);
        for (i = 0; i < compactInputCount; i++) {
          for (j = 0; j < compactOutputCount; j++) {
            if (mixLFEOnlyToLFE && compactInputConfig[i].isLFE &&
                  compactOutputConfig[j].isLFE) {
              compactDownmixMatrix[i][j];                            1          uimsbf
            } else if (mixLFEOnlyToLFE &&
                        (compactInputConfig[i].isLFE || compactOutputConfig[j].isLFE))  {
              compactDownmixMatrix[i][j] = 0;
            } else {
              compactDownmixMatrix[i][j] = flatCompactMatrix[count++];
              if (useCompactTemplate) {
                  compactDownmixMatrix[i][j] ^= compactTemplate[i][j];
              }
            }
          }
        }

    fullForAsymmetricInputs;                                         1          uimsbf
    rawCodingNonzeros;                                               1          uimsbf
    if (!rawCodingNonzeros) {
      gainLGRParam;                                                  3          uimsbf
      gainTableSize = generateGainTable(maxGain, minGain,
precisionLevel);
    }
    for (i = 0; i < compactInputCount; i++) {
      iType = compactInputConfig[i].pairType;
      for (j = 0; j < compactOutputCount; j++) {
        oType = compactOutputConfig[j].pairType;
        i1 = compactInputConfig[i].originalPosition;
        o1 = compactOutputConfig[j].originalPosition;

        if ((iType != SYMMETRIC) && (oType != SYMMETRIC)) {
          downmixMatrix[i1][o1] = 0.0;
          if (!compactDownmixMatrix[i][j]) continue;

          downmixMatrix[i1][o1] = DecodeGainValue();
        } else if (iType != SYMMETRIC) {
          o2 = compactOutputConfig[j].SymmetricPair.originalPosition;
          downmixMatrix[i1][o1] = 0.0;
          downmixMatrix[i1][o2] = 0.0;
          if (!compactDownmixMatrix[i][j]) continue;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
            downmixMatrix[i1][o1] = DecodeGainValue();
            useFull = (iType == ASYMMETRIC) &&
fullForAsymmetricInputs;
            if (isSymmetric[j] && !useFull) {
               downmixMatrix[i1][o2] = downmixMatrix[i1][o1];
            } else {
               downmixMatrix[i1][o2] = DecodeGainValue();
            }
         } else if (oType != SYMMETRIC) {
            i2 = compactInputConfig[i].SymmetricPair.originalPosition;
            downmixMatrix[i1][o1] = 0.0;
            downmixMatrix[i2][o1] = 0.0;
            if (!compactDownmixMatrix[i][j]) continue;

            downmixMatrix[i1][o1] = DecodeGainValue();
            if (isSymmetric[j]) {
               downmixMatrix[i2][o1] = downmixMatrix[i1][o1];
            } else {
               downmixMatrix[i2][o1] = DecodeGainValue();
            }
         } else {
            i2 = compactInputConfig[i].SymmetricPair.originalPosition;
            o2 = compactOutputConfig[j].SymmetricPair.originalPosition;
            downmixMatrix[i1][o1] = 0.0;
            downmixMatrix[i1][o2] = 0.0;
            downmixMatrix[i2][o1] = 0.0;
            downmixMatrix[i2][o2] = 0.0;
            if (!compactDownmixMatrix[i][j]) continue;

            downmixMatrix[i1][o1] = DecodeGainValue();
            if (isSeparable[j] && isSymmetric[j]) {
               downmixMatrix[i2][o2] = downmixMatrix[i1][o1];
            } else if (!isSeparable[j] && isSymmetric[j]) {
               downmixMatrix[i1][o2] = DecodeGainValue();
               downmixMatrix[i2][o1] = downmixMatrix[i1][o2];
               downmixMatrix[i2][o2] = downmixMatrix[i1][o1];
            } else if (isSeparable[j] && !isSymmetric[j]) {
               downmixMatrix[i2][o2] = DecodeGainValue();
            } else {
               downmixMatrix[i1][o2] = DecodeGainValue();
               downmixMatrix[i2][o1] = DecodeGainValue();
               downmixMatrix[i2][o2] = DecodeGainValue();
            }
         }
      }
   }
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 19 — Syntax of DecodeGainValue()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| DecodeGainValue()<br>{<br>  if (rawCodingNonzeros) {<br>  nAlphabet = ((maxGain - minGain) * 2 ^ precisionLevel) + 2;<br>  gainValueIndex = ReadRange(nAlphabet);<br>  gainValue = maxGain - gainValueIndex / 2 ^ precisonLevel;<br>  } else {<br>  **gainValueIndex**; | **varies** | **bslbf** |
| /* limited Golomb-Rice using p = gainLGRParam and N = gainTableSize*/<br>  gainValue = gainTable[gainValueIndex];<br>  }<br>  if (gainValue < minGain) gainValue = -infinity;<br>  return gainValue;<br>} | | |

**Table 20 — Syntax of ReadRange()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| ReadRange(alphabetSize)<br>{<br>  nBits = floor(log2(alphabetSize));<br>  nUnused = 2 ^ (nBits + 1) - alphabetSize;<br>  **range**; | **nBits** | **uimsbf** |
| if (range >= nUnused) {<br>  **rangeExtra**; | **1** | **uimsbf** |
| range = range * 2 - nUnused + rangeExtra;<br>  }<br>  return range;<br>} | | |

**Table 21 — Syntax of EqualizerConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| EqualizerConfig(inputConfig, inputCount)<br>{<br>  numEqualizers = escapedValue(3, 5, 0) + 1;<br><br>  **eqPrecisionLevel**; | **2** | **uimsbf** |
| **eqExtendedRange**; | **1** | **uimsbf** |
| for (i = 0; i < numEqualizers; i++) {<br>  numSections = escapedValue(2, 4, 0) + 1;<br>  lastCenterFreqP10 = 0;<br>  lastCenterFreqLd2 = 10;<br>  maxCenterFreqLd2 = 99;<br>  for (j = 0; j < numSections; j++) {<br>  centerFreqP10 = lastCenterFreqP10 + ReadRange(4 - lastCenterFreqP10);<br>  if (centerFreqP10 > lastCenterFreqP10) { lastCenterFreqLd2 = 10;}<br>  if (centerFreqP10 == 3) { maxCenterFreqLd2 = 24; }<br>  centerFreqLd2 = lastCenterFreqLd2 +<br>  ReadRange(1 + maxCenterFreqLd2 - lastCenterFreqLd2);<br>  lastCenterFreqP10 = centerFreqP10;<br>  lastCenterFreqLd2 = centerFreqLd2; | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| **qFactorIndex**; | **5** | **uimsbf** |
| if (qFactorIndex > 19) { | | |
| **qFactorExtra**; | **3** | **uimsbf** |
| } | | |
| cgBits = 4 + eqExtendedRange + eqPrecisionLevel; | | |
| **centerGainIndex**; | **cgBits** | **uimsbf** |
| } | | |
| sgBits = 4 + eqExtendedRange + min(eqPrecisionLevel + 1, 3); | | |
| **scalingGainIndex**; | **sgBits** | **uimsbf** |
| } | | |
| for (i = 0; i < inputCount; i++) { | | |
| **hasEqualizer[i]**; | **1** | **uimsbf** |
| if (hasEqualizer[i]) { | | |
| equalizerIndex[i] = ReadRange(numEqualizers); | | |
| } else { | | |
| equalizerIndex[i] = -1; | | |
| } | | |
| } | | |
| } | | |

#### 5.2.2.5 Syntax of HOA matrix elements

**Table 22 — Syntax of HoaRenderingMatrixSet()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HoaRenderingMatrixSet() | | |
| { | | |
| **numHoaRenderingMatrices**; | **5** | **uimsbf** |
| for (k=0; k< numHoaRenderingMatrices; ++k) { | | |
| **HoaRenderingMatrixId;** | **7** | **uimsbf** |
| **CICPspeakerLayoutIdx;** | **6** | **uimsbf** |
| **HoaMatrixLenBits** = escapedValue(8,8,12); | **8..28** | |
| HoaRenderingMatrix(NumOfHoaCoeffs, outputConfig(CICPspeakerLayoutIdx), outputCount(CICPspeakerLayoutIdx) ); | **HoaMatrixLenBits** | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 23 — Syntax of HoaRenderingMatrix()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HoaRenderingMatrix(NumOfHoaCoeffs, outputConfig, outputCount) | | |
| { | | |
| lfeExist = 0; | | |
| hasLfeRendering = 0; | | |
| for (i=0; i< inputCount; ++i) | | |
| isHoaCoefSparse[i] = 0; | | |
| **precisionLevel** | **2** | **uimsbf** |
| if (**gainLimitPerHoaOrder**) { | **1** | **uimsbf** |
| for (i = 0; i<(maxHoaOrder+1); ++i ) { | | |
| maxGain[i] = - escapedValue(3, 5, 6); | | |
| minGain[i] = -(escapedValue(4, 5, 6) + 1 – maxGain[i]); | | |
| } | | |
| } else { | | |
| maxGain[0] = - escapedValue (3, 5, 6); | | |
| minGain[0] = -( escapedValue (4, 5, 6) + 1- maxGain[0]); | | |
| for (i = 1; i<(maxHoaOrder+1); ++i ) { | | |
| maxGain[i] = maxGain[0]; | | |
| minGain[i] = minGain[0]; | | |
| } | | |
| } | | |
| if (**isFullMatrix**) { | **1** | **uimsbf** |
| **firstSparseOrder** | **1** | **uimsbf** |
| for (i = (firstSparseOrder*firstSparseOrder); i<inputCount; ++i) | | |
| isHoaCoefSparse[i] = 1; | | |
| } | | |
| for (i=0; i< outputCount; ++i){ | | |
| if (outputConfig[i].isLFE) | | |
| lfeExist = 1; | | |
| } | | |
| if (lfeExist) | | |
| **hasLfeRendering;** | **1** | **uimsbf** |
| numPairs = findSymmetricSpeakers(outputCount, outputConfig, hasLfeRendering); | | |
| for (i=0; i<numPairs; ++i) { | | |
| valueSymmetricPairs[i] = 0; | | |
| signSymmetricPairs[i] = 0; | | |
| } | | |
| **zerothOrderAlwaysPositive;** | **1** | **uimsbf** |
| if (**isAllValueSymmetric**) { | **1** | **uimsbf** |
| for (i=0; i<numPairs; ++i) { valueSymmetricPairs[i] = 1; } | | |
| } else { | | |
| if (**isAnyValueSymmetric**) { | **1** | **uimsbf** |
| for (i=0; i<numPairs; ++i) | | |
| valueSymmetricPairs[i] = **boolVal;** | **1** | **uimsbf** |
| if (**isAnySignSymmetric**) { | **1** | **uimsbf** |
| for (i=0; i<numPairs; ++i) { | | |
| if (0==valueSymmetricPairs[i]) | | |
| signSymmetricPairs[i] = **boolVal;** | **1** | **uimsbf** |
| } | | |
| } | | |
| } else { | | |
| if (**isAllSignSymmetric**) { | **1** | **uimsbf** |
| for (i=0; i<numPairs; ++i) | | |
| signSymmetricPairs[i] = 1; | | |
| } else { /* isAnyValueSymmetric==0 */ | | |
| if { **isAnySignSymmetric**) { | **1** | **uimsbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
            for (i=0; i<numPairs; ++i)
                signSymmetricPairs[i] = boolVal                    1          uimsbf
        }
    }
    hasVerticalCoef;                                               1          uimsbf
    DecodeHoaMatrixData()
}
```

**Table 24 — Syntax of DecodeHoaMatrixData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| DecodeHoaMatrixData()<br>{<br>j = 0;<br>for (i=0; i<outputCount; ++i) {<br>isValueSymmetric[i] = 0;<br>isSignSymmetric[i] = 0;<br>if ( (outputConfig[i].pairType == SP_PAIR_SYMMETRIC) &&<br>(outputConfig[i].symmetricPair != NULL)) {<br>if (0==(outputConfig[i].isLFE && (0==hasLfeRendering))) {<br>isValueSymmetric[i] = valueSymmetricPairs[j];<br>isSignSymmetric[i] = signSymmetricPairs[j++];<br>}<br>}<br>}<br>for (i = 0; i < inputCount; ++i) {<br>currentHoaOrder = ceil(sqrt(i+1)-1);<br>for (j = outputCount-1; j >= 0; --j) {<br>signMatrix[i * outputCount + j] = 1;<br>hoaMatrix [i * outputCount + j] = 0.0;<br>if ((vertBitmask[i] && hasVerticalCoef) \|\| !vertBitmask[i]) {<br>hasValue = 1;<br>if (0 == isValueSymmetric[j]) {<br>if ( (hasLfeRendering && outputConfig[j].isLFE) \|\|<br>(!outputConfig[j].isLFE)) {<br>if (isHoaCoefSparse[i]) {<br>**hasValue**; | 1 | **uimsbf** |
| }<br>if (hasValue) {<br>hoaMatrix [i * outputCount + j] =<br>DecodeHoaGainValue(currentHoaOrder);<br>if (0==isSignSymmetric[j]) {<br>if (hoaMatrix [i * outputCount + j] != 0.0) {<br>if (currentHoaOrder \|\| !zerothOrderAlwaysPositive) {<br>signMatrix[i * outputCount + j] =<br>**signVal***2-1; | 1 | **uimsbf** |
| }<br>}<br>} else { // isSignSymmetric[i] == 1<br>pairIdx = outputConfig[j].symmetricPair->originalPosition;<br>signMatrix[i * outputCount + j] =<br>symSigns[i] * signMatrix[i * outputCount + pairIdx];<br>}<br>} /* has value */<br>}<br>} | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
                else { /* isValueSymmetric */
                    pairIdx = outputConfig[j].symmetricPair->originalPosition;
                    hoaMatrix [i*outputCount+j] =
                                hoaMatrix [i*outputCount+pairIdx];
                    signMatrix[i *outputCount+j] =
                                symSigns[i] * signMatrix[i*outputCount+pairIdx];
                }
            }
        }
    }
    for (i = 0; i < inputCount; ++i) {
        for (j = 0; j < outputCount; ++j) {
            hoaMatrix[i *outputCount+j] *= signMatrix[i*outputCount+j];
            hoaMatrix[i *outputCount+j] /= sqrt(2 × ceil(sqrt(i+1)-1) + 1);
        }
    }
}
```

**Table 25 — Syntax of DecodeHoaGainValue()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| DecodeHoaGainValue(order)<br>{<br>nAlphabet = ((maxGain[order] – minGain[order]) * 2 ^ precisionLevel) + 2;<br>gainValueIndex = ReadRange(nAlphabet);<br>gainValue = maxGain[order] - gainValueIndex / 2 ^ precisonLevel;<br>if (gainValue < minGain[order]) {<br>gainValue = 0.0;<br>} else {<br>gainValue = 10.0 ^ (gainValue / 20.0);<br>}<br>return gainValue;<br>} | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

### 5.2.3 MPEG-H 3D Audio Core bitstream payloads

#### 5.2.3.1 Payloads for MPEG-H 3D Audio Core

The bit stream syntax is based on ISO/IEC 23003-3:2012, subclause 5.3.1. Modifications and amendments are listed below.

**Table 26 — Syntax of mpegh3daFrame(), top level payload for MPEG-H 3D Audio Core**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daFrame()<br>{ | | |
| **usacIndependencyFlag;** | **1** | **uimsbf** |
| for (elemIdx=0; elemIdx<numElements; ++elemIdx) {<br>if ( (usacElementType[elemIdx] != ID_USAC_EXT) &&<br>(elementLengthPresent == 1) ) { | | |
| **elementLength** | **16** | **uimsbf** |
| }<br>switch (usacElementType[elemIdx]) {<br>case ID_USAC_SCE: | | |


© ISO/IEC 2015 – All rights reserved

```
            mpegh3daSingleChannelElement(usacIndependencyFlag);    elementLength, NOTE 1
            break;
        case ID_USAC_CPE:
            mpegh3daChannelPairElement(usacIndependencyFlag);      elementLength, NOTE 1
            break;
        case ID_USAC_LFE:
            mpegh3daLfeElement(usacIndependencyFlag);              elementLength, NOTE 1
            break;
        case ID_USAC_EXT:
            mpegh3daExtElement(usacIndependencyFlag);
            break;
    }
}
```

NOTE 1: If present, elementLength represents the length of the corresponding element it refers to in number of bits.

**Table 27 — Syntax of mpegh3daSingleChannelElement()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daSingleChannelElement(indepFlag)<br>{<br>  mpegh3daCoreCoderData(1, indepFlag);<br><br>  if (sbrRatioIndex > 0) {<br>  UsacSbrData(1, indepFlag);<br>  }<br>} | | |

**Table 28 — Syntax of mpegh3daChannelPairElement()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daChannelPairElement(indepFlag)<br>{<br>  if (stereoConfigIndex == 1) {<br>  nrCoreCoderChannels = 1;<br>  } else {<br>  nrCoreCoderChannels = 2;<br>  }<br><br>  mpegh3daCoreCoderData(nrCoreCoderChannels, indepFlag);<br><br>  if (sbrRatioIndex > 0) {<br>  if (stereoConfigIndex == 0 \|\| stereoConfigIndex == 3) {<br>  nrSbrChannels = 2;<br>  } else {<br>  nrSbrChannels = 1;<br>  }<br>  UsacSbrData(nrSbrChannels, indepFlag);<br>  }<br><br>  if (stereoConfigIndex > 0) {<br>  Mps212Data(indepFlag);<br>  }<br>} | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 29 — Syntax of mpegh3daLfeElement()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daLfeElement(indepFlag)<br>{<br>  fd_channel_stream(0,0,0,0, indepFlag);<br>} | | |

**Table 30 — Syntax of UsacExtElement()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daExtElement(indepFlag)<br>{ | | |
| **usacExtElementPresent** | **1** | **uimsbf** |
| if (usacExtElementPresent==1) { | | |
| **usacExtElementUseDefaultLength;** | **1** | **uimsbf** |
| if (usacExtElementUseDefaultLength) { | | |
| usacExtElementPayloadLength = usacExtElementDefaultLength; | | |
| } else { | | |
| usacExtElementPayloadLength = escapedValue(8,16,0); | | |
| } | | |
| if (usacExtElementPayloadLength>0) { | | |
| if (usacExtElementPayloadFrag) { | | |
| **usacExtElementStart;** | **1** | **uimsbf** |
| **usacExtElementStop;** | **1** | **uimsbf** |
| } else { | | |
| usacExtElementStart = 1; | | |
| usacExtElementStop = 1; | | |
| } | | |
| for (i=0; i<usacExtElementPayloadLength; i++) { | | |
| usacExtElementSegmentData[i]; | **8** | **uimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |

**Table 31 — Syntax of ics_info()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| ics_info()<br>{ | | |
| **window_sequence;** | **2** | **uimsbf** |
| **window_shape;** | **1** | **uimsbf** |
| if (window_sequence == EIGHT_SHORT_SEQUENCE) { | | |
| **max_sfb;** | **4** | **uimsbf** |
| **scale_factor_grouping;** | **7** | **uimsbf** |
| } | | |
| else { | | |
| **max_sfb;** | **6** | **uimsbf** |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 5.2.3.2 Subsidiary payloads

**Table 32 — Syntax of mpegh3daCoreCoderData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daCoreCoderData(nrChannels, indepFlag) | | |
| { | | |
| for (ch=0; ch < nrChannels; ch++) { | | |
| **core_mode**[ch]**;** | **1** | **uimsbf** |
| } | | |
| if (nrChannels == 2) { | | |
| StereoCoreToolInfo(core_mode); | | |
| } | | |
| for (ch=0; ch<nrChannels; ch++) { | | |
| if (core_mode[ch] == 1) { | | |
| lpd_channel_stream(indepFlag); | | |
| } | | |
| else { | | |
| if ( (nrChannels == 1) \|\| (core_mode[0] != core_mode[1]) ) { | | |
| **tns_data_present**[ch]; | **1** | **uimsbf** |
| } | | |
| fd_channel_stream(common_window, common_tw, | | |
| tns_data_present[ch], noiseFilling, indepFlag); | | |
| } | | |
| } | | |
| } | | |

**Table 33 — Syntax of StereoCoreToolInfo()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| StereoCoreToolInfo(core_mode) | | |
| { | | |
| if (core_mode[0] == 0 && core_mode[1] == 0) { | | |
| **tns_active;** | **1** | **uimsbf** |
| if (**common_window**) { | **1** | **uimsbf** |
| ics_info(); | | |
| if (**common_max_sfb**) { | **1** | **uimsbf** |
| max_sfb1 = max_sfb; | | |
| } else { | | |
| if (window_sequence == EIGHT_SHORT_SEQUENCE) { | | |
| **max_sfb1;** | **4** | **uimsbf** |
| } else { | | |
| **max_sfb1;** | **6** | **uimsbf** |
| } | | |
| } | | |
| max_sfb_ste = max(max_sfb, max_sfb1); | | |
| if (enhancedNoiseFilling && !igfIndependentTiling) { | | |
| max_sfb_ste = min(max_sfb_ste, igf_sfb_start); | | |
| } | | |
| **ms_mask_present;** | **2** | **uimsbf** |
| if (ms_mask_present == 1) { | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
            for (g = 0; g < num_window_groups; g++) {
                for (sfb = 0; sfb < max_sfb_ste; sfb++) {
                    ms_used[g][sfb];                                    1           uimsbf
                }
            }
        }
        if (ms_mask_present == 3) {
            cplx_pred_data();
        } else {
            for (g = 0; g < num_window_groups; g++) {
                for (sfb = 0; sfb < max_sfb_ste; sfb++) {
                    alpha_q_re[g][sfb] = 0;                             [1]         [2]
                    alpha_q_im[g][sfb] = 0;
                }
            }
        }

        if (enhancedNoiseFilling && !igfIndependentTiling) {
            igf_ms_mask_present;                                        2           uimsbf
            if (igf_ms_mask_present == 1) {
                for (g = 0; g < num_window_groups; g++) {
                    for (sfb = igf_sfb_start; sfb < igf_sfb_stop; sfb += (2 – igfUseHighRes)) {
                        ms_used[g][sfb];                                1           uimsbf
                        if (!igfUseHighRes && (sfb + 1) < igf_sfb_stop) {
                            ms_used[g][sfb+1] = ms_used[g][sfb];
                        }
                    }
                }
            }
            if (igf_ms_mask_present == 3) {
                igf_stereo_pred_data();
            } else {
                for (g = 0; g < num_window_groups; g++) {
                    for (sfb = max_sfb_ste; sfb < igf_sfb_stop; sfb++) {
                        alpha_q_re[g][sfb] = 0;                         [3]         [4]
                        alpha_q_im[g][sfb] = 0;
                    }
                }
            }
        }
    }
    if (tw_mdct) {
        if (common_tw) {                                                1           uimsbf
            tw_data();
        }
    }
    if (tns_active) {
        if (common_window) {
            common_tns;                                                 1           uimsbf
        } else {
            common_tns = 0;
        }
        if (!enhancedNoiseFilling || igfAfterTnsSynth) {
            tns_on_lr;                                                  1           uimsbf
        } else {
            tns_on_lr = 1;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
            }
            if (common_tns) {
                tns_data();
                tns_data_present[0] = 0;
                tns_data_present[1] = 0;
            } else {
                if (tns_present_both) {                          1          uimsbf
                    tns_data_present[0] = 1;
                    tns_data_present[1] = 1;
                } else {
                    tns_data_present[1];                         1          uimsbf
                    tns_data_present[0] = 1 – tns_data_present[1];
                }
            }
        } else {
            common_tns = 0;
            tns_data_present[0] = 0;
            tns_data_present[1] = 0;
        }
    } else {
        common_window = 0;
        common_tw = 0;
    }
}
```

**Table 34 — Syntax of fd_channel_stream()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| fd_channel_stream(common_window, common_tw, tns_data_present, noiseFilling, indepFlag) | | |
| { | | |
| **global_gain;** | **8** | **uimsbf** |
| if (noiseFilling) { | | |
| **noise_level;** | **3** | **uimsbf** |
| **noise_offset;** | **5** | **uimsbf** |
| } else { | | |
| noise_level = 0; | | |
| } | | |
| if (!common_window) { | | |
| ics_info(); | | |
| } | | |
| if (tw_mdct && !common_tw) { | | |
| tw_data(); | | |
| } | | |
| scale_factor_data(); | | |
| | | |
| if (enhancedNoiseFilling) { | | |
| **igf_AllZero** | **1** | **uimsbf** |
| **igf_level(**igf_AllZero**,** indepFlag**);** | **0…** | NOTE |
| if (!igf_AllZero) { | | |
| igf_data(indepFlag); | | |
| } else { | | |
| igfPrevTileIdx = {3}; | | |
| igf_PrevWhiteningLevel = {0}; | | |
| } | | |
| } | | |
| if (tns_data_present) { | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
        tns_data();
    }
    ac_spectral_data(indepFlag);

    if (fac_data_present) {                                          1          uimsbf
        fac_length = (window_sequence == EIGHT_SHORT_SEQUENCE) ? ccfl/16 : ccfl/8;
        fac_data(1, fac_length);
    }
}
```

NOTE: For details on **igf_level()** see 5.5.5.

**Table 35 — Syntax of igf_stereo_pred_data()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| igf_stereo_pred_data(igf_sfb_start, igf_sfb_stop, indepFlag) | | |
| { | | |
| **igf_stereo_pred_all;** | **1** | **uimsbf** |
| if (igf_stereo_pred_all == 0) { | | |
| for (g = 0; g < num_window_groups; g++) { | | |
| for (sfb = igf_sfb_start; sfb < igf_sfb_stop; sfb += SFB_PER_PRED_BAND) { | | |
| **cplx_pred_used**[g][sfb]; | **1** | **uimsbf** |
| if ((sfb + 1) < igf_sfb_stop) { | | |
| cplx_pred_used[g][sfb+1] = cplx_pred_used[g][sfb]; | | |
| } | | |
| } | | |
| } | | |
| } else { | | |
| for (g = 0; g < num_window_groups; g++) { | | |
| for (sfb = igf_sfb_start; sfb < igf_sfb_stop; sfb++) { | | |
| cplx_pred_used[g][sfb] = 1; | | |
| } | | |
| } | | |
| } | | |
| **igf_pred_dir;** | **1** | **uimsbf** |
| if (indepFlag) { | | |
| igf_delta_code_time = 0; | | |
| } else { | | |
| **igf_delta_code_time;** | **1** | **uimsbf** |
| } | | |
| for (g = 0; g < num_window_groups; g++) { | | |
| for (sfb = igf_sfb_start; sfb < igf_sfb_stop; sfb += SFB_PER_PRED_BAND) { | | |
| if (cplx_pred_used[g][sfb]) { | | |
| **hcod_sf**[dpcm_alpha_q_re[g][sfb]]; | **1..19** | **vlclbf** |
| } else { | | |
| alpha_q_re[g][sfb] = 0; | **[5]** | [6] |
| } | | |
| alpha_q_im[g][sfb] = 0; | | |
| } | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 36 — Syntax of igf_data()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| igf_data(indepFlag) | | |
| { | | |
| if (!indepFlag) { | | |
| if (**igf_UsePrevTileIdx**) { | **1** | **uimsbf** |
| for (i = 0; i < igfNTiles; i++) { | | |
| igfCurrTileIdx[ i ] = igfPrevTileIdx[ i ]; | | |
| } | | |
| } | | |
| } else { | | |
| for (i = 0; i < igfNTiles; i++) { | | |
| **igfCurrTileIdx**[ i ]; | **2** | **uimsbf** |
| } | | |
| } | | |
| for (i = igfNTiles; i < 4; i++) { | | |
| igfCurrTileIdx[ i ] = 3; | | |
| igfPrevTileIdx[ i ] = 3; | | |
| } | | |
| for (i = 0; i < igfNTiles; i++) { | | |
| igfPrevTileIdx[ i ] = igfCurrTileIdx[ i ]; | | |
| } | | |
| if (igfUseWhitening && window_sequence != EIGHT_SHORT_SEQENCE) { | | |
| if (indepFlag) { | | |
| igf_UsePrevWhiteningLevel = 0; | | |
| } else { | | |
| **igf_UsePrevWhiteningLevel;** | **1** | **uimsbf** |
| } | | |
| if (igf_UsePrevWhiteningLevel) { | | |
| for (i = 0; i < igfNTiles; i++) { | | |
| igf_WhiteningLevel[ i ] = igf_PrevWhiteningLevel[ i ]; | | |
| } | | |
| } else { | | |
| if (**igf_WhiteningLevel**[0]) { | **1** | **uimsbf** |
| igf_WhiteningLevel[0] = 1 + **tmp**; | **1** | **uimsbf** |
| } | | |
| if (**remainingTilesDifferent**) { | **1** | **uimsbf** |
| for (i = 1; i < igfNTiles; i++) { | | |
| if (**igf_WhiteningLevel**[ i ]) { | **1** | **uimsbf** |
| igf_WhiteningLevel[ i ] = 1 + **tmp**; | **1** | **uimsbf** |
| } | | |
| } | | |
| } else { | | |
| for (i = 1; i < igfNTiles; i++) { | | |
| igf_WhiteningLevel[ i ] = igf_WhiteningLevel[0]; | | |
| } | | |
| } | | |
| } | | |
| } | | |
| for (i = igfNTiles; i < 4; i++) { | | |
| igf_WhiteningLevel[ i ] = 0; | | |
| igf_PrevWhiteningLevel[ i ] = 0; | | |
| } | | |
| if (igfUseEnf) { | | |
| If (window_sequence != EIGHT_SHORT_SEQENCE) { | | |
| **isIGFEnfFlat;** | **1** | **uimsbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
        }
    }
}
```

**Table 37 — References to USAC syntactic elements**

| Syntax of | Defined in |
|---|---|
| ics_info() | ISO/IEC 23003-3:2012, 5.3, Table 22 |
| tw_data() | ISO/IEC 23003-3:2012, 5.3, Table 27 |
| scale_factor_data() | ISO/IEC 23003-3:2012, 5.3, Table 28 |
| tns_data() | ISO/IEC 23003-3:2012, 5.3, Table 29 |
| ac_spectral_data() | ISO/IEC 23003-3:2012, 5.3, Table 30 |
| lpd_channel_stream() | ISO/IEC 23003-3:2012, 5.3, Table 31 |
| fac_data() | ISO/IEC 23003-3:2012, 5.3, Table 39 |
| UsacSbrData() | ISO/IEC 23003-3:2012, 5.3, Table 40 |
| Mps212Data() | ISO/IEC 23003-3:2012, 5.3, Table 52 |

#### 5.2.3.3 Extension Payloads

**Table 38 — Syntax of AudioPreRoll()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| AudioPreRoll() | | |
| { | | |
| **configLen** = escapedValue(4,4,8); | **4..16** | |
| Config() | **8*configLen** | |
| | | |
| **numPreRollFrames** = escapedValue(2,4,0); | **2..6** | |
| | | |
| for (frameIdx=0; frameIdx < numPreRollFrames; ++frameIdx) { | | |
| **auLen** = escapedValued(16,16,0) | **16..32** | **uimsbf** |
| AccessUnit() | **8*auLen** | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

## 5.3 Data Structure

### 5.3.1 General

The data structure is based on ISO/IEC 23003-3:2012, Clause 6.

Modifications and amendments are listed below.

Signal groups are represented by audio data elements which are configured in the mpegh3daDecoderConfig(). The bitstream element Signals3d() specifies the assignment of audio data elements to signal groups. In case a signal group requires extension payloads, the corresponding mpegh3daExtElementConfig()s shall directly follow the configuration elements for the audio data elements which belong to the associated signal group..

### 5.3.2 General Configuration Data Elements

**mpegh3daProfileLevelIndication**

Indication of MPEG-H 3D Audio profile and level according to Table 39.


© ISO/IEC 2015 – All rights reserved

**Table 39 — Value of mpegh3daProfileLevelIndication**

| value | Indication of Profile | Indication of Level |
|---|---|---|
| 0x00 | reserved for ISO use | - |
| 0x01-0xff | reserved for future profile definition | |

**coreQmfDelayCompensation** This flag forces the MPEG-H 3D Audio Core coder to apply a delay compensation for the case that eSBR is not applied. If its value is 0, no delay compensation shall be applied. If its value is 1, a delay of 577 samples shall be applied to all audio elements which do not use eSBR, i.e. which do not employ a QMF analysis during decoding.
Further, if processing blocks immediately following the MPEG-H 3D Audio Core require QMF domain input, the delay is bypassed and the output is fed to a QMF analysis instead, such that the MPEG-H 3D Audio Core decoder and the subsequent processing blocks can be efficiently combined in the QMF domain.
Audio elements which *do* employ eSBR are not affected.

**receiverDelayCompensation** This flag forces the format converter to apply a delay compensation for the case that it is not active. When its value is 0, no delay compensation shall be applied in the bypass path. When its value is 1, a delay of 4033 samples shall be applied, when the format converter is fed with time-domain input, a delay of 2432 samples (corresponding to 38 QMF time slots) shall be applied, when the format converter is fed with QMF domain input.

referenceLayout This structure describes the loudspeaker layout which the content of the present audio stream was originally designed or produced for. In that sense it represents the optimal loudspeaker layout from the content creator's perspective. This layout also acts as default rendering layout in the case where the targetLayout is not known to the decoder. The referenceLayout must contain a specific and real layout. A ChannelConfiguration value of 0 (“any setup”) of ISO/IEC 23001-8 is not allowed.

audioChannelLayout This structure describes the loudspeaker layout for a group with SignalGroupType == SignalGroupTypeChannels. If the audioChannelLayout is different from the referenceLayout (differsFromReferenceLayout==1), then the audioChannelLayout shall be smaller than the referenceLayout.

targetLayout This structure describes the target loudspeaker layout, i.e. the actual loudspeaker constellation which the audio content shall be rendered to. This information will typically come from outside of the decoder. If the targetLayout is unknown, it shall be assumed that targetLayout = referenceLayout.

**bsNumSignalGroups** This field defines the number of signal groups that are present in the bitstream.

**signal_groupID** This variable implicitly assigns an ID to the signal groups listed in the Signals3d() syntax element. The ID is used to assign signal groups to transmitted downmix matrices in the DownmixMatrixSet() syntax element. Only signal groups of type SignalGroupTypeChannels may be assigned to downmix matrices.

**signalGroupType** This field defines the type of a signal group.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 40 — Value of signalGroupType**

| signalGroupType | value | meaning |
|---|---|---|
| SignalGroupTypeChannels | 0 | Signal group contains channel signals, i.e. signals that should be played back from one static position, e.g. by a specific loudspeaker |
| SignalGroupTypeObject | 1 | Signal group contains object signals, i.e. signals that should be rendered to the reproduction layout |
| SignalGroupTypeSAOC | 2 | Signal group contains SAOC signals |
| SignalGroupTypeHOA | 3 | Signal group contains HOA signals |
| reserved | 4 - 7 | n.a. |

**differsFromReferenceLayout** This flag indicates whether the layout for which the signal group (of type SignalGroupTypeChannels) is intended differs from the reference layout.

numAudioChannels Explicit number of audio channels that are conveyed in the present stream by means of SCEs, CPEs, QCEs, and LFEs. The value of this bit stream field shall correspond to the number of channels as signaled with the help of audioChannelLayout. In the element loop of mpegh3daDecoderConfig() these elements shall be located before all object related elements.

numAudioObjects Explicit number of audio objects channels that are conveyed in the present stream by means of SCEs, CPEs, and QCEs and that are further processed in the object renderer. In the element loop of mpegh3daDecoderConfig() these elements shall be located after all audio channel related elements.

numSAOCTransportChannels Explicit number of SAOC audio transport channels that are conveyed in the present stream by means of SCEs and that are further processed in the SAOC 3D Decoder. In the element loop of mpegh3daDecoderConfig() these elements shall be located after all audio channel and audio object related elements.

numHOATransportChannels Explicit number of HOA transport channels that are conveyed in the present stream by means of SCEs, CPEs, QCEs and that are further processed in the HOA Decoder. In the element loop of mpegh3daDecoderConfig() these elements shall be located after all audio channel, audio object and SAOC related elements.

### 5.3.3 Loudspeaker Configuration Data Elements

**speakerLayoutType** This field indicates by which means the loudspeaker layout is conveyed in the bitstream element according to Table 41

**Table 41 — Meaning of speakerLayoutType**

| Value | Meaning |
|---|---|
| 0 | Loudspeaker layout ist signaled by means of ChannelConfiguration index as defined in ISO/IEC 23001-8. |
| 1 | Loudspeaker layout ist signaled by means of a list of LoudspeakerGeometry indices as defined in ISO/IEC 23001-8 |
| 2 | Loudspeaker layout is signaled by means of a list of explicit geometric position information. |
| 3 | *Reserved* |

**CICPspeakerLayoutIdx** ChannelConfiguration value as defined in ISO/MPEG 23001-8

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**CICPspeakerIdx** LoudspeakerGeometry value as defined in ISO/MPEG 23001-8

**angularPrecision** This flag signals the angular precision of loudspeaker geometry information according to Table 42

**Table 42 — Meaning of angularPrecision**

| Value of angularPrecision | angular precision in degrees (°) | Number of bits used for coding of bit stream field: | |
|---|---|---|---|
| | | ElevationAngleIdx | AzimuthAngleIdx |
| 0 | 5 | 5 | 6 |
| 1 | 1 | 7 | 8 |

**alsoAddSymmetricPair** This flag signals if a symmetric pair on the horizontal plane is directly following

**isCICPspeakerIdx** This flag signals if the loudspeaker position is signaled by means of a LoudspeakerGeometry according to ISO/IEC 23001-8.

**ElevationClass** Indicates loudspeaker elevation by means of a simple middle/upper/lower layer indication according to Table 43.

**Table 43 — Meaning of ElevationClass field**

| Value | Meaning | ElevationAngle |
|---|---|---|
| 0 | Loudspeaker is located in the Middle Layer. | 0° |
| 1 | Loudspeaker is located in the Upper Layer. | 35° |
| 2 | Loudspeaker is located in the Lower Layer | -15° |
| 3 | Loudspeaker position signaled explicitly | N.A. |

**ElevationAngleIdx** Index for calculating the elevation (i.e. vertical) angle for a given loudspeaker according to Table 44.

**Table 44 — Calculation of ElevationAngle from ElevationAngleIdx**

| ElevationAngleIdx | angularPrecision == 0 | angularPrecision == 1 |
|---|---|---|
| 0-18 | ElevationAngle = ElevationAngleIdx * 5° | ElevationAngle = ElevationAngleIdx * 1° |
| 19-31 | reserved | |
| 32-90 | N.A. | |
| 91-127 | N.A. | reserved |

**ElevationDirection** This flag signals the direction of increasing elevation according to Table 45.

**Table 45 — Meaning of ElevationDirection**

| Value of ElevationDirection | Direction |
|---|---|
| 0 | Upwards |
| 1 | Downwards |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**AzimuthAngleIdx** Index for calculating the azimuth (i.e. horizontal) angle for a given loudspeaker according to Table 46.

**Table 46 — Calculation AzimuthAngle from AzimuthAngleIdx**

| AzimuthAngleIdx | angularPrecision == 0 | angularPrecision == 1 |
|---|---|---|
| 0-36 | AzimuthAngle = AzimuthAngleIdx * 5° | AzimuthAngle = AzimuthAngleIdx * 1° |
| 37-180 | N.A. | |
| 181-255 | N.A. | reserved |

**AzimuthDirection** This flag signals the direction of increasing azimuth angles according to Table 47

**Table 47 — Meaning of AzimuthDirection**

| Value of AzimuthDirection | Direction |
|---|---|
| 0 | Counter-clockwise |
| 1 | clockwise |

**isLFE** This flag signals if the given loudspeaker is an LFE loudspeaker

### 5.3.4 Core Decoder Configuration Data Elements

**qceIndex** This index describes whether two subsequent elements of type mpegh3daChannelPairElement() are treated as a Quadruple Channel Element (QCE). The different QCE modes are given in Table 48. The qceIndex shall be the same for the two subsequent elements forming one QCE.

**Table 48 — Value of qceIndex**

| qceIndex | meaning |
|---|---|
| 0 | Stereo CPE |
| 1 | QCE without residual |
| 2 | QCE with residual |
| 3 | reserved |

**shiftIndex0** This flag signals if the first channel in the element is the next non-assigned channel or if the element is shifted in the channel map relative to the next non-assigned channel.

**shiftIndex1** This flag signals if the second channel in element is the next non-assigned channel or if the second channel in the element is shifted in the channel map relative to the next non-assigned channel.

**shiftChannel0** Offset by which the first channel in the element is shifted relative to the next non-assigned channel.

**shiftChannel1** Offset by which the second channel in the element is shifted relative to the next non-assigned channel.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**enhancedNoiseFilling** This flag signals the usage of the Enhanced Noise Filling tool.

**igfUseEnf** This flag signals the usage of IGF envelope noise flattening.

**igfUseWhitening** This flag signals the usage of IGF spectral whitening.

**igfAfterTnsSynth** This flag signals that IGF should be applied after TNS synthesis filtering.

**igfIndependentTiling** This flag signals that IGF is applied in discrete channel mode.

**igfStartIndex** This flag signals the IGF start index, which is mapped to a scalefactor band index.

**igfStopIndex** This flag signals the IGF stop index, which is mapped to a scalefactor band index.

**igfUseHighRes** This flag signals that for every scalefactor band in IGF range a IGF level value is transmitted. If the flag is zero, low resolution is used which implies, that for two scalefactor bands only one IGF level value is transmitted.

**igfCurrTileIdx[ ]** Vector of length 4 containing the tile index.

**isIgfEnfFlat** Flag indicating the application of IGF-ENF filtering

**igf_data()** Syntax element which reads IGF tile and whitening side information for each channel ch.

**igf_level()** Syntax element which reads the IGF level information for each channel ch.

**igf_AllZero** This flag signals that all levels in IGF range are zero.

**igf_UsePrevTileIdx** This flag signals that previous values of tile indices should be used.

**igf_UsePrevWhiteningLevel** This flag signals that previous values of whitening levels should be used.

**igf_WhiteningLevel[ ]** This values describe which whitening should be used, see Table 49.

**igf_ms_mask_present** this two bit field indicates that the MS mask is:
00 All zeros.
01 A mask of max_sfb bands of **ms_used** follows this field.
10 All ones.
11 M/S coding is disabled, real stereo prediction is enabled.

**igf_pred_dir** Indicates the direction of prediction (same as cplx_pred_dir).

**Table 49 — Value of igf_WhiteningLevel**

| igf_WhiteningLevel | Meaning |
|---|---|
| 0 | use medium whitening |
| 1 | do not use whitening |
| 2 | use pseudo-random noise |

**usacExtElementType** This element allows to signal bit stream extensions types. The meaning of usacExtElementType is defined in Table 50.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 50 — Value of usacExtElementType**

| usacExtElementType | Value |
|---|---|
| ID_EXT_ELE_FILL | 0 |
| ID_EXT_ELE_MPEGS | 1 |
| ID_EXT_ELE_SAOC | 2 |
| ID_EXT_ELE_AUDIOPREROLL | 3 |
| ID_EXT_ELE_UNI_DRC | 4 |
| ID_EXT_ELE_OBJ_METADATA | 5 |
| ID_EXT_ELE_SAOC_3D | 6 |
| ID_EXT_ELE_HOA | 7 |
| ID_EXT_ELE_FMT_CNVRTR | 8 |
| /* reserved for ISO use */ | 9-127 |
| /* reserved for use outside of ISO scope */ | 128 and higher |
| NOTE: Application-specific usacExtElementType values are mandated to be in the space reserved for use outside of ISO scope. These are skipped by a decoder as a minimum of structure is required by the decoder to skip these extensions. | |

**usacExtElementSegmentData**

The concatenation of all usacExtElementSegmentData from mpegh3daExtElement() of consecutive frames, starting from the mpegh3daExtElement() with usacExtElementStart==1 up to and including the mpegh3daExtElement() with usacExtElementStop==1 forms one data block. In case a complete data block is contained in one mpegh3daExtElement(), usacExtElementStart and usacExtElementStop shall both be set to 1. The data blocks are interpreted as a byte aligned extension payload depending on usacExtElementType according to Table 51.

**Table 51 — Interpretation of data blocks for extension payload decoding**

| usacExtElementType | The concatenated usacExtElementSegmentData represents: |
|---|---|
| ID_EXT_ELE_FILL | Series of **fill_byte** |
| ID_EXT_ELE_MPEGS | SpatialFrame() as defined in ISO/IEC 23003-1 |
| ID_EXT_ELE_SAOC | SAOCFrame() as defined in ISO/IEC 23003-2 |
| ID_EXT_ELE_AUDIOPREROLL | AudioPreRoll() |
| ID_EXT_ELE_UNI_DRC | uniDrcGain() as defined in ISO/IEC 23003-4 |
| ID_EXT_ELE_OBJ_METADATA | object_metadata() |
| ID_EXT_ELE_SAOC_3D | Saoc3DFrame() |
| ID_EXT_ELE_HOA | HOAFrame() |
| ID_EXT_ELE_FMT_CNVRTR | FormatConverterFrame() |
| unknown | unknown data. The data block shall be discarded. |

**usacConfigExtType**

This element allows to signal configuration extension types. The meaning of usacConfigExtType is defined in Table 52.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 52 — Value of usacConfigExtType**

| usacConfigExtType | Value |
|---|---|
| ID_CONFIG_EXT_FILL | 0 |
| ID_CONFIG_EXT_DOWNMIX | 1 |
| ID_CONFIG_EXT_LOUDNESS_INFO | 2 |
| ID_CONFIG_EXT_AUDIOSCENE_INFO | 3 |
| ID_CONFIG_EXT_HOA_MATRIX | 4 |
| /* reserved for ISO use */ | 5-127 |
| /* reserved for use outside of ISO scope */ | 128 and higher |

### 5.3.5 Downmix Matrix Data Elements

**downmixIdCount** number of downmixId definitions present in the bit stream element

**downmixId** This field uniquely defines an ID for a default downmix matrix available on decoder side or a transmitted downmix matrix. downmixId has two reserved values, which are forbidden, namely 0x0 and 0x7F. All other values can be freely chosen. Further details on the usage of downmixId can be found in clause 5.7.

**downmixType** This index defines whether a downmixId is connected with a default downmix matrix available on decoder side or a transmitted downmix matrix.

**Table 53 — Value of downmixType**

| downmixType | meaning |
|---|---|
| 0 | Format conversion with default downmix matrix available on decoder side |
| 1 | Format conversion with transmitted downmix matrix |
| 2 | reserved |
| 3 | reserved |

**CICPspeakerLayoutIdx** This value describes the target loudspeaker layout for the given downmix matrix. The value corresponds to **ChannelConfiguration** as defined in ISO/IEC 23001-8.

**DmxMatrixLenBits** length of the following bit stream element in bits

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| paramConfig,<br>inputConfig,<br>outputConfig | Channel configuration vectors specifying the information about each speaker. The information is assumed to be known from the channel configurations of the input and output layouts. Each entry, paramConfig[i], is a structure with the members:<br>AzimuthAngle, the absolute value of the speaker azimuth angle;<br>AzimuthDirection, the direction of increasing azimuth, 0 (left) or 1 (right);<br>ElevationAngle, the absolute value of the speaker elevation angle;<br>ElevationDirection, the direction of increasing elevation, 0 (up) or 1 (down);<br>isLFE, indicates whether the speaker is a LFE speaker. |
| paramCount,<br>inputCount,<br>outputCount | Number of speakers in the corresponding channel configuration vectors. |
| compactParamConfig,<br>compactInputConfig,<br>compactOutputConfig | Compact channel configuration vectors specifying the information about each speaker group. Each entry, compactParamConfig[i], is a structure with the members:<br>pairType, type of the speaker group, which can be SYMMETRIC (a symmetric pair of two speakers), CENTER, or ASYMMETRIC;<br>isLFE, indicates whether the speaker group consists of LFE speakers;<br>originalPosition, position in the original channel configuration of the first (i.e. left) speaker, or the only speaker, in the group;<br>symmetricPair.originalPosition, position in the original channel configuration of the second (i.e. right) speaker in the group, for SYMMETRIC groups only. |
| compactParamCount,<br>compactInputCount,<br>compactOutputCount | Number of speaker groups in the corresponding compact channel configuration vectors. |
| equalizerPresent | Boolean indicating whether equalizer information that is to be applied to the input channels is present. |
| precisionLevel | Precision used for uniform quantization of the gains according to Table 54. |

**Table 54 — Uniform quantization step size of gain depending on precisionLevel**

| precisionLevel | smallest quantization step size [dB] |
|---|---|
| 0 | 1 |
| 1 | 0.5 |
| 2 | 0.25 |
| 3 | reserved |

| | |
|---|---|
| maxGain | Maximum actual gain in the matrix, expressed in dB: possible values from 0 to 22. |
| minGain | Minimum actual gain in the matrix, expressed in dB: possible values from -1 to -47. |
| isAllSeparable | Boolean indicating whether all the output speaker groups satisfy the |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| | separability property. |
| isSeparable[i] | Boolean indicating whether the output speaker group with index i satisfies the separability property. |
| isAllSymmetric | Boolean indicating whether all the output speaker groups satisfy the symmetry property. |
| isSymmetric[i] | Boolean indicating whether the output speaker group with index i satisfies the symmetry property. |
| mixLFEOnlyToLFE | Boolean indicating whether the LFE speakers are mixed only to LFE speakers and, at the same time, the non-LFE speakers are mixed only to non-LFE speakers. |
| rawCodingCompactMatrix | Boolean indicating whether compactDownmixMatrix is coded raw (using one bit per entry) or it is coded using run-length coding followed by limited Golomb-Rice. |
| compactDownmixMatrix[i][j] | An entry in compactDownmixMatrix corresponding to input speaker group i and output speaker group j, indicating whether any of the associated gains is nonzero:<br>0 = all gains are zero, 1 = at least one gain is nonzero. |
| useCompactTemplate | Boolean indicating whether to apply an element-wise XOR to compactDownmixMatrix with a predefined compact template matrix, to improve the efficiency of the run-length coding. |
| runLGRParam | Limited Golomb-Rice parameter p used to code the zero run-lengths in the linearized flatCompactMatrix. |
| flatCompactMatrix | Linearized version of compactDownmixMatrix with the predefined compact template matrix already applied;<br>When mixLFEOnlyToLFE is enabled, it does not include the entries known to be zero (due to mixing between non-LFE and LFE) or those used for LFE to LFE mixing. |
| compactTemplate | Predefined compact template matrix, having “typical” entries, which is XORed element-wise to compactDownmixMatrix, in order to improve coding efficiency by creating mostly zero value entries. |
| zeroRunLength | The length of a zero run always followeed by a one, in the flatCompactMatrix, which is coded with limited Golomb-Rice coding, using the parameter runLGRParam. |
| fullForAsymmetricInputs | Boolean indicating whether to ignore the symmetry property for every asymmetric input speaker group;<br>When enabled, every asymmetric input speaker group will have two gain values decoded for each symmetric output speaker group with index i, regardless of isSymmetric[i]. |
| gainTable | Dynamically generated gain table which contains the list of all possible gains between minGain and maxGain with precision precisionLevel. |
| rawCodingNonzeros | Boolean indicating whether the nonzero gain values are coded raw (uniform coding, using the ReadRange function) or their indexes in the gainTable list are coded using limited Golomb-Rice coding. |
| gainLGRParam | Limited Golomb-Rice parameter p used to code the nonzero gain |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

indexes, computed by searching each gain in the gainTable list.

### 5.3.6 HOA Rendering Matrix Data Elements

**numOfHoaRenderingMatrices** Number of *HoaRenderingMatrixId* definitions present in the bit stream element.

**HoaRenderingMatrixId** This field uniquely defines an ID for a default HOA rendering matrix available on decoder side or a transmitted HOA rendering matrix.

**CICPspeakerLayoutIdx** This value describes the output loudspeaker layout for the given HOA rendering matrix. The value corresponds to **ChannelConfiguration** as defined in ISO/IEC 23001-8.

**HoaRenderingMatrixLenBits** Length of the following bit stream element in bits

outputConfig Channel configuration vectors specifying the information about each loudspeaker. The information is assumed to be known from the channel configurations of the output layout. Each entry, outputConfig[i], is a structure with the members:

AzimuthAngle, the absolute value of the speaker azimuth angle;
AzimuthDirection, direction of increasing azimuth, 0 (left) or 1 (right);
ElevationAngle, the absolute value of the speaker elevation angle;
ElevationDirection, direction of increasing elevation, 0 (up) or 1 (down);
isLFE, indicates whether the speaker is a LFE speaker.

The helper function *findSymmetricSpeakers* further specifies the following specifications.

pairType, can be SYMMETRIC (a symmetric pair of two speakers), CENTER, or ASYMMETRIC;
symmetricPair->originalPosition, position in the original channel configuration of the second (i.e. right) speaker in the group, for SYMMETRIC groups only.

outputCount Number of loudspeakers the HOA rendering matrix is defined for.

numPairs Number of symmetric loudspeaker pairs identified in the output loudspeaker setup which can be considered for efficient symmetry coding.

**precisionLevel** Precision used for uniform quantization of the gains according to Table 55.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 55 — Uniform quantization step size for HOA rendering matrices as a function of the precisionLevel**

| precisionLevel | smallest quantization step size [dB] |
|---|---|
| 0 | 1.0 |
| 1 | 0.5 |
| 2 | 0.25 |
| 3 | 0.125 |

| | |
|---|---|
| **gainLimitPerHoaOrder** | This flag indicates if the maxGain and minGain are individually specified for each order or for the entire HOA matrix |
| **maxGain**[i] | Maximum actual gain in the matrix for coefficients for the HOA order *i*, expressed in dB. |
| **minGain**[i] | Minimum actual gain in the matrix for coefficients of the HOA order *i*, expressed in dB. |
| **isFullMatrix** | This flag indicates if the HOA matrix is sparse or full |
| **firstSparseOrder** | In case the HOA matrix was specified as sparse, this field defines the first HOA order which is sparsely coded |
| isHoaCoefSparse | Bitmask vector derived from firstSparseOrder |
| currentHoaOrder | Variable which indicates the current HOA order during the decoding process |
| **hasValue** | A flag used in case the matrix element is sparsely coded. |
| lfeExist | This flag indicates if one or more LFE channels exist in outputConfig |
| **hasLfeRendering** | This flag indicates if the rendering matrix contains non-zero elements for the LFE channels. The signals for the LFE channels are rendered without additional spectral filtering. |
| **zerothOrderAlwaysPositive** | This flag indicates if the $0^{th}$ HOA order has only positive values |
| **isAllValueSymmetric** | This fag indicates if all symmetric loudspeaker pairs have equal absolute values in the HOA rendering matrix |
| **isAnyValueSymmetric** | if *isAllValueSymmetric* is false, this flag indicates if some of the symmetric loudspeaker pairs have equal absolute values in the HOA rendering matrix |
| **valueSymmetricPairs** | Bitmask of length *numPairs* indicating the loudspeaker pairs with value-symmetry. |
| isValueSymmetric | Bitmask derived from *valueSymmetricPairs* |
| **isAllSignSymmetric** | If there are no value symmetries in the matrix, this flag indicates if all symmetric loudspeaker pairs have at least number sign symmetries. |
| **isAnySignSymmetric** | This flag indicates if there are at least some symmetric loudspeaker pairs with number sign symmetries. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| **signSymmetricPairs** | Bitmask of length *numPairs* indicating the loudspeaker pairs with sign-symmetry. |
| **signVal** | Number sign value |
| isSignSymmetric | Bitmask derived from *signSymmetricPairs* |
| **hasVerticalCoef** | This flag indicates if this is a horizontal-only HOA rendering matrix |
| **boolVal** | A variable used in the decoding loop |
| signMatrix | Matrix with the sign values of the HOA rendering matrix in linearized vector-form |
| hoaMatrix | Final HOA Rendering matrix values in linearized vector-form |

## 5.4 Configuration Element Descriptions

### 5.4.1 General

The configuration elements are based on ISO/IEC 23003-3:2012, subclause 6.1. Additional information is given in the following subclause.

The mapping of core audio elements to audio channels is applied according to the following rule:

— Obtain the number of audio channels (numAudioChannels) from Signals3d()

— Determine the already decoded successive elements until the currently processed element from mpegh3daDecoderConfig()

— Decode shiftChannel0 and shiftChannel1 from corresponding CPE elements. The default value for shiftChannel0 and shiftChannel1 shall be zero.

The decoding procedure is specified as follows:

a) Create an array with the size of the number of all loudspeaker signals and initialize it with -1.

b) Determine the smallest non-assigned channel ch that is missing in the array starting with ch=0.

c) Proceed to the next position pos in the array that equals -1. i.e. find the smallest pos such that array[pos] = -1.

d) if shiftIndexX is zero: write the channel index ch into the array at the position pos. i.e. array[pos] = ch.

e) if shiftIndexX is greater than zero: increment ch by (shiftIndexX+1) and write it into the array at the position pos. i.e. array[pos] = ch + shiftIndexX + 1

f) Repeat b) - e) for all channels in an element.

g) Repeat b) - f) for all successive elements.

Apply the same algorithm to audio objects for mapping the core audio elements to audio object channels.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

### 5.4.2 Downmix configuration

#### 5.4.2.1 General

Downmix matrix coefficients and/or active downmix setting parameters may be transmitted by the encoder to enable control over the format conversion process at the decoder. Transmission is facilitated by means of a mpegh3daConfigExtension of Type ID_CONFIG_EXT_DOWNMIX for downmixType == 1. The mpegh3daConfigExtension may contain downmix matrices as well as an active downmix setting parameter. If downmix matrices are transmitted, each downmix matrix signals its associated target speaker layout that determines the matrix dimensions and identifies which kind of downmix matrix operation the transmitted coefficients are suitable for.

The transmission of a unique *downmixId* allows referencing to a default downmix matrix available on decoder side, or to a transmitted downmix matrix from outside of the audio stream, e.g. from Dynamic Range Compression related information in higher Systems layers.

#### 5.4.2.2 Data Elements and Variables

**downmixConfigType** This parameter allows to signal whether an active downmix control parameter (value 0) or downmix matrices (value 1) or both (value 2) are transmitted. Value 3 is reserved.

**passiveDownmixFlag** Signals that a passive downmix shall be applied in the format converter downmix if value=1. If not transmitted, **passiveDownmixFlag** shall be set to 0, resulting in the application of the active downmix processing of the format converter downmix.

**immersiveDownmixFlag** Signals whether the immersive rendering format converter or the generic format converter shall be applied in the decoder to convert channel signals to the target loudspeaker configuration as defined in Table 56. If not transmitted, **immersiveDownmixFlag** shall be set to 0.

**Table 56 — Meaning of immersiveDownmixFlag**

| immersiveDownmixFlag | Meaning |
|---|---|
| 0 | Generic format converter shall be applied as defined in clause 10. |
| 1 | If the local loudspeaker setup, signaled by LoudspeakerRendering(), is signaled as<br>(speakerLayoutType==0,CICPspeakerLayoutIdx==5)<br>or as<br>(speakerLayoutType==0,CICPspeakerLayoutIdx==6),<br>independently of potentially signaled loudspeaker displacement angles, then immersive rendering format converter shall be applied as defined in clause 11.<br>In all other case the generic format converter shall be applied as defined in clause 10. |

#### 5.4.2.3 Golomb-Rice Coding

Golomb-Rice coding is used to code any non-negative integer $n \geq 0$, using a given non-negative integer parameter $p \geq 0$ as follows: first code the number $h = \lfloor n/2^p \rfloor$ using unary coding, as $h$ one bits followed by a terminating zero bit; then code the number $l = n - h \cdot 2^p$ uniformly using $p$ bits.

Limited Golomb-Rice coding is a trivial variant used when it is known in advance that $n < N$, for a given integer $N \geq 1$. It does not include the terminating zero bit when coding the maximum possible value of $h$,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

which is $h_{max} = \lfloor (N-1)/2^p \rfloor$. More exactly, to encode $h = h_{max}$ we write only $h$ one bits, but not the terminating zero bit, which is not needed because the decoder can implicitly detect this condition.

#### 5.4.2.4 Helper Functions

The function `ConvertToCompactConfig()` specified below is used to convert the given *paramConfig* configuration consisting of *paramCount* speakers into the compact *compactParamConfig* configuration consisting of *compactParamCount* speaker groups. The *compactParamConfig[i].pairType* field can be SYMMETRIC (S), when the group represents a pair of symmetric speakers, CENTER (C), when the group represents a center speaker, or ASYMMETRIC (A), when the group represents a speaker without a symmetric pair.

```
ConvertToCompactConfig(paramConfig, paramCount)
{
    for (i = 0; i < paramCount; ++i) {
        alreadyUsed[i] = 0;
    }

    idx = 0;
    for (i = 0; i < paramCount; ++i) {
        if (alreadyUsed[i]) continue;
        compactParamConfig[idx].isLFE = paramConfig[i].isLFE;

        if ((paramConfig[i].AzimuthAngle == 0) ||
                (paramConfig[i].AzimuthAngle == 180°) {
            compactParamConfig[idx].pairType = CENTER;
            compactParamConfig[idx].originalPosition = i;
        } else {
            j = SearchForSymmetricSpeaker(paramConfig, paramCount, i);
            if (j != -1) {
                compactParamConfig[idx].pairType = SYMMETRIC;
                if (paramConfig.AzimuthDirection == 0) {
                    compactParamConfig[idx].originalPosition = i;
                    compactParamConfig[idx].symmetricPair.originalPosition = j;
                } else {
                    compactParamConfig[idx].originalPosition = j;
                    compactParamConfig[idx].symmetricPair.originalPosition = i;
                }
                alreadyUsed[j] = 1;
            } else {
                compactParamConfig[idx].pairType = ASYMMETRIC;
                compactParamConfig[idx].originalPosition = i;
            }
        }
        idx++;
    }

    compactParamCount = idx;

    return (compactParamConfig, compactParamCount);
}


CalculateTotalCount()
{
    if (mixLFEOnlyToLFE) {
        compactInputLFECount = 0;
        compactOutputLFECount = 0;
        for (i = 0; i < compactInputCount; i++) {
            if (compactInputConfig[i].isLFE) compactInputLFECount++;
        }
        for (i = 0; i < compactOutputCount; i++) {
            if (compactOutputConfig[i].isLFE) compactOutputLFECount++;
        }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
        totalCount = (compactInputCount - compactInputLFECount) *
            (compactOutputCount - compactOutputLFECount);
    } else {
        totalCount = compactInputCount * compactOutputCount;
    }
    return totalCount;
}
```

The function `FindCompactTemplate(inputConfig, inputCount, outputConfig, outputCount)` is used to find a compact template matrix matching the input channel configuration represented by *inputConfig* and *inputCount*, and the output channel configuration represented by *outputConfig* and *outputCount*.

The compact template matrix is found by searching in a predefined list of compact template matrices, available at both the encoder and decoder, for the one with the same the list of input speakers as *inputConfig* and the same list of output speakers as *outputConfig*.

Use of template matrices for efficient coding of downmix matrices is only allowed when both the input and output channel configurations exactly correspond to a codepoint ChannelConfiguration of ISO/IEC 23001-8 and if an exactly matching template matrix is defined in this standard. This ensures that the correct order of channels are obeyed at the decoder as well as at the encoder. Template matrices are defined in this standard only for pairs of ChannelConfiguration codepoints. For matching of a template matrix, if an input or output configuration is given, instead of a ChannelConfiguration codepoint, as a list of predefined speakers indexes or geometry description according to 5.2.2.2, an additional check shall be made to verify whether this provided description exactly matches a ChannelConfiguration codepoint. In case of an exact match, the corresponding ChannelConfiguration codepoint will be therefore used in the selection procedure of a matching template matrix.

The function `SearchForSymmetricSpeaker(paramConfig, paramCount, i)` is used to search the channel configuration represented by *paramConfig* and *paramCount* for the symmetric speaker corresponding to the speaker *paramConfig[i]*. This symmetric speaker, *paramConfig[j],* shall be situated after the speaker *paramConfig[i]*, therefore *j* can be in the range *i+1* to paramCount – *1*, inclusive.

#### 5.4.2.5 Gain Value Quantization

The function `readRange()` is used to read a uniformly distributed integer in the range [0 *alphabetSize*-1] inclusive.

The function `generateGainTable(maxGain, minGain, precisionLevel)` is used to dynamically generate the gain table *gainTable* which contains the list of all possible gains between *minGain* and *maxGain* with precision *precisionLevel*. The table is terminated by adding an element signaling minus infinity dB gain having the placeholder value *minGain-1*. The order of the values is chosen so that the most frequently used values and also more “round” values would be typically closer to the beginning of the list. The gain table of length gainTableSize with the list of all possible gain values is generated as follows:

— add integer multiples of 3 dB, going down from 0 dB to *minGain*;

— add integer multiples of 3 dB, going up from 3 dB to *maxGain*;

— add remaining integer multiples of 1 dB, going down from 0 dB to *minGain*;

— add remaining integer multiples of 1 dB, going up from 1 dB to *maxGain*;

— stop here if *precisionLevel* is 0 (corresponding to 1 dB) and add *minGain-1*;

— add remaining integer multiples of 0.5 dB, going down from 0 dB to *minGain*;

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

— add remaining integer multiples of 0.5 dB, going up from 0.5 dB to *maxGain*;

— stop here if *precisionLevel* is 1 (corresponding to 0.5 dB) and add *minGain-1*;

— add remaining integer multiples of 0.25 dB, going down from 0 dB to *minGain*;

— add remaining integer multiples of 0.25 dB, going up from 0.25 dB to *maxGain*;

— add *minGain-1.*

EXAMPLE When *maxGain* is 2 dB and *minGain* is -6 dB, and precisionLevel is 0.5 dB, the following list is created: 0, -3, -6, -1, -2, -4, -5, 1, 2, -0.5, -1.5, -2.5, -3.5, -4.5, -5.5, 0.5, 1.5, -7.

#### 5.4.2.6 Equalizer Config

| | |
|---|---|
| numEqualizers | Number of different equalizer filters present |
| eqPrecisionLevel | Precision used for uniform quantization of the gains:<br>0 = 1 dB, 1 = 0.5 dB, 2 = 0.25 dB, 3 = 0.1 dB |
| eqExtendedRange | Boolean indicating whether to use an extended range for the gains; if enabled, the available range is doubled |
| numSections | Number of sections of an equalizer filter, each one being a peak filter |
| centerFreqLd2 | The leading two decimal digits of the center frequency for a peak filter; the maximum range is 10 .. 99 |
| centerFreqP10 | Number of zeros to be appended to centerFreqLd2; the maximum range is 0 .. 3 |
| qFactorIndex | Quality factor index for a peak filter |
| qFactorExtra | Extra bits for decoding a quality factor larger than 1.0 |
| centerGainIndex | Gain at the center frequency for a peak filter |
| scalingGainIndex | Scaling gain for an equalizer filter |
| hasEqualizer[i] | Boolean indicating whether the input channel with index *i* has an equalizer associated to it |
| equalizerIndex[i] | The index of the equalizer associated with the input channel with index *i* |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

#### 5.4.2.7 Decoding of Downmix Coefficients

The syntax element *DownmixMatrixSet()* contains one or more downmix matrices that may be applied to achieve a format conversion to a desired loudspeaker layout. A given bit stream must not contain more than one instance of *DownmixMatrixSet()*.

The syntax element *DownmixMatrix()* contains the downmix matrix information. The decoder first reads the equalizer information represented by the syntax element *EqualizerConfig(),* if enabled. The fields *precisionLevel*, *maxGain,* and *minGain* are then read. The input and output configurations are converted to compact configurations using the function *ConvertToCompactConfig()*. Then, the flags indicating if the separability and symmetry properties are satisfied for each output speaker group are read.


© ISO/IEC 2015 – All rights reserved

The significance matrix *compactDownmixMatrix* is then read, either a) raw using one bit per entry, or b) using the limited Golomb-Rice coding of the run lengths, and then copying the decoded bits from *flatCompactMatrix* to *compactDownmixMatrix* and applying the *compactTemplate* matrix.

Finally, the nonzero gains are read. For each nonzero entry of *compactDownmixMatrix*, depending on the field *pairType* of the corresponding input group and the field *pairType* of the corresponding output group, a sub-matrix of size up to 2 by 2 has to be reconstructed. Using the associated separability and symmetry properties, a number of gain values are read using the function *DecodeGainValue()*. A gain value can be decoded uniformly, by using the function *ReadRange()*, or using the limited Golomb-Rice decoding of the index of the gain in the *gainTable* table, which contains all the possible gain values.

#### 5.4.2.8 Decoding of Equalizer Config

The syntax element *EqualizerConfig()* contains the equalizer information that is to be applied to the input channels. A number of *numEqualizers* equalizer filters is first decoded and thereafter selected for specific input channels using *equalizerIndex[i]*. The fields *eqPrecisionLevel* and *eqExtendedRange* indicate the quantization precision and the available range of the scaling gains and of the peak filter gains.

Each equalizer filter is a serial cascade consisting in a number of *numSections* of peak filters and one *scalingGain*. Each peak filter is fully defined by its *centerFreq*, *qualityFactor*, and *centerGain*.

The *centerFreq* parameters of the peak filters which belong to a given equalizer filter must be given in ascending order. The parameter is limited to 10 .. 24000 Hz inclusive, and it is calculated as

$$centerFreq = centerFreqLd2 \times 10^{centerFreqP10}$$

The *qualityFactor* parameter of the peak filter can represent values between 0.05 and 1.0 inclusive with a precision of 0.05 and from 1.1 to 10.6 inclusive with a precision of 0.1 and it is calculated as

$$\text{qualityFactor} = \begin{cases} 0.05 \times (\text{qFactorIndex} + 1), & \text{if qFactorIndex} \leq 19 \\ 1.0 + 0.1 \times \left[(\text{qFactorIndex} - 20) \times 8 + \text{qFactorExtra} + 1\right], & \text{otherwise} \end{cases}$$

The vector *eqPrecisions* gives the precision in dB corresponding to a given *eqPrecisionLevel*, and the *eqMinRanges* and *eqMaxRanges* matrices give the minimum and maximum values in dB for the gains corresponding to a given *eqExtendedRange* and *eqPrecisionLevel*.

*eqPrecisions*[4] = {1.0, 0.5, 0.25, 0.1};

*eqMinRanges*[2][4] = {{-8.0, -8.0, -8.0, -6.4}, {-16.0, -16.0, -16.0, -12.8}};

*eqMaxRanges*[2][4] = {{7.0, 7.5, 7.75, 6.3}, {15.0, 15.5, 15.75, 12.7}};

The parameter *scalingGain* uses the precision level $\min(\text{eqPrecisionLevel} + 1, 3)$, which is the next better precision level if not already the last one. The mappings from the fields *centerGainIndex* and *scalingGainIndex* to the gain parameters *centerGain* and *scalingGain* are calculated as

$$centerGain = eqMinRanges[eqExtendedRange][eqPrecisionLevel] + eqPrecisions[eqPrecisionLevel] \times centerGainIndex$$

$$scalingGain = eqMinRanges[eqExtendedRange][\min(eqPrecisionLevel + 1, 3)] + eqPrecisions[\min(eqPrecisionLevel + 1, 3)] \times scalingGainIndex$$

### 5.4.3 HOA rendering matrix configuration

#### 5.4.3.1 General

Higher Order Ambisonics (HOA) rendering matrices may be transmitted by the encoder to enable control over the HOA rendering process at the decoder. Transmission is facilitated by means of a

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

mpegh3daConfigExtension of Type ID_CONFIG_EXT_HOA_MATRIX. The mpegh3daConfigExtension may contain several HOA rendering matrices for different loudspeaker reproduction configurations. If HOA rendering matrices are transmitted, each HOA rendering matrix signals its associated target loudspeaker layout that determines together with the *HoaOrder* (see Table 119) the dimensions of the rendering matrix.

The transmission of a unique *HoaRenderingMatrixId* allows referencing to a default HOA rendering matrix available on decoder side, or to a transmitted HOA rendering matrix from outside of the audio stream.

A signaled HOA rendering matrix shall expect full 3D normalised (N3D) HOA coefficients as its input. Further, a rendering matrix shall adhere to the ordering of the HOA coefficients $c_n^m$ that is used by the HOA decoder, where $n$ is the order, $m$ is the degree, and $N$ is the maximum HOA order. (see also Annex F):

$$\boldsymbol{c} = \left[c_0^0 \quad c_1^{-1} \quad c_1^0 \quad c_1^1 \quad c_2^{-2} \quad c_2^{-1} \quad c_2^0 \quad c_2^1 \quad c_2^2 \quad \ldots c_N^{N-1} \quad c_N^N\right]^T$$

#### 5.4.3.2 Helper Functions

The function `findSymmetricSpeakers` (not displayed here) indicates the total number and position of all loudspeaker pairs within the provided loudspeaker setup which are left-right symmetric for a listener at the sweet spot.

```
int findSymmetricSpeakers(int outputCount, SpeakerInformation* outputConfig, int
hasLfeRendering);
```

The function *createSymSigns* is used to compute a vector of 1 and -1 values which is used to generate the matrix elements associated with symmetric loudspeakers.

```
void createSymSigns(int* symSigns, int hoaOrder)
{
    int n, m, k = 0;
    for (n = 0; n<=hoaOrder; ++n) {
        for (m = -n; m<=n; ++m)
            symSigns[k++] = ((m>=0)*2)-1;
    }
}
```

The function *create2dBitmask* generates a bitmask to identify the HOA coefficients that are only used in the horizontal plane.

```
void create2dBitmask(int* bitmask, int hoaOrder)
{
    int n, m, k = 0;
    bitmask[k++] = 0;
    for (n = 1; n<=hoaOrder; ++n) {
        for (m = -n; m<=n; ++m)
            bitmask[k++] = abs(m)!=n;
    }
}
```

#### 5.4.3.3 Decoding of HOA Rendering Matrix Coefficients

The syntax element *HoaRenderingMatrixSet()* contains one or more HOA rendering matrices that may be applied to achieve HOA rendering to a desired loudspeaker layout. A given bit stream shall not contain more than one instance of *HoaRenderingMatrixSet()*.

The syntax element *HoaRenderingMatrix()* contains the HOA rendering matrix information. The decoder first reads the config information that guides the reading and matrix decoding process. After that the matrix elements are read from the bit stream accordingly.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

At the beginning the fields `precisionLevel` and `gainLimitPerOrder` are read. If the flag `gainLimitPerOrder` is set, then the following fields `maxGain`, and `minGain` are read and decoded for each HOA order separately. If the flag `gainLimitPerOrder` is not set, then the fields `maxGain` and `minGain` are only read and decoded once and applied to all HOA orders during the decoding process. The `minGain` value must be between 0db and -101 dB. The `maxGain` value must be between 1dB and 111dB lower than the minGain value.

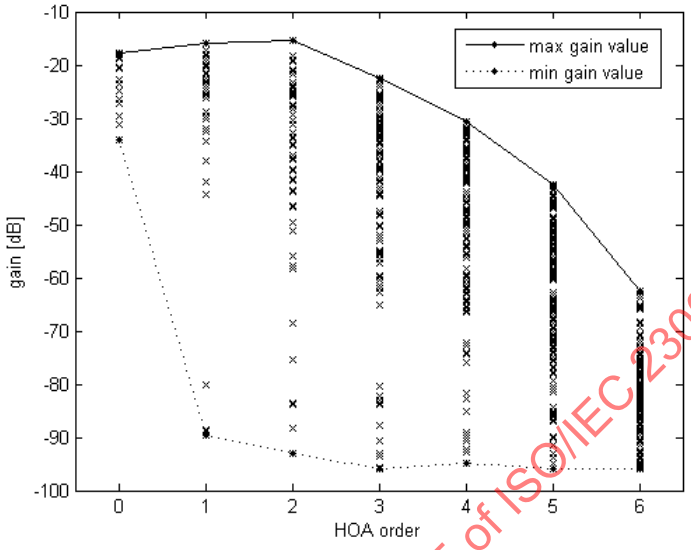



**Figure 3 — Example for HOA order dependent min and max gains within a HOA rendering matrix**

The next field to read is the flag `isFullMatrix` which signals if a matrix is defined as full or as partially sparse. In case the matrix is defined as partially sparse, the next field `firstSparseOrder` specifies the starting HOA order, after which the HOA rendering matrix is sparsely coded. For example, if `firstSparseOrder` signals 4, then all matrix elements associated with the HOA order 4 and above are sparsely coded. HOA rendering matrices are dense for low order and often become sparse in the higher orders, depending on the loudspeaker reproduction setup. As an example Figure 4 depicts a partially sparse 6$^{th}$ order HOA rendering matrix for 22 loudspeakers.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

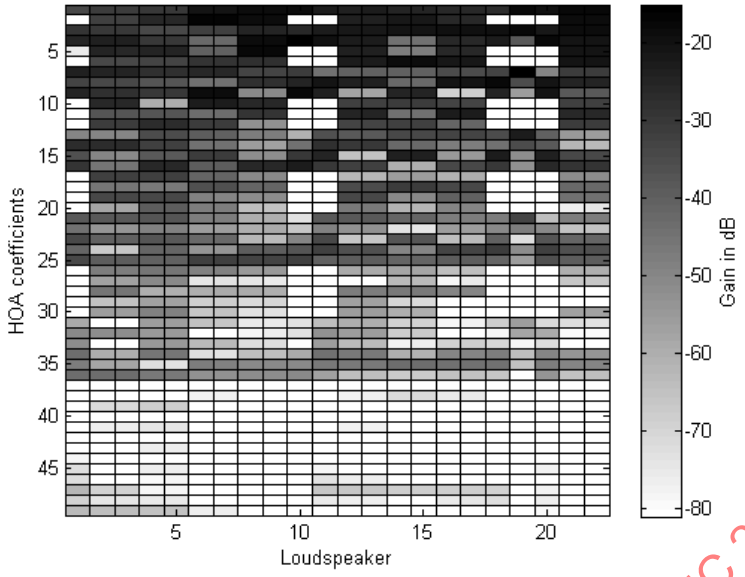



**Figure 4 — Example for a HOA rendering matrix with sparsity starting at the 26th HOA coefficient (HOA order 5)**

Depending whether LFE channels exist within the loudspeaker reproduction setup, the field `hasLfeRendering` is read. If `hasLfeRendering` is not set, then the matrix elements related to the LFE channels are assumed to be digital zeroes. The next field read by the decoder is the flag `zerothOrderAlwaysPositive` and signals if the matrix elements associated with the coefficient of the 0th order are positive. In this case no number signs are coded.

In the following, properties of the HOA rendering matrix are signaled for loudspeaker pairs symmetric with regards to the median plane. There are two symmetry properties: a) value symmetry and b) sign symmetry. In the case of value symmetry, the matrix elements of the left loudspeaker of the symmetric loudspeaker pair are not coded, but derived from the decoded matrix elements of the right loudspeaker by employing the helper function `createSymSigns`:

```
pairIdx = outputConfig[j].symmetricPair->originalPosition;
hoaMatrix[i * outputCount + j] = hoaMatrix[i * outputCount + pairIdx];
signMatrix[i * outputCount + j] = symSigns[i] * signMatrix[i * outputCount + pairIdx];
```

If a loudspeaker pair is not value symmetric, then the matrix elements may be symmetric with regards to their number signs. If a loudspeaker pair is sign symmetric, the number signs of the matrix elements of the left loudspeaker of the symmetric loudspeaker pair are not coded, but derived from the number signs of the matrix elements associated to the right loudspeaker by employing the helper function `createSymSigns`:

```
pairIdx = outputConfig[j].symmetricPair->originalPosition;
signMatrix[i * outputCount + j] = symSigns[i] * signMatrix[i * outputCount + pairIdx];
```

The signalling of the symmetry properties is visualised in Figure 5. A loudspeaker pair cannot be defined as value symmetric and sign symmetric at the same time. The final decoding flag `hasVerticalCoef` specifies if only the matrix elements associated with circular (i.e., 2D) HOA coefficients are coded. If `hasVerticalCoef` is not set the matrix elements associated with the HOA coefficients defined with the helper function `create2dBitmask` are set to digital zero.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

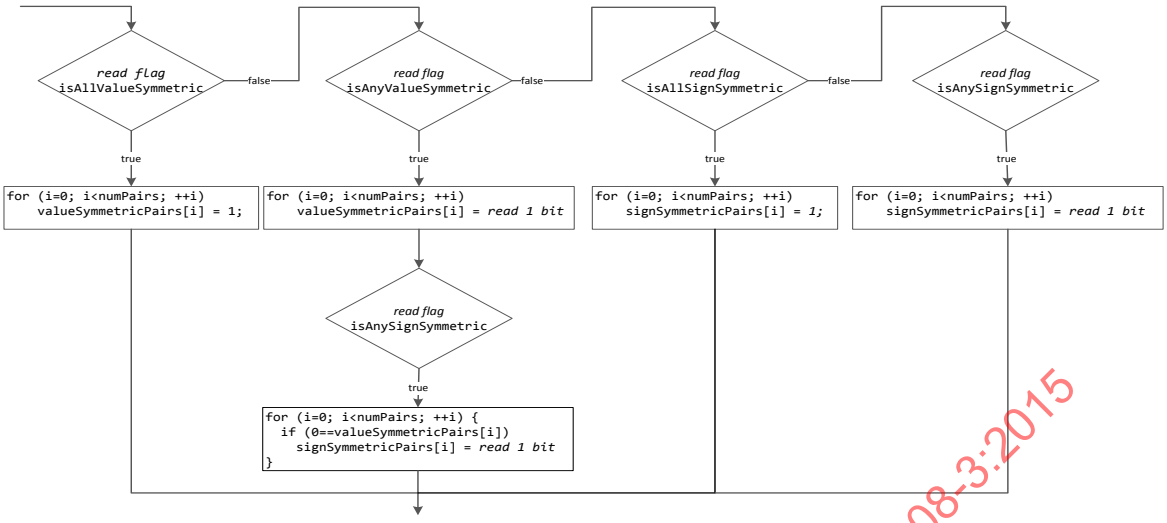



**Figure 5 — Signaling of symmetry properties in HOA rendering matrices**

Using the above mentioned properties, a series of matrix element gain values are read. To read the absolute gain value, the function *DecodeGainValue()* is used. The gain values are uniformly decoded by using the function *ReadRange()* of the alphabet index. If the decoded gain value is not digital zero, the number sign value (`signVal`) is additionally read (see also Figure 6a). If the matrix element is associated with an HOA coefficient that was signaled to be sparse (via `isHoaCoefSparse`), the `hasValue` flag precedes the `gainValueIndex` (see Figure 6b). If the `hasValue` flag is zero, this element is set to digital zero and no `gainValueIndex` and `signVal` are signalled.

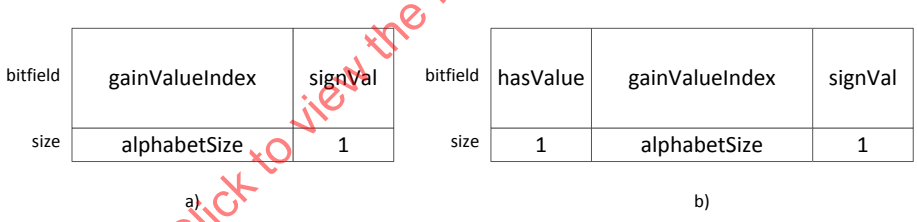



**Figure 6 — Examples for bit stream syntax to decode a rendering matrix element**

Depending on the specified symmetry properties for loudspeaker pairs, the matrix elements associated with the left loudspeaker are derived from the right loudspeaker. In this case the code words to decode the individual matrix elements for the left loudspeaker are reduced or completely omitted accordingly.

## 5.5 Tool Descriptions

### 5.5.1 General

The tool descriptions are based on ISO/IEC 23003-3:2012, clause 7.

Modifications and amendments to the existing tool descriptions are listed below.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

### 5.5.2 Quad Channel Element

#### 5.5.2.1 Tool description

The Quad Channel Element (QCE) is a method for joint coding of four channels for more efficient coding of horizontally and vertically distributed channels. A QCE consists of two consecutive CPEs and is formed by hierarchically combining the Joint Stereo tool with possibility of Complex Stereo Prediction in horizontal direction and the MPEG Surround based stereo tool in vertical direction. This is achieved by enabling both stereo tools and swapping output channels between applying the tools. Stereo SBR is performed in horizontal direction to preserve the left-right relations of high frequencies.

#### 5.5.2.2 Terms and Definitions

**Help elements:**

| | |
|---|---|
| cplx_out_dmx_L[ ] | First channel of first CPE after complex prediction stereo decoding. |
| cplx_out_dmx_R[ ] | Second channel of first CPE after complex prediction stereo decoding. |
| cplx_out_res_L[ ] | Second CPE after complex prediction stereo decoding.<br>(zero if qceIndex = 1) |
| cplx_out_res_R[ ] | Second channel of second CPE after complex prediction stereo decoding.<br>(zero if qceIndex = 1) |
| mps_out_L_1[ ] | First output channel of first MPS box. |
| mps_out_L_2 [ ] | Second output channel of first MPS box. |
| mps_out_R_1[ ] | First output channel of second MPS box. |
| mps_out_R_2[ ] | Second output channel of second MPS box. |
| sbr_out_L_1[ ] | First output channel of first Stereo SBR box. |
| sbr_out_R_1[ ] | Second output channel of first Stereo SBR box. |
| sbr_out_L_2[ ] | First output channel of second Stereo SBR box. |
| sbr_out_R_2[ ] | Second output channel of second Stereo SBR box. |

#### 5.5.2.3 Decoding process

The syntax element **qceIndex** in mpegh3daChannelPairElementConfig() indicates whether a CPE belongs to a QCE and if residual coding is used. In case that **qceIndex** is unequal 0, the current CPE forms a QCE together with its subsequent element which shall be a CPE having the same **qceIndex**. Stereo SBR is always used for the QCE, thus the syntax item **stereoConfigIndex** shall be 3 and **bsStereoSbr** shall be 1.

In case of **qceIndex** == 1 only the payloads for MPEG Surround and SBR and no relevant audio signal data is contained in the second CPE and the syntax element **bsResidualCoding** is set to 0.

The presence of a residual signal in the second CPE is indicated by **qceIndex** == 2. In this case the syntax element **bsResidualCoding** is set to 1.

Decoding of Joint Stereo is performed as specified in 7.7 of ISO/IEC 23003-3:2012. The resulting output of the first CPE are the MPS downmix signals cplx_out_dmx_L[] and cplx_out_dmx_R[]. If residual coding is used

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

(i.e. **qceIndex** == 2), the output of the second CPE are the MPS residual signals cplx_out_res_L[], cplx_out_res_R[], if no residual signal has been transmitted (i.e. **qceIndex** == 1), zero signals are inserted.

The structure of the QCE decoding process is illustrated in the following schematics (Figure 7).

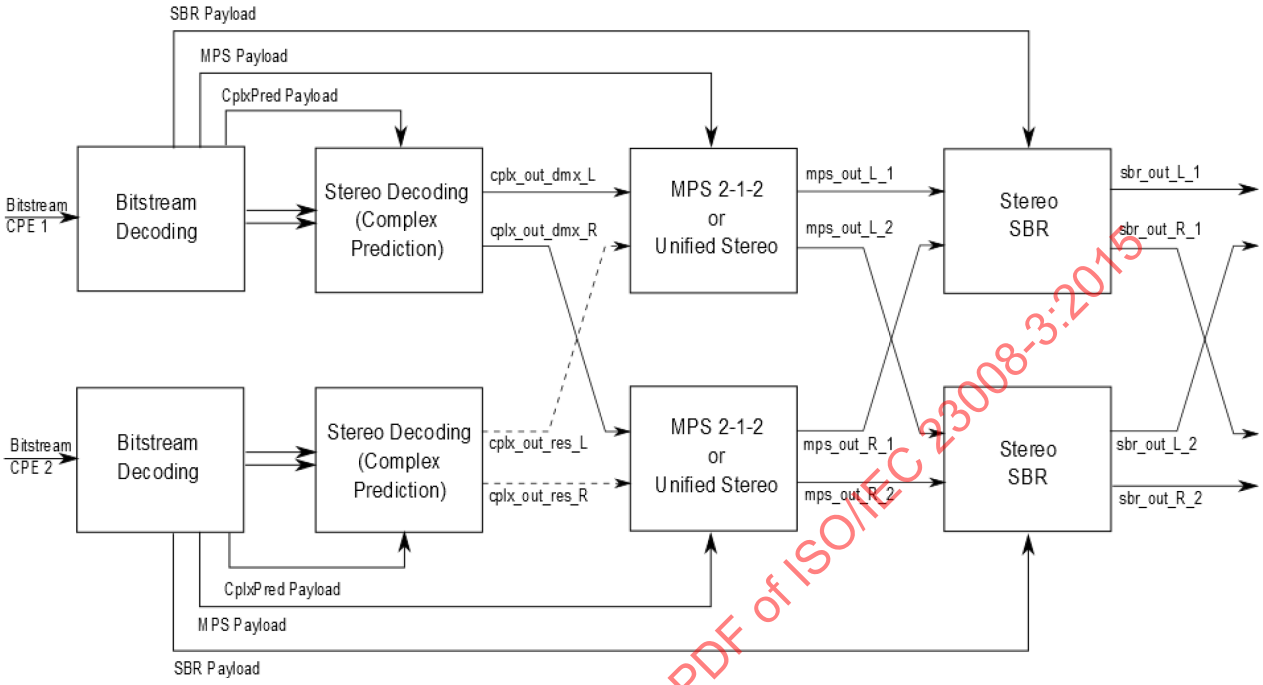



**Figure 7 — QCE Decoder Schematics**

Before applying MPEG Surround decoding, the second channel of the first element (cplx_out_dmx_R[]) and the first channel of the second element (cplx_out_res_L[]) are swapped.

Decoding of MPEG Surround with residual is performed as specified in 7.11 of ISO/IEC 23003-3:2012. Decoding of MPEG Surround without residual using SBR as defined in ISO/IEC 23003-3:2012, subclause 7.11.2.7 (Figure 23), is modified so that Stereo SBR is also used for **bsResidualCoding** == 1, resulting in the following decoder schematics (see Figure 8).

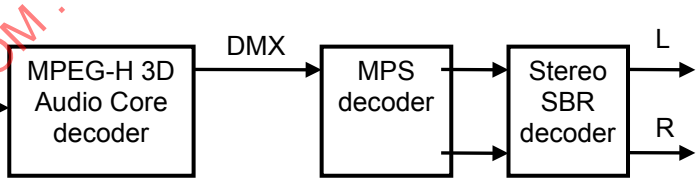



**Figure 8 — bsResidualCoding == 0, bsStereoSbr == 1**

Before applying Stereo SBR, the second channel of the first element (mps_out_L_2[]) and the first channel of the second element (mps_out_R_1[]) are swapped to allow right-left Stereo SBR. After application of Stereo SBR, the second output channel of the first element (sbr_out_R_1[]) and the first channel of the second element (sbr_out_L_2[]) are swapped again to restore the input channel order.

### 5.5.3 Transform Splitting

#### 5.5.3.1 Tool Description

When Transform splitting (TS) is active in a long transform, two half-length MDCTs are employed instead of one full-length MDCT. The coefficients of the two MDCTs are transmitted in a line-by-line interleaved fashion

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

as a traditional frequency domain (FD) transform, with the coefficients of the first-in-time MDCT placed at even and the coefficients of the second-in-time MDCT placed at odd indices.

#### 5.5.3.2 Operational Constraints

TS can only be used in a FD *long-start* or *stop-start* window (**window_sequence** == 1). Also, TS can only be applied when **noiseFilling** is 1 in mpegh3daCoreConfig( ). When TS is signaled, all FD tools except for TNS and inverse MDCT operate on the interleaved (long) set of TS coefficients. This allows the reuse of the scalefactor band offset and long-transform arithmetic coder tables as well as the window shapes and overlap lengths.

#### 5.5.3.3 Terms and Definitions

**Help Elements:**

| | |
|---|---|
| **common_window** | indicates if channel 0 and channel 1 of a CPE use identical window parameters (see ISO/IEC 23003-3:2012 subclause 6.2.5.1.1). |
| **window_sequence** | FD window sequence type for the current frame and channel (see ISO/IEC 23003-3:2012 subclause 6.2.9). |
| **tns_on_lr** | indicates the mode of operation for TNS filtering (see ISO/IEC 23003-3:2012 subclause 7.8.2). |
| **noiseFilling** | this flag signals the usage of the noise filling of spectral holes in the FD core coder (see ISO/IEC 23003-3:2012 subclause 6.1.1.1). |
| **noise_offset** | noise-fill offset to modify scale factors of zero-quantized bands (see ISO/IEC 23003-3:2012 subclause 7.2). |
| **noise_level** | noise-fill level representing amplitude of added spectrum noise (see ISO/IEC 23003-3:2012 subclause 7.2). |
| split_transform | binary flag indicating whether TS is utilized in the current frame and channel. |
| half_transform_length | one half of coreCoderFrameLength (ccfl, the transform length, see ISO/IEC 23003-3:2012 subclause 6.1.1). |
| half_lowpass_line | one half of the number of MDCT lines transmitted for the current channel. |

#### 5.5.3.4 Decoding Process

##### 5.5.3.4.1 General

The decoding of an FD (stop-)start transform with TS is performed in three sequential steps as follows.

##### 5.5.3.4.2 Decoding of split_transform and half_lowpass_line

The help element split_transform does not represent an independent bit-stream element but is derived from the noise filling elements, **noise_offset** and **noise_level**, and in case of a mpegh3daChannelPairElement(), the **common_window** flag in StereoCoreToolInfo(). If **noiseFilling** == 0, split_transform is 0. Otherwise,

```
if ((noiseFilling != 0) &&
    (noise_level == 0) &&
    (noise_offset != 0) &&
    (window_sequence == 1)) {
  split_transform = 1;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
  noise_level      = (noise_offset & 28) / 4;
  noise_offset     = (noise_offset & 3) * 8;
}
else {
  split_transform = 0;
}
```

In other words, if split_transform == 1, **noise_offset** contains 5 bits of noise filling data, which are then rearranged. Since this operation changes the values of **noise_level** and **noise_offset**, it must be executed before the noise filling process of ISO/IEC 23003-3:2012 subclause 7.2.

Furthermore, if **common_window** == 1 in a mpegh3daChannelPairElement(), split_transform is determined only in the left (first) channel; the right channel's split_transform is set equal to (i.e. copied from) the left channel's split_transform, and the following pseudo-code is not executed in the right channel.

The help element half_lowpass_line is determined from the "long" scalefactor band offset table, swb_offset_long_window, and the max_sfb of the current channel, or in case of stereo and **common_window** == 1, max_sfb_ste.

$$\text{lowpass_sfb} = \begin{cases} \text{max_sfb_ste} & \text{in elements with StereoCoreToolInfo() and } \textbf{common_window} == 1, \\ \text{max_sfb} & \text{otherwise.} \end{cases}$$

Based on the **enhancedNoiseFilling** flag, half_lowpass_line is derived:

```
if (enhancedNoiseFilling != 0) {
  lowpass_sfb = max(lowpass_sfb, m_igFStopSfb);
}
half_lowpass_line = swb_offset_long_window[lowpass_sfb] / 2;
```

#### 5.5.3.4.3 De-interleaving of Half-length Spectra for Temporal Noise Shaping

After spectrum de-quantization, noise filling, and scalefactor application and prior to the application of Temporal Noise Shaping (TNS), the TS coefficients in spec[] are de-interleaved using a helper buffer[]:

```
for (i = 0, i2 = 0; i < half_lowpass_line; i += 1, i2 += 2) {
  spec[i]   = spec[i2]; /* isolate 1st window */
  buffer[i] = spec[i2+1]; /* isolate 2nd window */
}
for (i = 0; i < half_lowpass_line; i += 1) {
  spec[i+half_lowpass_line] = buffer[i]; /* copy 2nd window */
}
```

The in-place de-interleaving effectively places the two half-length TS spectra on top of each other, and the TNS tool now operates as usual on the resulting full-length pseudo-spectrum.

#### 5.5.3.4.4 Temporary Re-interleaving, Two sequential Inverse MDCTs

If **common_window** == 1 in the current frame or the stereo decoding is performed after TNS decoding (**tns_on_lr** == 0 in ISO/IEC 23003-3:2012 subclause 7.8), spec[] must be re-interleaved temporarily into a full-length spectrum:

```
for (i = 0; i < half_lowpass_line; i += 1) {
  buffer[i] = spec[i]; /* copy 1st window */
}
for (i = 0, i2 = 0; i < half_lowpass_line; i += 1, i2 += 2) {
  spec[i2]   = buffer[i]; /* merge 1st window */
  spec[i2+1] = spec[i+half_lowpass_line];   /* merge 2nd window */
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

The resulting pseudo-spectrum is used for stereo decoding (ISO/IEC 23003-3:2012 subclause 7.7) and to update dmx_re_prev[] (ISO/IEC 23003-3:2012 subclause 7.7.2 and 5.5.3.6). In case of **tns_on_lr** == 0, the stereo-decoded full-length spectra are again de-interleaved by repeating the process of subclause 5.5.3.4.3. Finally, the 2 inverse MDCTs are calculated with ccfl and the channel's **window_shape** of the current and last frame.

### 5.5.3.5 Filterbank and block switching

#### 5.5.3.5.1 IMDCT

The processing for TS follows the description given in ISO/IEC 23003-3:2012 subclause "7.9 Filterbank and block switching". The following additions shall be taken into account.

The TS coefficients in spec[] are de-interleaved using a helper buffer[] with *N*, the window length based on the window_sequence value:

```
for (i = 0, i2 = 0; i < N/2; i += 1, i2 += 2) {
  spec[0][i]   = spec[i2]; /* isolate 1st window */
  buffer[i] = spec[i2+1];  /* isolate 2nd window */
}
for (i = 0; i < N/2; i += 1) {
  spec[1][i] = buffer[i];  /* copy 2nd window */
}
```

The IMDCT for the half-length TS spectrum is then defined as:

$$x_{(0,1),n} = \frac{2}{N}\sum_{k=0}^{\frac{N}{4}-1} spec[(0,1)][k]\cos\left(\frac{4\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right) \quad \text{for } 0 \le n < \frac{N}{2}$$

where:

n = sample index

k = spectral coefficient index

$$n_0 = (N/4+1)/2$$

Subsequent windowing and block switching steps are defined in the next subclauses.

#### 5.5.3.5.2 Transform Splitting with STOP_START_SEQUENCE

The STOP_START_SEQUENCE in combination with transform splitting is depicted in Figure 9. It comprises two overlapped and added half-length windows with a length of $N_l/2$ which is 1024 (768). $N_s$ is set to 256 (192) respectively.

**Figure 9 — Transform splitting for STOP_START_WINDOW**

The windows (0,1) for the two half-length IMDCTs are given as follows:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$W_{(0,1)}(n)=\begin{cases}0.0, & \text{for } 0\le n<\dfrac{N_l/2-N_s}{4}\\ W_{(0,1),LEFT,N_s}(n-\dfrac{N_l/2-N_s}{4}), & \text{for } \dfrac{N_l/2-N_s}{4}\le n<\dfrac{N_l/2+N_s}{4}\\ 1.0, & \text{for } \dfrac{N_l/2+N_s}{4}\le n<\dfrac{3N_l/2-N_s}{4}\\ W_{(0,1),RIGHT,N_s}(n-\dfrac{3N_l/2-N_s}{4}), & \text{for } \dfrac{3N_l/2-N_s}{4}\le n<\dfrac{3N_l/2+N_s}{4}\\ 0.0, & \text{for } \dfrac{3N_l/2+N_s}{4}\le n<N_l/2\end{cases}$$

where for the first IMDCT the windows

$$W_{0,LEFT,N_s}(n)=\begin{cases}W_{KBD_LEFT,N_s}(n), \text{ if window_shape_previous_block == 1}\\ W_{SIN_LEFT,N_s}(n), \text{ if window_shape_previous_block == 0}\end{cases},$$

$$W_{0,RIGHT,N_s}(n)=\begin{cases}W_{KBD_RIGHT,N_s}(n), \text{ if window_shape == 1}\\ W_{SIN_RIGHT,N_s}(n), \text{ if window_shape == 0}\end{cases}$$

are applied and for the second IMDCT the windows

$$W_{1,LEFT,N_s}(n)=\begin{cases}W_{KBD_LEFT,N_s}(n), \text{ if window_shape == 1}\\ W_{SIN_LEFT,N_s}(n), \text{ if window_shape == 0}\end{cases},$$

$$W_{1,RIGHT,N_s}(n)=\begin{cases}W_{KBD_RIGHT,N_s}(n), \text{ if window_shape == 1}\\ W_{SIN_RIGHT,N_s}(n), \text{ if window_shape == 0}\end{cases}$$

are applied.

The overlap and add between the two half-length windows resulting in the windowed time domain values $z_{i,n}$ is specified as follows. Here, $N_l$ is set to 2048 (1536), $N_s$ to 256 (192) respectively:

$$Z_{i,n}(n)=\begin{cases}0.0, & \text{for } 0\le n<N_s\\ x_{0,n-N_s}\cdot W_0(n-N_s), & \text{for } N_s\le n<\dfrac{2N_l-N_s}{4}\\ x_{0,n-N_s}\cdot W_0(n-N_s)+x_{1,n-(N_l/2-N_s)}\cdot W_1(n-(N_l/2-N_s)), & \text{for } \dfrac{2N_l-N_s}{4}\le n<\dfrac{2N_l+N_s}{4}\\ x_{1,n-(N_l/2-N_s)}\cdot W_1(n-(N_l/2-N_s)), & \text{for } \dfrac{2N_l+N_s}{4}\le n<N_l-N_s\\ 0.0, & \text{for } N_l-N_s\le n<N_l\end{cases}$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

#### 5.5.3.5.3 Transform Splitting with LONG_START_SEQUENCE

The LONG_START_SEQUENCE in combination with transform splitting is depicted in Figure 10.

**Figure 10 — Transform splitting for LONG_START_WINDOW**

© ISO/IEC 2015 – All rights reserved

It comprises three windows defined as follows, where $N_l/$ is set to 1024 (768), $N_s$ is set to 256 (192) respectively.

$$W_0(n) = \begin{cases} 1.0, & \text{for } 0 \le n < \dfrac{3N_l/2 - N_s}{4} \\ W_{0,RIGHT,N_s}\left(n - \dfrac{3N_l/2 - N_s}{4}\right), & \text{for } \dfrac{3N_l/2 - N_s}{4} \le n < \dfrac{3N_l/2 + N_s}{4} \\ 0.0, & \text{for } \dfrac{3N_l/2 + N_s}{4} \le n < N_l/2 \end{cases}$$

$$W_1(n) = \begin{cases} 0.0, & \text{for } 0 \le n < \dfrac{N_l/2 - N_s}{4} \\ W_{1,LEFT,N_s}\left(n - \dfrac{N_l/2 - N_s}{4}\right), & \text{for } \dfrac{N_l/2 - N_s}{4} \le n < \dfrac{N_l/2 + N_s}{4} \\ 1.0, & \text{for } \dfrac{N_l/2 + N_s}{4} \le n < \dfrac{3N_l/2 - N_s}{4} \\ W_{1,RIGHT,N_s}\left(n - \dfrac{3N_l/2 - N_s}{4}\right), & \text{for } \dfrac{3N_l/2 - N_s}{4} \le n < \dfrac{3N_l/2 + N_s}{4} \\ 0.0, & \text{for } \dfrac{3N_l/2 + N_s}{4} \le n < N_l/2 \end{cases}$$

The left/right window halves are given by:

$$W_{1,LEFT,N_s}(n) = \begin{cases} W_{KBD_LEFT,N_s}(n), \text{ if window_shape} == 1 \\ W_{SIN_LEFT,N_s}(n), \text{ if window_shape} == 0 \end{cases},$$

$$W_{(0,1),RIGHT,N_s}(n) = \begin{cases} W_{KBD_RIGHT,N_s}(n), \text{ if window_shape} == 1 \\ W_{SIN_RIGHT,N_s}(n), \text{ if window_shape} == 0 \end{cases}$$

The third window equals the left half of a LONG_START_WINDOW:

$$W_2(n) = \begin{cases} W_{LEFT,N_l}(n), & \text{for } 0 \le n < N_l/2 \\ 1.0, & \text{for } N_l/2 \le n < N_l \end{cases}$$

with

$$W_{LEFT,N_l}(n) = \begin{cases} W_{KBD_LEFT,N_l}(n), \text{ if window_shape_previous_block} == 1 \\ W_{SIN_LEFT,N_l}(n), \text{ if window_shape_previous_block} == 0 \end{cases}$$

The overlap and add between the two half-length windows resulting in intermediate windowed time domain values $\tilde{Z}_{i,n}$ is specified as follows. Here, $N_l$ is set to 2048 (1536), $N_s$ to 256 (192) respectively.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$\tilde{Z}_{i,n}(n)=\begin{cases} -1\cdot x_{0,N_s-n-1}\cdot W_0(N_s-n-1), & \text{for } 0\le n<N_s \\ x_{0,n-N_s}\cdot W_0(n-N_s), & \text{for } N_s\le n<\dfrac{2N_l-N_s}{4} \\ x_{0,n-N_s}\cdot W_0(n-N_s)+x_{1,n-(N_l/2-N_s)}\cdot W_1(n-(N_l/2-N_s)), & \text{for } \dfrac{2N_l-N_s}{4}\le n<\dfrac{2N_l+N_s}{4} \\ x_{1,n-(N_l/2-N_s)}\cdot W_1(n-(N_l/2-N_s)), & \text{for } \dfrac{2N_l+N_s}{4}\le n<N_l-N_s \\ 0.0, & \text{for } N_l-N_s\le n<N_l \end{cases}$$

The final windowed time domain values $Z_{i,n}$ are obtained by applying $W_2$:

$$Z_{i,n}(n)=\tilde{Z}_{i,n}(n)\cdot W_2(n), \qquad \text{for} \qquad 0\le n<N_l$$

#### 5.5.3.6 Modification to Complex Prediction Stereo Decoding

Since the FD stereo tools operate on an interleaved pseudo-spectrum when TS is active in a channel pair, no changes are necessary to the underlying M/S or Complex Prediction processing. However, the derivation of the previous frame's downmix dmx_re_prev[] and the computation of the downmix MDST dmx_im[] in ISO/IEC 23003-3:2012 subclause 7.7.2 need to be adapted if TS is used in either channel in the last or current frame:

— use_prev_frame must be 0 if the TS activity changed in either channel from last to current frame. In other words, dmx_re_prev[] must not be used in that case due to transform length switching.

— If TS was or is active, dmx_re_prev[] and dmx_re[] are interleaved pseudo-spectra and must be de-interleaved into their corresponding two half-length TS spectra for correct MDST calculation.

— Upon TS activity, 2 half-length MDST downmixes are computed using adapted filter coefficients (filter_coefs in Table 57 and Table 58) and interleaved into a full-length spectrum dmx_im[ ] (just like dmx_re[]).

— window_sequence: Downmix MDST estimates are computed for each group window pair. use_prev_frame is evaluated only for the first of the two half-window pairs. For the remaining window pair, the preceding window pair is always used in the MDST estimate, which implies use_prev_frame = 1.

— Window shapes: The MDST estimation parameters for the current window, which are filter coefficients as specified below, depend on the shapes of the left and right window halves. For the first window, this means that the filter parameters are a function of the current and previous frames' window_shape flags. The remaining window is only affected by the current window_shape.

**Table 57 — MDST Filter Parameters for Current Window (filter_coefs)**

| **Current Window Sequence** | **Left Half: Sine Shape<br>Right Half: Sine Shape** | **Left Half: KBD Shape<br>Right Half: KBD Shape** |
|---|---|---|
| LONG_START_SEQUENCE<br>STOP_START_SEQUENCE | [0.185618, 0.000000, 0.627371, 0.000000,<br>-0.627371, 0.000000, -0.185618] | [ 0.204932, 0.000000, 0.634159, 0.000000,<br>-0.634159, 0.000000, -0.204932] |
| **Current Window Sequence** | **Left Half: Sine Shape<br>Right Half: KBD Shape** | **Left Half: KBD Shape<br>Right Half: Sine Shape** |
| LONG_START_SEQUENCE<br>STOP_START_SEQUENCE | [0.194609, 0.006202, 0.630536, 0.000000,<br>-0.630536, -0.006202, -0.194609] | [0.194609, -0.006202, 0.630536, 0.000000,<br>-0.630536, 0.006202, -0.194609] |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 58 — MDST Filter Parameters for Previous Window (filter_coefs_prev)**

| Current Window Sequence | Left Half of Current Window: Sine Shape | Left Half of Current Window: KBD Shape |
|---|---|---|
| LONG_START_SEQUENCE<br>STOP_START_SEQUENCE | [0.069608, 0.075028, 0.078423, 0.079580,<br>0.078423, 0.075028, 0.069608 ] | [0.042172, 0.043458, 0.044248, 0.044514,<br>0.044248, 0.043458, 0.042172] |

### 5.5.4 MPEG Surround for Mono to Stereo upmixing

#### 5.5.4.1 Calculation of pre-matrix M1 and mix-matrix M2

5.5.4.1.1 **General**

The calculation of mix-matrix M2, which is a interpolated version of $\mathbf{R}_2^{l,m}$, is done according to ISO/IEC 23003-3:2012, 7.11.2.3, but with the modifications specified in the following subclause.

5.5.4.1.2 **Upmix without decorrelation**

In case a format conversion step is included, the 4.3.3 MPEG Surround for Mono to Stereo upmixing tool is modified as follows.

10.3.4 defines a downmixing matrix $\mathrm{M}_{\mathrm{Dmx}}$ which is used to calculate a mix matrix $\mathrm{M}_{\mathrm{Mix}}$ as follows. Here $N_{in}$ is the number of source channels and $N_{out}$ is the number of destination channels.

```
M_Mix = zero N_in x N_in Matrix
for i = 1 to N_out
    for j = 1 to N_in
        set_j = 0
        if  M_Dmx(i, j) > 0.0
            set_j = 1
        end
        for k = 1 to N_in
            set_k = 0
            if  M_Dmx(i, k) > 0.0
                set_k = 1
            end
            if  set_j == 1 and set_k == 1
                M_Mix(j, k)= 1
            end
        end
    end
end
```

Each OTT decoding block yields two output signals corresponding to channel number *i* and *j*. If the mix matrix $\mathrm{M}_{\mathrm{Mix}}(i,j)$ equals one, decorrelation is switched off for this decoding block which means that the elements $H11_{OTT}^{l,m}$ and $H21_{OTT}^{l,m}$ of the upmix matrix $\mathrm{R}_2^{\mathrm{l,m}}$ shall be calculated by setting $ICC^{l,m}=1$.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

### 5.5.4.2 Combined Parametric and Residual Decoding (Hybrid Residual Coding)

#### 5.5.4.2.1 **Overview**

In addition to using either decorrelator based mono to stereo upmixing or residual coding as specified in 7.11.1 of ISO/IEC 23003-3:2012, Hybrid Residual Coding allows a signal dependent combination of both modes. Residual signal and decorrelator output are blended together, using time and frequency dependent weighting factors depending on the signal energies and the spatial parameters, as illustrated in the schematics below (Figure 11 ).

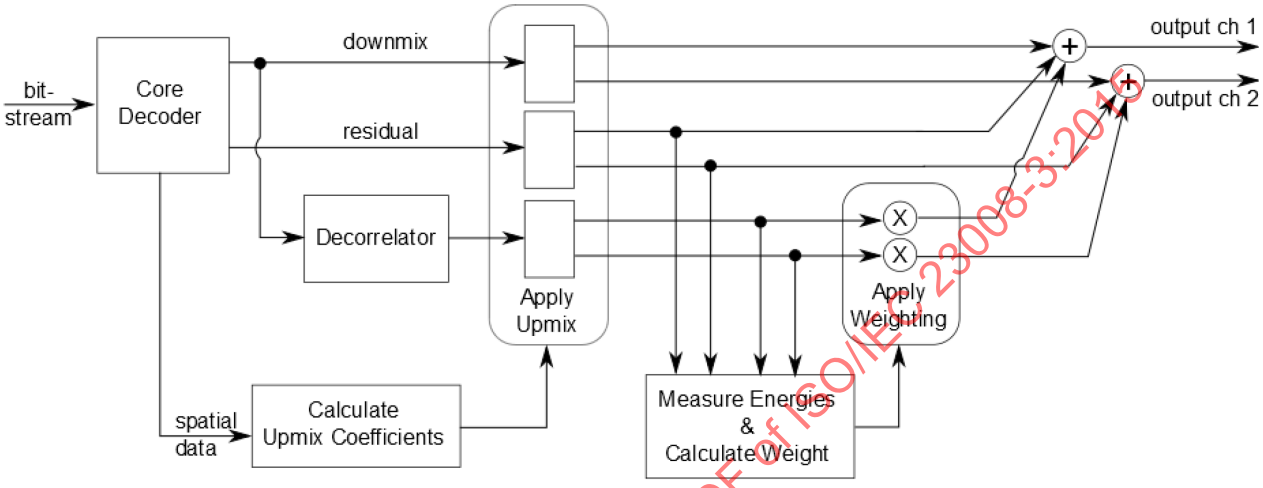



**Figure 11 — Schematics of Hybrid Residual Decoder**

#### 5.5.4.2.2 **Decoding process**

Hybrid Residual Coding mode is indicated by the syntax elements **bsResidualCoding** == 1 and **bsResidualBands** == 1 in Mps212Config(). The calculation of mix-matrix M2 is performed as if **bsResidualCoding** == 0, following the calculation in ISO/IEC 23003-3:2012, subclause 7.11.2.3. The matrix $\mathrm{R}_2^{l,m}$ for the decorrelator based part is defined as

$$\mathrm{R}_2^{l,m} = \begin{bmatrix} H11_{OTT}^{l,m} & H12_{OTT}^{l,m} \\ H21_{OTT}^{l,m} & H22_{OTT}^{l,m} \end{bmatrix}$$

The upmixing process is split up into Downmix, decorrelator output and residual. The upmixed Downmix $\mathrm{u}_{dmx}$ is calculated using:

$$\mathrm{R}_{2,\,dmx}^{l,m} = \begin{bmatrix} H11_{OTT}^{l,m} & 0 \\ H21_{OTT}^{l,m} & 0 \end{bmatrix}$$

The upmixed decorrelator output $\mathrm{u}_{dec}$ is calculated using:

$$\mathrm{R}_{2,\,dec}^{l,m} = \begin{bmatrix} 0 & H12_{OTT}^{l,m} \\ 0 & H22_{OTT}^{l,m} \end{bmatrix}$$

The upmixed residual signal $\mathrm{u}_{res}$ is calculated using:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$R_{2,\,res}^{l,m} = \begin{bmatrix} 0 & H12_{RES}^{l,m} \\ 0 & H22_{RES}^{l,m} \end{bmatrix} = \begin{bmatrix} 0 & \max\{0.5, H11_{OTT}^{l,m}\} \\ 0 & -\max\{0.5, H21_{OTT}^{l,m}\} \end{bmatrix}$$

The energies of the upmixed residual signal $E_{res}$ and of the upmixed decorrelator output $E_{dec}$ are calculated per hybrid band as sum over both output channels ch and all timeslots ts and of one frame as:

$$E_{res} = \sum_{ch}\sum_{ts} \left\| u_{res}(ch, ts) \right\|$$

$$E_{dec} = \sum_{ch}\sum_{ts} \left\| u_{dec}(ch, ts) \right\|$$

The upmixed decorrelator output is weighted using a weighting factor $r_{dec}$ calculated for each hybrid band per frame as:

$$r_{dec} = \begin{cases} 0 & \text{if} \quad E_{res} > E_{dec} \\ 1 & \text{if} \quad E_{res} < \varepsilon \\ \sqrt{\left| \dfrac{E_{dec} - E_{res} + \varepsilon}{E_{dec} + \varepsilon} \right|} & \text{otherwise} \end{cases}$$

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

All three upmix signals are added to form the decoded output signal.

### 5.5.5 Enhanced Noise Filling

#### 5.5.5.1 Introduction

The enhanced noise filling is achieved through a tool named Intelligent Gap Filling (IGF). IGF extends the noise filling in the decoder to alternatively exploit neighboring spectral portions, predefined source tiles, for filling spectral gaps in predefined target tiles. These gaps originate from coarse quantization in the encoder. Using these tiles for gap filling often provides a perceptually better match than just injecting random noise. The additional side information, igf_data(), conveys control data, e.g. tile source and tile target information, tile target level, etc.

#### 5.5.5.2 Data Elements

**indepFlag** the MPEG-H 3D audio independency flag.

**pred_dir** indicates the direction of prediction (see ISO/IEC 23003-3:2012, 7.7.2)

**ms_used[ ][ ]** one-bit flag per scalefactor band indicating that M/S coding or prediction is being used in windowgroup g and scalefactor band sfb, shared with the joint stereo tool (ISO/IEC 23003-3:2012 subclause 7.7).

**cplx_pred_used[g][sfb]** One-bit flag per window group g and scalefactor band sfb (after mapping from prediction bands), shared with the complex stereo prediction tool (see ISO/IEC 23003-3:2012 subclause 7.7.2).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 5.5.5.3 Helper Elements

| | |
|---|---|
| igfStartSfbLB | the IGF start scalefactor band index used with a long window sequence. |
| igfStartSfbSB | the IGF start scalefactor band index used with a short window sequence. |
| igfStopSfbLB | the IGF stop scalefactor band index used with a long window sequence. |
| igfStopSfbSB | the IGF stop scalefactor band index used with a shortwindow sequence. |
| igfMin | is a subband index; this index is sampling frequency and core coder framelength dependent and determine a minimal frequency which is used to assign a source tile range. |
| igfBgn | the IGF start subband; this helper element is used for both, long and short window sequences and is mapped for every frame. |
| igfEnd | the IGF stop subband; this helper element is used for both, long and short window sequences and is mapped for every frame. |
| igfP | is 1 if the flag **igfUseHighRes** is set to one, 2 otherwise. |
| m_igfStartSfb | the mapped IGF start scalefactorband; this helper element is used for both, long and short window sequences and is mapped for every frame. |
| m_igfStopSfb | the mapped IGF stop scalefactorband; this helper element is used for both, long and short window sequences and is mapped for every frame. |
| tile[ ] | vector of length 4 containing width information. |
| igfNTiles | number of target tiles in IGF range. |
| sbs | start subband of the current target tile. |
| rng | length of the IGF range in subbands. |
| ch | channel in scope, this value is either 0 or 1. |
| sfb | scafactor band in scope. |
| num_windows[ch] | contains the current number of windows. |
| num_window_groups[ch] | contains the number of window groups. |
| sb | index of the source subband. |
| tb | index of the target subband. |
| tileIdx | index of the IGF target tile in which a tb under scope is located. |
| group_len | identifies the actual length of a window group. |
| igf_sN[ ] | vector containing information on energies per scalefactor which have been not quantized to zero. |
| igf_pN[ ] | vector containing information on energies per scalefactor which will be copied from the source tile range. |
| wg, wa | counter for the actual window (0,1,…,7). |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| w | the window index in a window group. |
| window_sequence | signals the USAC window sequence. |
| isShortWindow | 1 if window_sequence == EIGHT_SHORT_SEQUENCE, else 0. |
| pMDCT[ ] | array containing the MDCT spectrum. |
| pMDCT_flat[ ] | array containing the whitened MDCT spectrum. |
| width | number of lines in the sfb under scope. |
| E | energy for the current subband. |
| val | spectral values for the current IGF subband. |
| noise_offset | noise-fill offset to modify scale factors of zero-quantized bands (see ISO/IEC 23003-3:2012 subclause 7.2). |
| noise_level | noise-fill level representing amplitude of added spectrum noise (see ISO/IEC 23003-3:2012 subclause 7.2). |
| stereo_filling | flag indicating whether SF is utilized in the current frame and channel. |

#### 5.5.5.4 IGF Signal Processing

##### 5.5.5.4.1 Mapping of IGF bitstream elements

###### 5.5.5.4.1.1 Helper Elements

| | |
|---|---|
| swb_offset_long[ ] | offset table for scalefactor bands to use with long windows. |
| swb_offset_short[ ] | offset table for scalefactor bands to use with short windows. |
| num_swb_short_window | number of scalefactor bands with a short window sequence. |
| num_swb_long_window | number of scalefactor bands with a long window sequence. |
| sampleRate | sampling rate of the core coder. |
| bl | length of the current block in dependency of the window sequence. |

###### 5.5.5.4.1.2 Detailed description

The bit stream elements **igfStartIndex** and **igfStopIndex** are mapped to scale factor band indices:

```
igfStartSfbLB = min(11 + igfStartIndex * 2, num_swb_long_window - 5)
```

If **igfStopIndex** is not 7, calculate:

```
igfStopSfbLB = min( num_swb_long_window, max( igfStartSfbLB + ( ( ( num_swb_long_window-
(igfStartSfbLB +1 ) ) * ( igfStopIndex +1 ) ) >> 3), igfStartSfbLB + 2) )
```

If **igfStopIndex** equals 7, set:

```
igfStopSfbLB = num_swb_long_window
```

For the appropriate start and stop boundaries for IGF with short window sequence calculate:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
igfStartSfbSB = -1;
for (sfb = 0; sfb < num_swb_short_window; sfb++) {
  if (swb_offset_short[sfb] >= swb_offset_long[igfStartSfbLB] >> 3) {
    if (igfStartSfbSB < 0) igfStartSfbSB = sfb;
  }
}
igfStopSfbSB = -1;
for (sfb = 0; sfb < num_swb_short_window; sfb++) {
  if (swb_offset_short[sfb] >= swb_offset_long[igfStopSfbLB] >> 3) {
    if (igfStopSfbSB < 0) igfStopSfbSB = sfb;
  }
}
```

The final mapping to m_igfStartSfb, igfBgn and m_igfStopSfb, igfEnd depends on the window sequence in the actual processed frame.

```
if (window_sequence == EIGHT_SHORT_SEQUENCE) {
  m_igfStartSfb = igfStartSfbSB;
  m_igfStopSfb  = igfStopSfbSB;
  igfBgn = swb_offset_short[m_igfStartSfb];
  igfEnd = swb_offset_short[m_igfStopSfb];
  swb_offset = swb_offset_short;
} else {
  m_igfStartSfb = igfStartSfbLB;
  m_igfStopSfb  = igfStopSfbLB;
  igfBgn = swb_offset_long[m_igfStartSfb];
  igfEnd = swb_offset_long[m_igfStopSfb];
  swb_offset = swb_offset_long;
}
```

If the bitstream element **igfUseHighRes** equals zero, the IGF frequency resolution will be reduced by pairing IGF scalefactor bands.

igfMin is the lowest IGF source subband and it is calculated as follows:

$$igfMin = sb + (sb \bmod 2)$$

where

$$sb = INT(1125 * bl * (2 / sampleRate))$$

The helper element bl is mapped according to Table 59:

**Table 59 — Value of bl**

| window_sequence | bl |
|---|---|
| EIGHT_SHORT_SEQUENCE | ccfl / 8 |
| All other sequences | ccfl |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

5.5.5.4.2 **Computing IGF tiles**

IGF works with so called tiles to determine target regions shown in Figure 12.

© ISO/IEC 2015 – All rights reserved

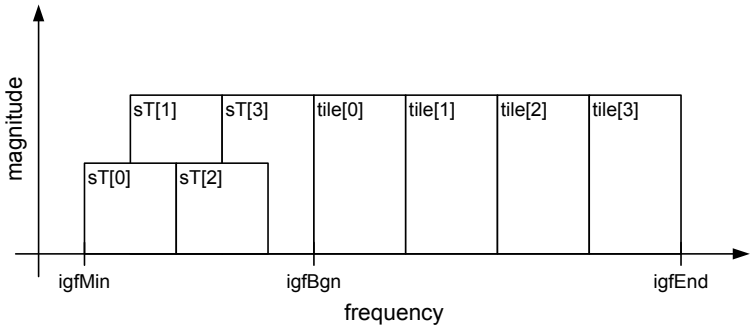



**Figure 12 — Computing target tiles, overview**

The flowchart in Figure 13 produces the tile vector tile[ ] of length 4 containing the width of each IGF target tile. Please note that this flowchart is used for both, short and long window sequences.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

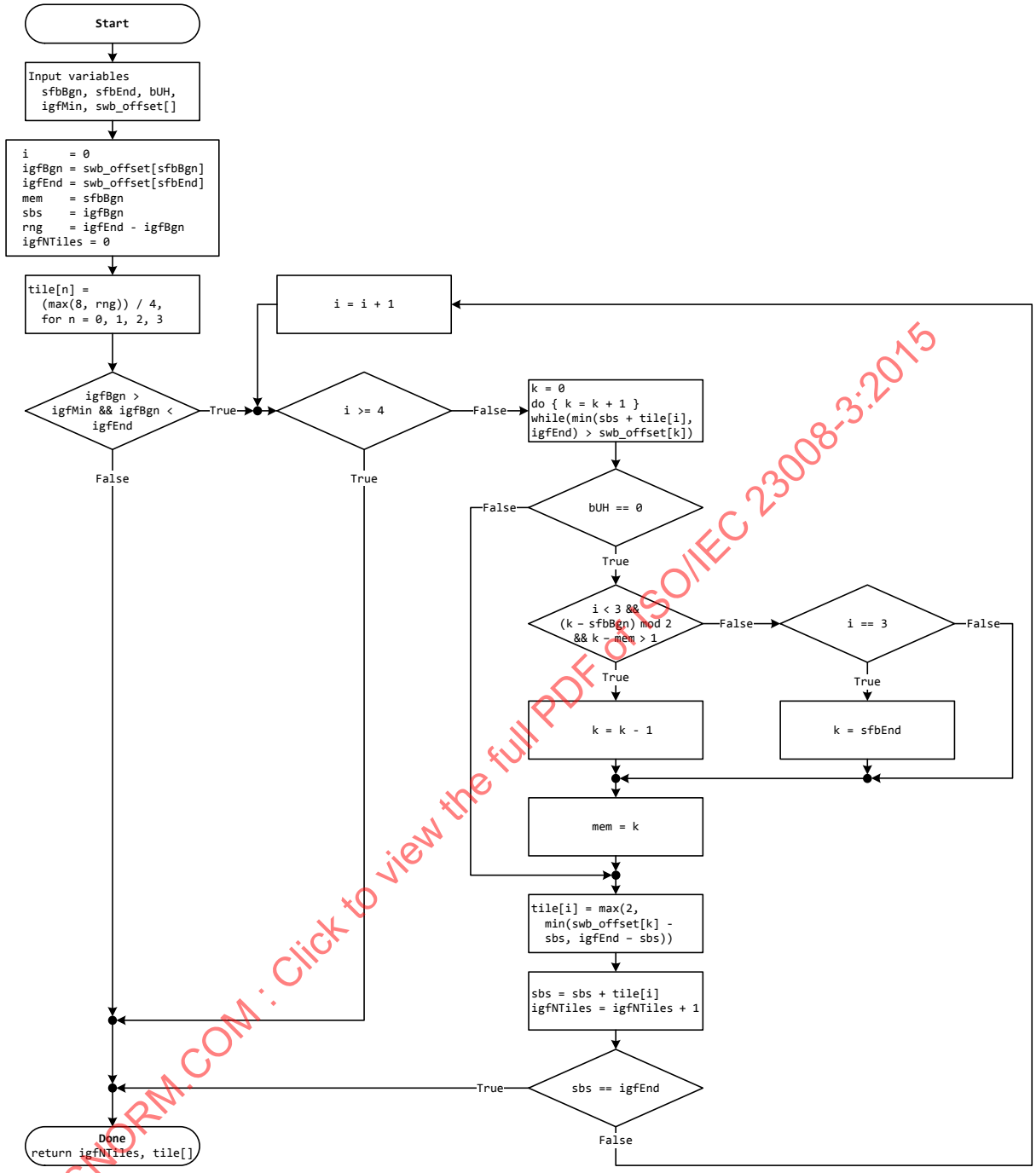



**Figure 13 — Flowchart for calculation of tile width and number of tiles**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

5.5.5.4.3 **Decoding of IGF levels**

#### 5.5.5.4.3.1 Helper Elements

| | |
|---|---|
| igf_curr[ch][ ][ ] | vector containing the IGF levels for the current window. |
| igf_prev[ch][ ] | vector containing the IGF levels for the previous window. |
| igf_arith_t[ch] | time index, to be increment per window since the last reset. |

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| igf_prevD[ch] | context form the window before the previous window. |
| prev_num_windows[ch] | contains the previous number of windows. |

#### 5.5.5.4.3.2 Decoding Process

The subroutine **igf_level()** is performed as outlined below. Please note that all helper elements associated with **igf_level()** needs their own instance of context memory per audio element (CPE, SCE) and per channel ch in case of CPE, for proper decoding.

First of all, the context memory (igf_prev, igf_prevD, igf_arith_t) of the IGF arithmetic coder, together with other helper elements, needs a reset under the following conditions:

```
if (igf_AllZero || indepFlag || (num_windows[ch] != prev_num_windows[ch])) {
  igf_curr[ch] = {0};
  igf_prev[ch] = {0};
  igf_arith_t[ch] = 0;
  igf_prevD[ch] = 0;
  prev_num_windows[ch] = 0;
}
```

If the bitstream element **igf_AllZero** is not true, the subroutine igf_arith_decode() (see 5.5.5.4.4.1) is called as outlined below:

```
if (!igf_AllZero) {
  for (g = 0; g < num_window_groups; g++) {
    igf_curr[ch][g] = igf_arith_decode(igf_arith_t[ch], igf_prev[ch], igf_prevD[ch])
    igf_prevD[ch] = igf_prev[m_igfStartSfb];
    for (sfb = m_igfStartSfb; sfb < m_igfStopSfb; sfb++) {
      igf_prev[ch][sfb] = igf_curr[ch][g][sfb];
      igf_curr[ch][g][sfb] = igf_curr[ch][g][sfb] * igFP;
    }
    igf_arith_t[ch]++;
  }
  arith_decode_flush(); /* push back 14 bits to the bitstream */
}
```

The result, quantized energy information of scalefactor bands in the IGF region, is stored in igf_curr[ ][ ][ ]. For requantization use the formula:

```
igf_curr[ch][g][sfb] = 2^(igf_curr[ch][g][sfb]*0.25)
```

for each channel ch, windowgroup g and scalefactor band sfb in scope.

### 5.5.5.4.4 Arithmetic decoding of IGF average levels

#### 5.5.5.4.4.1 Helper Elements

| | |
|---|---|
| nBits | number of bits to read. |
| nBitRead | decoded number read, of length nBits bits. |
| cfTable[ ] | cumulative frequency table. |
| tableOffset | offset to frequency table. |
| decRes | decoded prection residual. |
| extra | the position of the residual in one tail of the distribution. |
| pred | predicted value computed using the neighbors. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| t | time index since the last reset. |
| prevD | first IGF scf value from the previous frame. |
| ctx | index to the context containing the probability distribution. |

#### 5.5.5.4.4.2 Decoding Process

The IGF scalefactors are encoded by using the function arith_decode() as in ISO/IEC 23003-3:2012 subclause 7.4.3, and by using new probability tables (cf_se01, cf_se02[ ], cf_se10, cf_se20[ ], cf_se11[ ][ ], cf_off_se01, cf_off_se02[ ], cf_off_se10, cf_off_se20[ ] and cf_off_se11[ ][ ]), see Annex A.

The subroutine **igf_level()**, see Table 34, provides a vector containing the IGF average energy information per IGF scalefactor, called SFE, of the MDCT spectral lines for each scale factor band or group of scale factor bands.

We take into account the SFEs from up to previous two frames and the already decoded SFEs from the current frame to derive a context providing the probability distribution for coding. We also employ a fixed linear predictor in each context, based on the same data as used for the quantized context, and decode the prediction residuals instead of the original values. For large prediction residuals, outside of the center of the coding distribution, escape coding is used.

```
arith_decode_bits(nBits)
{
    cf_for_bit[2] = {8192, 0};
    nBitRead = 0;
    for (i = nBits - 1; i >= 0; --i) {
        bit = arith_decode(cf_for_bit, 2);
        nBitRead = nBitRead + (bit << i);
    }
    return nBitRead;
}
```

The helper function arith_decode_bits() uses the USAC arithmetic decoder function arith_decode() to obtain a value of length nBits bits from the bit stream.

```
arith_decode_residual(cfTable, tableOffset)
{
    val = arith_decode(cfTable, 27);
    if ((val != 0) && (val != 26)) {
        decRes = val - 13;
    } else {
        extra = arith_decode_bits(4);
        if (extra == 15) {
            extra = 15 + arith_decode_bits(7);
        }
        if (val == 0) {
            decRes = -13 - extra;
        } else {
            decRes = 13 + extra;
        }
    }
    decRes -= tableOffset;
    return decRes;
}
```

The helper function arith_decode_residual() returns the decoded residual value, which has to be added to the predicted value (pred) to obtain the original value.

```
igf_arith_decode(t, igf_prev, prevD)
{
    igf_prev += igfStartSfb;
    igf_curr = {0};
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
    igf_curr += igfStartSfb;
    igfInc = igfP;
    if (isShortBlock) igfInc = 1;
    for (f = 0; f < m_igfStopSfb - m_igfStartSfb; f += igfInc) {
        if (t == 0) {
            if (f == 0) {
                igf_curr[f] = arith_decode_bits(7);
            } else if (f == igfInc) {
                pred = igf_curr[f - igfInc];
                igf_curr[f] = pred + arith_decode_residual(cf_se01, cf_off_se01);
            } else {
                pred = igf_curr[f - igfInc];
                ctx = quant_ctx(igf_curr[f - igfInc] - igf_curr[f - 2 * igfInc]);
                igf_curr[f] = pred + arith_decode_residual(
                    cf_se02[3 + ctx], cf_off_se02[3 + ctx]);
            }
        } else if (f == 0) {
            if (t == 1) {
                pred = igf_prev[f];
                igf_curr[f] = pred + arith_decode_residual(cf_se10, cf_off_se10);
            } else {
                pred = igf_prev[f];
                ctx = quant_ctx(igf_prev[f] - prevD);
                igf_curr[f] = pred + arith_decode_residual(
                    cf_se20[3 + ctx], cf_off_se20[3 + ctx]);
            }
        } else {
            pred = igf_prev[f] + igf_curr[f - igfInc] - igf_prev[f - igfInc];
            ctx_f = quant_ctx(igf_prev[f] - igf_prev[f - igfInc]);
            ctx_t = quant_ctx(igf_curr[f - igfInc] - igf_prev[f - igfInc]);
            igf_curr[f] = pred + arith_decode_residual(
                cf_se11[3 + ctx_t][3 + ctx_f],
                cf_off_se11[3 + ctx_t][3 + ctx_f]);
        }
        for (z = f + 1; z < min(f + igfInc, m_igfStopSfb - m_igfStartSfb); ++z) {
            igf_curr[z] = igf_curr[f];
        }
    }
    igf_curr -= igfStartSfb;
    return igf_curr;
}
```

The quantization function quant_ctx(), limits large integer values to ±3. It is defined as:

quant_ctx(x) = x; for |x| ≤ 3 and

quant_ctx(x) = 3 · sign(x); for |x| > 3

5.5.5.4.5 **Applying IGF**

**5.5.5.4.5.1 Introduction**

Application of IGF shall occur after the following processing steps:

a) inverse quantization of MDCT values

b) if applicable: noise filling (as specified in 7.2 of ISO/IEC 23003-3), but only up to subband swb_offset[m_igfStartSfb]-1

c) if applicable: Joint Stereo coding (as specified in 7.7 of ISO/IEC 23003-3), but only up to subband swb_offset[max_sfb_ste]-1

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

If the bit-stream element **igfUseEnf** is 1, IGF envelope noise flattening shall be applied in the IGF decoding process.

Please note that the Temporal Noise Shaping (TNS) tool was extended in case of IGF to additionally perform Temporal Tile Shaping (TTS) on IGF tiles. Therefore, the TNS shaping filters are also applied on IGF generated frequency tiles if indicated by **igfAfterTnsSynth**. The following flowchart (Figure 14) shows the position of the IGF tool in dependency of **igfAfterTnsSynth** in the core coder.

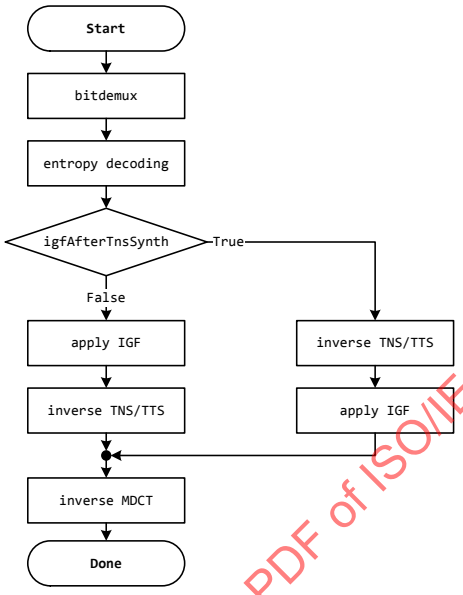



**Figure 14 — Position of IGF in the core coder**

#### 5.5.5.4.5.2 Helper Elements

top TNS stop band, see 14496-3:2009 subclause 4.6.9

#### 5.5.5.4.5.3 Decoding Process

There are two options to apply IGF, signaled with **igfAfterTnsSynth**:

- before TNS synthesis; IGF is applied to the TNS residual, and subsequently TNS/TTS is applied.
- after TNS synthesis; IGF is applied to the fully reconstructed and TNS filtered core coder signal.

Please note that the behavior of TNS changes in case of IGF. The following code sequence adjusts the TNS stop band according to the IGF start and stop scale factor band offset indices:

```
nbands = max_sfb;
if (enhancedNoiseFilling) {
  if (!igfAfterTnsSynth) {
    if (!isShortWindow) {
      if (igfStopSfbLB > nbands) nbands = igfStopSfbLB;
    } else {
      if (igfStopSfbSB > nbands) nbands = igfStopSfbSB;
    }
  } else {
    if (!isShortWindow) {
      if (igfStartSfbLB < nbands) nbands = igfStartSfbLB;
    } else {
      if (igfStartSfbSB < nbands) nbands = igfStartSfbSB;
    }
  }
  top = MIN(top, nbands);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
}
```

#### 5.5.5.4.5.4 Creation of source tile spectra with independent noise filling

If the bit-stream element **igfUseEnf** equals 1, $nT$ copies of the dequantized MDCT core-coder spectrum shall be created and noise filling shall be applied for each copy observing the descriptions and restrictions defined in 5.5.5.4.5 and 5.5.5.4.9. With **igfUseEnf** active the $nT$ MDCT core-coder spectra with different applied noise filling are the dedicated sources for further IGF processing where every `tileIdx` uses one of the created copies.

### 5.5.5.4.6 Computing a source subband in IGF

#### 5.5.5.4.6.1 Helper Elements

| | |
|---|---|
| oS | offset between sb and tb. |
| nST | number of source tiles. |
| src | length of the IGF source range in subbands. |

#### 5.5.5.4.6.2 Detailed description

While applying IGF, target subbands (tb) are identified which have been quantized to zero by the encoder. For each of those target subbands a source subband (sb) is obtained with the flowchart below (Figure 15). Please note that for each element and each channel, there could be a different set of igfCurrTileIdx obtained from the bitstream.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

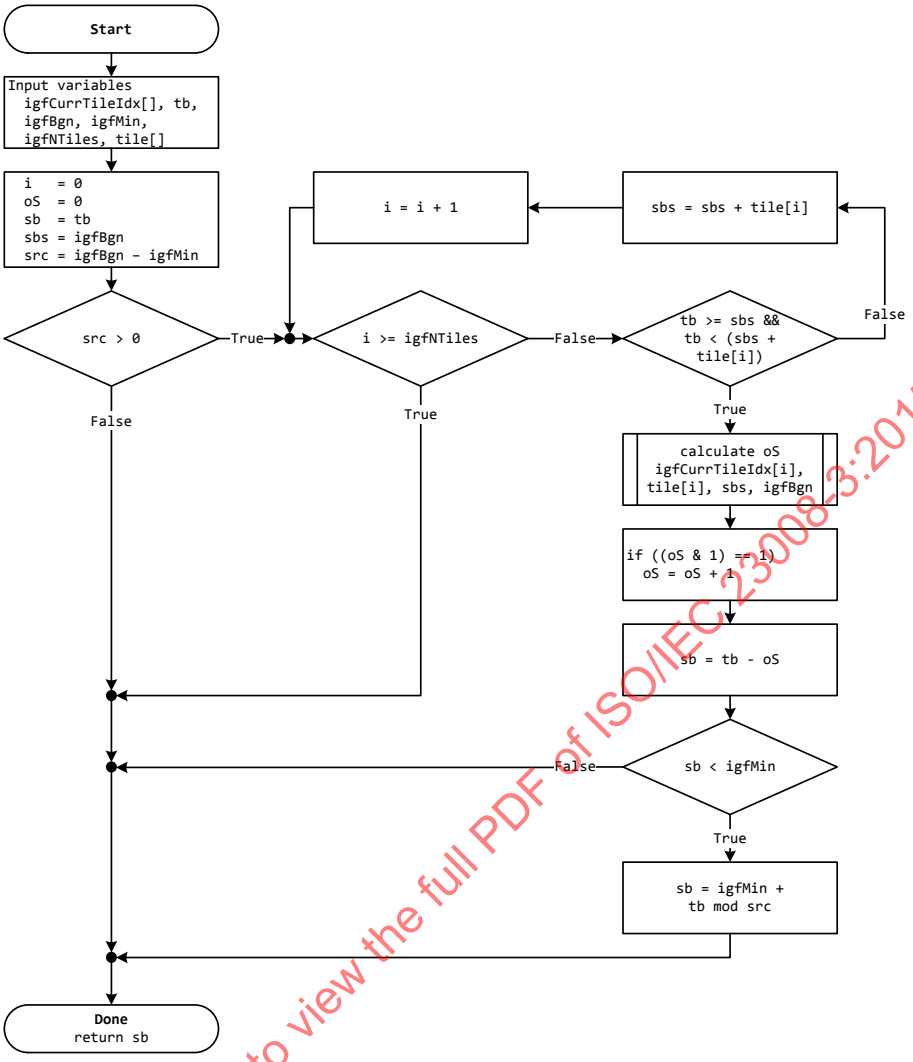



**Figure 15 — Flowchart for computing a source subband sb**

The function “calculate oS()” returns the offset oS between sb and tb using the following formula:

$$oS = INT((-0.5 * \textbf{igfCurrTileIdx[ch][i]} + 2.5) * tile[i] + sbs - igfBgn)$$

where tile[ ] is a vector with a maximum length of 4 containing the width of each IGF target tile according to 5.5.5.4.2.

The helper function get_IGF_tile_idx() identifies the tile where tb is located.

```
get_IGF_tile_idx(tb, igfBgn, igfNTiles, tile[]) {
  sbs = igfBgn;
  for (tileIdx = 0; tileIdx < igfNTiles; tileIdx ++) {
    sbs += tile[tileIdx];
    if (tb < sbs) break;
  }
  return tileIdx;
}
```

#### 5.5.5.4.6.3 Requirements

The computing of a source subband sb is constrained to the following requirement for the transmitted **igfCurrTileIdx**.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**igfCurrTileIdx[ch][i]** + nST[i] >= 4, for i = 0, 1, .., igfNTiles

The vector nST shall be computed as follows:

nST[i] = max( 1, min( (src / (tile[i] / 2)) – 1, 4 ) ), for i = 0, 1, ..., igfNTiles

5.5.5.4.7 **Spectral whitening in IGF**

To reduce tonality or spectral tilt of an IGF tile, the following procedure uses the MDCT signal, if the bitstream element **igfUseWhitening** is equal to 1, to produce the whitened spectrum pMDCT_flat[ ], which is further used in IGF_mono() and IGF_stereo().

```
igf_apply_whitening(pMDCT[],pMDCT_flat[])
{
  stop = swb_offset[m_igfStartSfb];
  for (i = igfMin -3; i < stop; i++) {
    power_spec[i] = pMDCT[i] * pMDCT[i];
  }

  for (i = igfMin; i < stop-3; i++) {
    env[i] = 1e-3;
    for (j = -3; j <= 3; j++) {
      env[i] += power_spec[i+j];
    }
  }

  for (i = stop-3; i < stop; i++) {
    env[i] = env[stop-4];
  }

  for (i = igfMin; i < stop; i++) {
    n = INT(log(env[i])/log(2));
    pMDCT_flat[i] = pMDCT[i] * pow(2, 21 - 0.5 * n);
  }
}
```

5.5.5.4.8 **Single/Dual channel processing in IGF**

**5.5.5.4.8.1 Helper Elements**

randomSign() function returning (pseudo) random sign as defined in ISO/IEC 23003-3:2012 subclause 7.2.4 using latest seed state from USAC noise filling.

**5.5.5.4.8.2 Decoding Process**

If IGF is applied in the discrete channel mode, e.g. in a SCE element or in a CPE element which does not make use of joint stereo feature (**igfIndependentTiling** is true) the following calling sequence shall be applied:

```
IGF_mono(ch, num_window_groups, group_len) {
  if (!isShortWindow && igfUseWhitening) {
    igf_apply_whitening (pMDCT, pMDCT_flat);
  }
  wg = wa = 0;
  for (g = 0; g < num_window_groups; g++) {
    igf_sN[ch] = {0};
    igf_pN[ch] = {0};
    for (w = 0; w < group_len[g]; w++) {
       IGF_calc_mono(ch, wg, group_len[g]);
       wg++;
    }
    for (w = 0; w < group_len[g]; w++) {
       IGF_apply_mono(ch, wa);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
        wa++;
      }
    }
  }
```

IGF_calc_mono() is used to compute the vectors igf_sN[ ] and igf_pN[ ] respectively. The subroutine get_IGF_sb() is specified in 5.5.5.4.6. Please note that igfCurrTileIdx[ch] is a vector of length 4, describing the current tile indices of the current channel ch, where ch is 0 in a SCE and ch is 0 or 1 in a CPE.

```
IGF_calc_mono(ch, w, group_len) {
  igfInc = igfP;
  if (isShortBlock) igfInc = 1;
  for (sfb = m_igfStartSfb; sfb < m_igfStopSfb; sfb += igfInc) {
    width = (swb_offset[MIN(sfb + igfInc, m_igfStopSfb)] - swb_offset[sfb]);
    E = 1.12E-38;
    for (bin = 0; bin < width; bin++) {
      tb = swb_offset[sfb]+bin;
      E += pMDCT[w][tb] * pMDCT[w][tb];
    }
    igf_sN[ch][sfb] += E/group_len;
    E = 1.12E-38;
    for (bin = 0; bin < width; bin++) {
      tb = swb_offset[sfb]+bin;
      if (pMDCT[w][tb] == 0) {
        sb = get_IGF_sb(igfCurrTileIdx[ch], tb);
        val = pMDCT[w][sb];
        if (!isShortWindow && igfUseWhitening) {
          tileIdx = get_IGF_tile_idx(tb);
          if (igf_WhiteningLevel[tileIdx] == 0) {
            val = pMDCT_flat[w][sb];
          }
          if (igf_WhiteningLevel[tileIdx] == 2) {
            val = randomSign()*pow(2,21);
          }
        }
        E += val * val;
      }
    }
    igf_pN[ch][sfb] += E/group_len;
  }
}
```

IGF_apply_mono() will fill spectral gaps with previous calculated values:

```
IGF_apply_mono(ch, w) {

  igfInc = igfP;
  if (isShortBlock) igfInc = 1;
  for (sfb = m_igfStartSfb; sfb < m_igfStopSfb; sfb+= igfInc) {
    width = (swb_offset[MIN(sfb + igfInc, m_igfStopSfb)] - swb_offset[sfb]);

    dE = igf_curr[sfb];
    sN = igf_sN[ch][sfb];
    pN = igf_pN[ch][sfb];
    mN = (dE*dE)*width -sN;

    if (mN > 0 && pN > 0) {
      gn = min(10, sqrt(mN/pN));
    } else {
      gn = 0;
    }
    for (bin = 0; bin < width; bin++) {
      tb = swb_offset[sfb]+bin;
      if (pMDCT[w][tb] == 0) {
        sb = get_IGF_sb(igfCurrTileIdx[ch], tb);
        val = pMDCT[w][sb];
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
        if (!isShortWindow && igfUseWhitening) {
          tileIdx = get_IGF_tile_idx(tb);
          if (igf_WhiteningLevel[tileIdx] == 0) {
            val = pMDCT_flat[w][sb];
          }
          if (igf_WhiteningLevel[tileIdx] == 2) {
            val = randomSign()*pow(2,21);
          }
        }
        pMDCT[w][tb] = gn * val;
      }
    }
  }
}
```

5.5.5.4.9 **Stereo Filling in IGF**

#### 5.5.5.4.9.1 Introduction

When stereo filling (SF) is active in a FD-only CPE, the MDCT coefficients of empty (i.e. fully zero-quantized) scale factor bands of the right (second) channel are replaced by a sum or difference of the corresponding decoded left and right channels MDCT coefficients of the previous frame. If USAC noise filling is active for the second channel, pseudo-random values are also added to each coefficient. The resulting coefficients of each scale factor band are then scaled such that the RMS (root of the mean coefficient square) of each band matches the value transmitted by way of that band's scale factor. See in ISO/IEC 23003-3:2012 subclause 7.3.

#### 5.5.5.4.9.2 Helper Elements

| | |
|---|---|
| downmix_prev[ ][ ] | downmix (i.e. sum or difference) of the previous frame's left & right channels. |
| noiseFillingStartOffset | line index at or above which noise filling is used (ISO/IEC 23003-3:2012, 7.2). |
| sfbWidth[ ] | array containing the number of lines per sfb. |
| energy[ ] | energy of the signal per sfb. |
| energy_dmx[ ] | energy of the downmix signal per sfb. |
| spectrum[ ][ ] | MDCT spectrum for group g after noise filling (i.e. x_ac_invquant[g][ ][sfb][ ]). |
| window | indices of windows of group in scope, i.e. all windows belonging to group g. |

#### 5.5.5.4.9.3 Operational Constraints

SF can only be used in the right FD channel of a common FD channel pair element (CPE), i.e. a channel pair element transmitting a StereoCoreToolInfo() with **common_window** == 1. Besides, due to its signaling, SF can only be applied when **noiseFilling** == 1 in mpegh3daCoreConfig(). If either of the channels in the pair is in LPD **core_mode**, SF is not used, even if the right channel is in FD mode.

#### 5.5.5.4.9.4 Decoding Process

The decoding of a joint-stereo coded FD channel with SF is executed in 3 sequential steps as follows.

**Step 1:** Decoding of stereo_filling

stereo_filling does not represent an independent bit-stream element but is derived from the noise-fill elements, noise_offset and noise_level, in a mpegh3daChannelPairElement() and the **common_window** flag in StereoCoreToolInfo(). If **noiseFilling** == 0 or **common_window** == 0 or the current channel is the left (first) channel in the element, stereo_filling is 0, and the stereo filling process ends. Otherwise,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
if ((noiseFilling != 0) &&
    (noise_level == 0) &&
    (noise_offset != 0) &&
    (common_window != 0)) {
  stereo_filling = 1;
  noise_level    = (noise_offset & 28) / 4;
  noise_offset   = (noise_offset & 3) * 8;
}
else {
  stereo_filling = 0;
}
```

In other words, if stereo_filling == 1, **noise_offset** contains 5 bits of noise filling data, which are then rearranged. Since this operation alters the values of **noise_level** and **noise_offset**, it must be performed before the noise filling process as in ISO/IEC 23003-3:2012 subclause 7.2. Moreover, the above pseudo-code is not executed in the left channel of a mpegh3daChannelPairElement() or any other element.

**Step 2:** Calculation of downmix_prev

downmix_prev[], the spectral mix which is to be used for stereo filling, is identical to the dmx_re_prev[] used for the MDST spectrum estimation in complex stereo prediction (see ISO/IEC 23003-3:2012 subclause 7.7.2.3). This means that:

- All coefficients of downmix_prev[ ] must be zero if any of the channels of the frame and element with which the downmixing is performed – i.e. the frame before the currently decoded one – use **core_mode** == 1 (LPD) or if the channels use unequal transform lengths (**split_transform** == 1 or block switching to **window_sequence** == EIGHT_SHORT_SEQUENCE in only one channel).

- All coefficients of downmix_prev[ ] must be zero during the stereo filling process if the channel's transform length changed from the last to the current frame (i.e. **split_transform** == 1 preceded by **split_transform** == 0, or **window_sequence** == EIGHT_SHORT_SEQUENCE preceded by **window_sequence** != EIGHT_SHORT_SEQUENCE, or vice versa resp.) in the current element or usacIndependencyFlag == 1.

- If transform splitting is applied in the channels of the previous or current frame, downmix_prev[ ] represents a line-by-line interleaved spectral downmix. See the transform splitting tool for details.

- If complex stereo prediction is not utilized in the current frame and element, pred_dir equals 0.

Consequently, the previous downmix only has to be computed once for both tools, saving complexity. The only difference between downmix_prev[ ] and dmx_re_prev[ ] in ISO/IEC 23003-3:2012 subclause 7.7.2 is the behavior when complex stereo prediction is not currently used, or when it is active but **use_prev_frame** == 0. In that case, downmix_prev[ ] is computed for stereo filling decoding according to ISO/IEC 23003-3:2012 subclause 7.7.2.3 even though dmx_re_prev[] is not needed for complex stereo prediction decoding and is, therefore, undefined/zero.

**Step 3:** Stereo Filling of Empty Scale Factor Bands

If stereo_filling == 1, the following procedure is carried out after the noise filling process in all initially empty scale factor bands sfb at or above noiseFillingStartOffset (see FD noise filling) and below max_sfb_ste, i.e. all bands in which all MDCT lines were quantized to zero. First, the energies energy[ ] and energy_dmx[ ] of the given sfb and the corresponding lines in downmix_prev[], respectively, are computed via sums of the squares:

```
energy_dmx[sfb] = 1e-8;
energy[sfb] = 0;
for (index = swb_offset[sfb]; index < swb_offset[sfb+1]; index++) {
  energy_dmx[sfb] += downmix_prev[window][index] * downmix_prev[window][index];
  spectrum[window][index] *= 4;
  energy[sfb] += spectrum[window][index] * spectrum[window][index];
}
```

Then, given sfbWidth[ ] containing the number of lines per sfb,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
if (energy[sfb] < sfbWidth[sfb]) {
  tmp = sqrt((sfbWidth[sfb] - energy[sfb]) / energy_dmx[sfb]);
  factor = 0;
  for (index = swb_offset[sfb]; index < swb_offset[sfb+1]; index++) {
    spectrum[window][index] += downmix_prev[window][index] * tmp;
    factor += spectrum[window][index] * spectrum[window][index];
  }
  if ((factor != sfbWidth[sfb]) && (factor > 0)) {
    factor = sqrt(sfbWidth[sfb] / (factor + 1e-8));
    for (index = swb_offset[sfb]; index < swb_offset[sfb+1]; index++) {
      spectrum[window][index] *= factor;
    }
  }
}
```

for the spectrum of each group window. Then the scale factors are applied on the resulting spectrum as in 7.3 of ISO/IEC 23003-3:2012, with the scale factors of the empty bands being processed like regular scale factors.

5.5.5.4.10 **MS and Complex Prediction processing in IGF**

#### 5.5.5.4.10.1 Introduction

IGF stereo coding is applied to CPEs where **igfIndependentTiling** is zero and then replaces the channel-wise decoding in 5.5.5.4.8.

The IGF joint stereo tool is based on the joint stereo tool, the decoding after filling is done the same way, in comparison to independent patching the residuals are transformed from the already decoded left/right values to the appropriate mid/side values for bands where joint stereo is active.

#### 5.5.5.4.10.2 Helper Elements

| | |
|---|---|
| l_spec[ ] | Array containing the left channel spectrum of the respective channel pair. |
| r_spec[ ] | Array containing the right channel spectrum of the respective channel pair. |
| l_pMDCT_flat[ ] | Array containing the left channel whitened spectrum. |
| r_pMDCT_flat[ ] | Array containing the right channel whitend spectrum. |
| l_E, r_E | Energies for the current subband, for the left and right channel respectively. |
| l_igf_sN[ ], r_igf_sN[ ] | vectors containing information on energies per scalefactor which have been not quantized to zero, for the left and right channel respectively. |
| l_igf_pN[ ], r_igf_pN[ ] | vector containing information on energies per scalefactor which will be copied from the source tile range, for the left and right channel respectively. |
| l_sb, r_sb | current IGF source subbands, for the left and right channel respectively. |
| l_val, r_val | left and right spectral values for the current IGF subband. |
| sfb_per_ms_band | Number of scalefactor bands per M/S band, dependent on igfUseHighRes. |
| SFB_PER_PRED_BAND | Number of scalefactor bands per complex prediction band, equal to 2. |
| dpcm_alpha_q_re[ ][ ] | Differentially coded real part of prediction coefficient of group g, scalefactor band sfb. |
| alpha_q[ ][ ] | real or imaginary parts of prediction coefficients. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 5.5.5.4.10.3 Decoding Process

The decoding of MS and Complex Prediction is divided into 3 parts as follows.

**Part 1:** Decoding of prediction coefficients

Equal to the decoding of the complex prediction coefficients, but only for the real part as in ISO/IEC 23003-3:2012 subclause 7.7.2.3.2.

**Part 2:** Inverse quantization of prediction coefficients

Equal to the inverse quantization of the complex prediction coefficients as in ISO/IEC 23003-3:2012 subclause 7.7.2.3.3.

**Part 3:** IGF apply joint stereo process

If IGF is used in a joint stereo manner (**igfIndependentTiling** is false) the following calling sequnce shall be applied:

```
IGF_stereo(num_window_groups, num_windows, group_len) {
  If (!isShortWindow && igfUseWhitening) {
    igf_apply_whitening (l_spec, l_pMDCT_flat);
    igf_apply_whitening (r_spec, r_pMDCT_flat);
  }

  wg = wa = 0;
  for (g = 0; g < num_window_groups; g++) {
    l_igf_sN = {0};
    r_igf_sN = {0};
    l_igf_pN = {0};
    r_igf_pN = {0};
    for (w = 0; w < group_len[g]; w++) {
       IGF_calc_stereo(wg, group_len[g]);
       wg++;
    }
    for (w = 0; w < group_len[g]; w++) {
       IGF_apply_stereo(wa);
       wa++;
    }
  }
}
```

IGF_calc_stereo() is used to compute the value sets igf_sN and igf_pN respectively. The subroutine get_IGF_sb() is specified in 5.5.5.4.6. Please note that igfCurrTileIdx[0] is a vector of length 4, describing the current tile indices of channel 0, and igfCurrTileIdx[1] is a vector of length 4 too, describing the current tile indices of channel 1 of the actual CPE element.

```
IGF_calc_stereo(ch, w, group_len) {
  igfInc = igfP;
  if (isShortBlock) igfInc = 1;
  for (sfb = m_igfStartSfb; sfb < m_igfStopSfb; sfb+=igfInc) {
    width = (swb_offset[MIN(sfb + igfInc, m_igfStopSfb)] - swb_offset[sfb]);

    r_E = 1.12E-38; l_E = 1.12E-38;

    for (bin = 0; bin < width; bin++) {
      tb = swb_offset[sfb]+bin;
      l_E   += l_spec[w][tb] * l_spec[w][tb];
      r_E   += r_spec[w][tb] * r_spec[w][tb];
    }
    l_igf_sN[sfb] += l_E/group_len;
    r_igf_sN[sfb] += r_E/group_len;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
    r_E = 1.12E-38; l_E = 1.12E-38;
    for (bin = 0 ; bin < width ; bin++) {
      tb = swb_offset[sfb]+bin;
      l_sb = get_IGF_sb(igfCurrTileIdx[0], tb);
      r_sb = get_IGF_sb(igfCurrTileIdx[1], tb);
      l_val = l_spec[w][l_sb];
      r_val = r_spec[w][r_sb];
      if (!isShortWindow && igfUseWhitening) {
        l_tileIdx = get_IGF_tile_idx(tb);
        if (l_igf_WhiteningLevel[l_tileIdx] == 0) {
          l_val = l_pMDCT_flat[w][l_sb];
        }
        if (l_igf_WhiteningLevel[l_tileIdx] == 2) {
          l_val = randomSign()*pow(2,21);
        }
        r_tileIdx = get_IGF_tile_idx(tb);
        if (r_igf_WhiteningLevel[r_tileIdx] == 0) {
          r_val = r_pMDCT_flat[w][r_sb];
        }
        if (r_igf_WhiteningLevel[r_tileIdx] == 2) {
          r_val = randomSign()*pow(2,21);
        }
      }
      if (ms_used[g][sfb] || cplx_pred_used[g][sfb]) {
        tmp = l_val;
        l_val = 0.5 * (tmp + r_val);
        r_val = 0.5 * (tmp - r_val);
      }
      if (l_spec[w][tb] == 0) {
        l_E += l_val * l_val;
      }
      if (r_spec[w][tb] == 0) {
        r_E += r_val * r_val;
      }
    }
    l_igf_pN[sfb] += l_E/group_len;
    r_igf_pN[sfb] += r_E/group_len;
  }
}
```

IGF_apply_stereo will fill spectral gaps with previous calculated values:

```
IGF_apply_stereo(w) {
  igfInc = igfP;
  if (isShortBlock) igfInc = 1;
  for (sfb = m_igfStartSfb; sfb < m_igfStopSfb; sfb += igfInc) {
    width = (swb_offset[MIN(sfb + igfInc, m_igfStopSfb)] - swb_offset[sfb]);

    l_dE = l_igf_curr[sfb];
    r_dE = r_igf_curr[sfb];
    l_sN = l_igf_sN[sfb];
    r_sN = r_igf_sN[sfb];
    l_pN = l_igf_pN[ch][sfb];
    r_pN = r_igf_pN[ch][sfb];
    l_mN = (l_dE*l_dE) * width - l_sN;
    r_mN = (r_dE*r_dE) * width - r_sN;


    if (l_mN > 0 && l_pN > 0) {
      l_gn = min(10, sqrt(l_mN/l_pN));
    } else {
      l_gn = 0;
    }
    if (r_mN > 0 && r_pN > 0) {
      r_gn = min(10, sqrt(r_mN/r_pN));
    } else {
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
      r_gn = 0;
    }
    for (bin = 0; bin < width; bin++) {
      tb = swb_offset[sfb]+bin;
      l_sb= get_IGF_sb(igfCurrTileIdx[0], tb);
      r_sb= get_IGF_sb(igfCurrTileIdx[1], tb);
      l_val = l_spec[w][l_sb];
      r_val = r_spec[w][r_sb];
      if (!isShortWindow && igfUseWhitening) {
        l_tileIdx = get_IGF_tile_idx(tb);
        if (l_igf_WhiteningLevel[l_tileIdx] == 0) {
          l_val = l_pMDCT_flat[w][l_sb];
        }
        if (l_igf_WhiteningLevel[l_tileIdx] == 2) {
          l_val = randomSign()*pow(2,21);
        }
        r_tileIdx = get_IGF_tile_idx(tb);
        if (r_igf_WhiteningLevel[r_tileIdx] == 0) {
          r_val = r_pMDCT_flat[w][r_sb];
        }
        if (r_igf_WhiteningLevel[r_tileIdx] == 2) {
          r_val = randomSign()*pow(2,21);
        }
      }
      if (ms_used[g][sfb] || cplx_pred_used[g][sfb]) {
        tmp = l_val;
        l_val = 0.5 * (tmp + r_val);
        r_val = 0.5 * (tmp - r_val);
      }
      if (l_spec[w][tb] == 0) {
        l_spec[w][tb] = l_gn * l_val;
      }
      if (r_spec[w][tb] == 0) {
        r_spec[w][tb] = r_gn * r_val;
      }
    }
  }
}
```

After the filling process, the upmix is now applied from band subband swb_offset[m_igfStartSfb] to subband swb_offset[m_igfStopSfb] as specified in 7.7 of ISO/IEC 23003-3:2012, replacing pred_dir by **igf_pred_dir** if prediction is active. Note that since only real prediction is used, no MDST has to be generated for these bands.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

#### 5.5.5.4.11 **IGF-ENF Processing**

If **isIgfEnfFlat** is 1, the reconstructed signal by IGF is temporally flattened in the frequency domain. The temporal flattening is performed in a frequency-selective manner: The selection of the spectral contents to be temporally flattened is done by comparing the quantized MDCT coefficients with 0 and the contents whose coefficients are quantized to 0 are selected.

In order to maintain the significant MDCT contents, they are temporarily replaced by the MDCT coefficients which are similarly generated to the filled coefficients by IGF:

$$pMDCT_{temp}[w][tb] = \begin{cases} pMDCT[w][tb] & ,\text{if } pMDCT_{before}[w][tb] = 0 \\ gn \cdot val & ,\text{if } abs(pMDCT_{before}[w][tb]) > 0 \end{cases}'$$

where $pMDCT_{before}[w][tb]$ are the quantized MDCT coefficients, $pMDCT[w][tb]$ are the MDCT coefficients after applying IGF, and *gn* and *val* are obtained by the same manner as defined in IGF_apply_mono() and IGF_apply_stereo(), see 5.5.5.4.8.

© ISO/IEC 2015 – All rights reserved

The linear prediction of the MDCT coefficients $pMDCT_{temp}[w][tb]$ is done and the linear prediction coefficients $a_{igfTemp}(m),\ m = 1, 2, \ldots, 8$ are calculated. Then the temporally flattened MDCT coefficients are given by the following filtering:

$$pMDCT_{temp}'[w][tb] = pMDCT_{temp}[w][tb] + \sum_{m=1}^{M} a_{igfTemp}(m) \cdot pMDCT_{temp}[w][tb-m] .$$

Finally, the significant MDCT contents are restored by:

$$pMDCT_{temp}'[w][tb] = \begin{cases} pMDCT_{temp}'[w][tb] & , \text{if } pMDCT_{before}[w][tb] = 0 \\ pMDCT[w][tb] & , \text{if } abs(pMDCT_{before}[w][tb]) > 0 \end{cases},$$

and then the frequency-selectively temporally flattened MDCT coefficients $pMDCT_{temp}'[w][tb]$ are output to IMDCT for getting the time domain signal.

### 5.5.6 Audio Pre-Roll

#### 5.5.6.1 General

The *AudioPreRoll()* syntax element is used to transmit audio information of previous frames along with the data of the present frame. The additional audio data can be used to compensate the decoder startup delay (pre-roll), thus enabling random access at stream access points (SAP) that make use of *AudioPreRoll()*.

A *mpegh3daExtElement()* with the usacExtElementType of ID_EXT_ELE_AUDIOPREROLL shall be used to transmit the *AudioPreRoll()*.

#### 5.5.6.2 Semantics

**configLen** Size of the configuration syntax element in bytes.

Config() The decoder configuration syntax element. In the context of this standard this shall be the Mpegh3daConfig() as defined in 5.2.2.1. The Config() field may be transmitted to be able to respond to changes in the audio configuration (e.g. switching of streams).

**numPreRollFrames** The number of pre-roll access units (AUs) transmitted as audio pre-roll data. The reasonable number of AUs depends on the decoder start-up delay.

**auLen** AU length in bytes.

AccessUnit() The pre-roll AU(s).

NOTE The pre-roll data carried in the extension element may be transmitted “out of band”, i. e. the buffer requirements may not be satisfied

In order to use *AudioPreRoll()* for both random access and bitrate adaptation the following restrictions apply:

— The first element of every frame shall be an extension element (mpegh3daExtElement) of type ID_EXT_ELE_AUDIOPREROLL.

— The corresponding mpegh3daExtElement() shall be configured as specified in Table 60.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

— Consequently, if pre-roll data is present, this mpegh3daFrame() shall start with the following bit sequence:

  — "1": usacIndependencyFlag.

  — "1": usacExtElementPresent (referring to audio pre-roll extension element).

  — "0": usacExtElementUseDefaultLength (referring to audio pre-roll extension element).

— If no *AudioPreRoll()* is transmitted, the extension payload shall not be present (usacExtElementPresent = 0).

— The pre-roll frames with index "0" shall be independently decodable, i.e. *usacIndependencyFlag* shall be set to "1".

  — If an extension element (mpegh3daExtElement) of type ID_EXT_ELE_SAOC_3D is present, in the pre-roll frames with index "0" bsIndependencyFlag shall be set to "1" in the extension payload Saoc3DFrame().

**Table 60 — Setup of mpegh3daExtElementConfig() for AudioPreRoll()**

| Bitstream Field | Value |
|---|---|
| usacExtElementType | ID_EXT_ELE_AUDIOPREROLL |
| usacExtElementConfigLength | 0 |
| usacExtElementDefaultLengthPresent | 0 |
| usacExtElementPayloadFrag | 0 |

### 5.5.6.3 Decoding Process

#### 5.5.6.3.1 Introduction

This section describes the decoding process for both random access / immediate play-out and bitrate adoption scenarios.

#### 5.5.6.3.2 Random access and immediate play-out

Random access and immediate play-out is possible at every frame that utilizes the *AudioPreRoll*() structure as specified in this subclause. The following pseudo-code describes the decoding process:

```
if(usacIndependencyFlag == 1){
  if(usacExtElementPresent == 1){

    /* In this case usacExtElementUseDefaultLength must be 0! */
    if(usacExtElementUseDefaultLength != 0) goto error;

    /* Not used */
    getmpegh3daExtElementPayloadLength();

    /* Check for presence of config and re-initialize if necessary */
    int configLen = getConfigLen();
    if(configLen > 0){
      config c = getConfig(configLen);
      ReConfigureDecoder(c);
      if (ID_EXT_ELE_SAOC_3D) { /*if SAOC 3D payload is present in the bitstream */
        config saoc3dConfig = getSaoc3dConfig(c);
        ReConfigureSaoc3dDecoder(saoc3dConfig);
      }
    }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
    /* Get pre-roll AUs and decode, discard output samples */
    int numPreRollFrames = getNumPreRollFrames();
    for(auIdx = 0; auIdx < numPreRollFrames; auIdx++)
      int auLen = getAuLen();
      AU nextAU = getPreRollAU(auLen);
      if (ID_EXT_ELE_SAOC_3D) { /*if SAOC 3D payload is present in the bitstream */
        frame nextSaoc3dFrame = Saoc3DFrame(nextAU);
        [outSamplesFrame, saocInSamplesFrame] = DecodeAU(nextAU);
        saocOutSamplesFrame = Saoc3dDecode(saocInSamplesFrame, nextSaoc3dFrame);
        /* outSamplesFrame and saocOutSamplesFrame are discarded */
      } else {
        DecodeAU(nextAU);
      }
    }
  }
}
/* Internal decoder states are initialized at this point. Continue normal decoding */
```

5.5.6.3.3 **Bitrate adaption**

Bitrate adaption may be utilized by switching between different encoded representations of the same audio content. The *AudioPreRoll*() structure as specified in this subclause may be used for that purpose. The decoding process in case of bitrate adaption is specified by the following pseudo-code:

```
if(usacIndependencyFlag == 1){
  if(usacExtElementPresent == 1{

    /* In this case usacExtElementUseDefaultLength must be 0! */
    if(usacExtElementUseDefaultLength != 0) goto error;

    /* Not used */
    getmpegh3daExtElementPayloadLength();

    int configLen = getConfigLen();
    if(configLen > 0){
      config newConfig = getConfig(configLen);

      /* Configuration did not change, skip AudioPreRoll and continue decoding as normal
*/
      if(newConfig == currentConfig){
        SkipAudioPreRoll();
        goto finish;
      }

      /* Configuration changed, prepare for bitstream switching*/
      outSamplesFlush = FlushDecoder();
      ReConfigureDecoder(c);

      if (ID_EXT_ELE_SAOC_3D) { /*if SAOC 3D payload is present in the bitstream */
        int differentSaoc3dStream = 0;
        config newSaoc3dConfig = getSaoc3dConfig(newConfig);
        if (newSaoc3dConfig != currentSaoc3dConfig) {
          differentSaoc3dStream = 1;
          ReConfigureSaoc3dDecoder(newSaoc3dConfig);
        }
      }

      /* Get pre-roll AUs and decode, discard output samples */
      int numPreRollFrames = getNumPreRollFrames();
      for(auIdx = 0; auIdx < numPreRollFrames; auIdx++)
        int auLen = getAuLen();
        AU nextAU = getPreRollAU(auLen);
        /*if SAOC 3D payload is present in the bitstream */
        if (ID_EXT_ELE_SAOC_3D) && (differentSaoc3dStream == 1) {
          frame nextSaoc3dFrame = Saoc3DFrame();
          [outSamplesFrame, saocInSamplesFrame] = DecodeAU(nextAU);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
            saocOutSamplesFrame = Saoc3dDecode(saocInSamplesFrame, nextSaoc3dFrame);
            /* outSamplesFrame and saocOutSamplesFrame are discarded */
          } else {
            DecodeAU(nextAU);
          }
        }

        /* Get "regular" AU and decode */
        AU au = mpegh3daFrame();
        /*if SAOC 3D payload is present in the bitstream */
        if (ID_EXT_ELE_SAOC_3D) && (differentSaoc3dStream == 1) {
          frame Saoc3dFrame = Saoc3DFrame();
          [outSamplesFrame, saocInSamplesFrame] = DecodeAU(au);
          saocOutSamplesFrame = Saoc3dDecode(saocInSamplesFrame, Saoc3dFrame);
        } else {
          outSamplesFrame = DecodeAU(au);
        }

        /* Apply crossfade only on the non-SAOC output samples*/
        for(i = 0; i < 128; i++){
          outSamples[i] = outSamplesFlush[i] * (1-i/127) +
                          outSamplesFrame[i] * (i/127)
        }
        for(i = 128; i < outputFrameLength; i++){
          outSamples[i] = outSamplesFrame[i];
        }
      }
    }
  }
```

If a configuration change is detected by the decoder the following steps shall be applied:

— Flush the internal decoder states and buffers (`FlushDecoder()`), i.e. decode a hypothetical access unit composed of all zero samples. Store the resulting output samples corresponding to non-SAOC transport channels (`outSamplesFlush`) in a temporary buffer.

— Re-initialize the decoder with the new configuration (`ReConfigureDecoder()`).

— If an SAOC 3D configuration change happened, re-initialize the SAOC 3D decoder with the new configuration (`ReConfigureSaoc3dDecoder`).

— Decode all contained pre-roll AUs and discard the resulting output samples corresponding to non-SAOC transport channels.
The output samples corresponding to SAOC transport channels (`saocInSamplesFrame`), if present, are further decoded together with the SAOC 3D payload (`nextSaoc3dFrame`) by the SAOC 3D decoder. Discard the output samples of the SAOC 3D decoder.

— Decode the current AU (`mpegh3daFrame()`). Store the resulting output samples corresponding to non-SAOC transport channels (`outSamplesFrame`) in a temporary buffer.
Decode the output samples corresponding to SAOC transport channels (`saocInSamplesFrame`), if present, together with the SAOC 3D payload (`Saoc3dFrame`). Store the output of the SAOC 3D decoder in a temporary buffer (`saocOutSamplesFrame`).

— To avoid switching artifacts apply a linear cross-fade of length 128 on `outSamplesFlush` and `outSamplesFrame`. Play-out the result of the cross-fade (`outSamples`) and the SAOC 3D decoder output (`saocOutSamplesFrame`).
If the decoded signal is available in the QMF domain, no cross-fade is applied.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

## 5.6 Buffer requirements

### 5.6.1 Minimum decoder input buffer

The following rules are used to calculate the maximum number of bits in the input buffer both for the bitstream as a whole, for any given program, or for any given elements of type ID_USAC_SCE (SCE) and ID_USAC_CPE (CPE):

The input buffer size is 6144 bits per SCE plus 12288 bits per CPE ($6144 \times NCC$), independent whether these elements are used for channels, objects, SAOC transport channels or HOA transport channels. Both, the total buffer and the individual buffer sizes are limited, such that the buffering limit can be calculated for either the entire bitstream payload or the individual audio elements permitting to strip out not required bitstream elements in the 3D audio bitstream payload to form a new bitstream payload with reduced buffer requirements. All bits for LFEs must be supplied from the total buffer requirements based on the SCEs and CPEs.

Furthermore, all bits contained in an access unit including any extension element payloads and MHAS transport overhead must also be considered in the total buffer requirements.

For the definition of NCC (Number of Considered Channels), see 1.3 of ISO/IEC 14496-3:2009.

### 5.6.2 Bit reservoir

The bit reservoir is controlled at the encoder. The maximum bit reservoir in the encoder depends on the NCC and the mean bitrate. The maximum bit reservoir size for constant rate channels can be calculated by subtracting the mean number of bits per frame from the minimum decoder input buffer size.

For example, at 32 kbit/s for two mono objects at 48 kHz sampling frequency and a frame length of 1024 samples the mean number of bits per frame ($mean_aulengh$) is:

$$mean_aulength = \frac{32000\,\frac{bits}{sec} \times 1024\,\frac{samples}{frame}}{48000\,\frac{samples}{sec}} = 682.666\ldots\,\frac{bits}{frame}$$

This leads to a maximum bit reservoir size (*max_bit_reservoir*) of

$$max_bit_reservoir = FLOOR\left(12288\,\frac{bits}{frame} - 682.666\ldots\,\frac{bits}{frame}\right) = 11605\,\frac{bits}{frame}.$$

For variable bitrate channels the encoder must operate in a way that the input buffer requirements do not exceed the minimum decoder input buffer.

For a delay optimized decoder start up the state of the bit reservoir ($bit_reservoir_state$) can be transmitted in a *buffer_fullness* field. If the buffer fullness is known at the decoder, the decoder does not have to wait until the input buffer is fully filled. Instead, before starting to decode the decoder only needs to buffer the actual frame and a number of additional bits as indicated in $bit_reservoir_state$ .

The $bit_reservoir_state$ of subsequent frames can be derived as follows:

$$bit_reservoir_state[frame] = bit_reservoir_state[frame-1] + mean_aulength - aulength[frame]$$

$aulength$ has to be adjusted such that the following restriction is met:

$$0 \le bit_reservoir_state[frame] \le max_bit_reservoir$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

### 5.6.3 Maximum bit rate

The maximum bitrate depends on the audio sampling rate (as given by **usacSamplingFrequencyIndex** and **usacSamplingFrequency**) and the output frame length (as given by **coreSbrFrameLengthIndex**). It can be calculated based on the minimum input buffer size according to the formula:

$$\frac{6144\ \frac{bits}{block} \times usacSamplingFrequency}{outputFrameLength} \times NCC$$

To give an example the maximum bitrate per channel for a sampling rate of 48 kHz and outputFrameLength of 768 samples is $384\ {}^{kbit}/_{sec}$ per *NCC*.

## 5.7 Stream Access Point requirements and inter-frame dependency

For the sake of increased coding efficiency the MPEG-H 3D audio coding scheme may exploit inter-frame redundancy. This coding strategy introduces inter-frame dependencies, which means that the decoding process of one frame requires the knowledge of the previously decoded frame.

To allow for (random) stream access MPEG-H 3D audio features the concept of the independently decodable frame in which no such inter-frame dependencies exist. Decoders can easily identify and use this independently decodable frame for start-up or as stream access points.

The bit stream field which indicates that a frame is independently decodable is usacIndependencyFlag. The flag is conveniently located at the very beginning of each mpegh3daFrame(). If usacIndependencyFlag == 1 then the present frame shall be independently decodable. See also the definition of usacIndependencyFlag in ISO/IEC 23003-3.

Consequently, an independently decodable frame shall have the following properties:

- No processing block may operate in a state in which it requires information from the previous frame to perform the decoding process. In particular:
- The context of the arithmetic coder shall be reset (arith_reset_flag == 1).
- if eSBR is used:
  - it shall not employ delta coding in the time direction over SBR frame borders.
  - sbrInfo() and sbrHeader() shall be present (sbrInfoPresent == 1, sbrHeaderPresent == 1).
  - re-use of PVC IDs is not allowed (reuse_pvcID == 0).
- if IGF is used:
  - it shall not employ delta coding in time direction over frame borders
  - it shall not make use of previous tile indices nor previous whitening levels
- if complex prediction stereo is used:
  - it shall not employ delta coding in time direction (delta_code_time == 0) over frame borders
  - it shall not make use of previous frame information (use_prev_frame==0)

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

— if MPS212 is used, bsIndependencyFlag shall be 1.

— if HOA is used, hoaIndependencyFlag shall be 1.

Note that most of the above requirements are fulfilled intrinsically by the design of the bitstream syntax.

An independently decodable frame may contain audio pre-roll information as described in 5.5.6.

In an MHAS stream an independently decodable frame may be signaled by means of a previously transmitted MHAS packet of type PACTYP_MARKER as defined in 14.4.6.3.

To be clear, in order to be able to decode the frame the information from mpegh3daConfig() must be known to the decoder as well.

# 6 Dynamic Range Control and Loudness Processing

## 6.1 Introduction

This clause describes the decoding process of loudness metadata and dynamic range control (DRC) metadata. These are needed for different tasks including loudness monitoring and normalization, dynamic range control in noisy and quiet playback environments, or for other audio enhancement scenarios.

## 6.2 Description

Coding, transmission and application of loudness information and dynamic range control gains shall be based on ISO/IEC 23003-4.

## 6.3 Syntax

### 6.3.1 Loudness Metadata

The loudness metadata is located in an mpegh3daConfigExtension as defined in Table 14. The content of mpegh3daLoudnessInfoSet() is listed in Table 61. loudnessInfo() and loudnessInfoSetExtension() are defined in ISO/IEC 23003-4. Other syntax elements are either defined in 6.3.3 or in other parts of this standard.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 61 — Syntax of mpegh3daLoudnessInfoSet()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daLoudnessInfoSet () | | |
| { | | |
| **loudnessInfoCount**; | **6** | **uimsbf** |
| for (i=0; i<loudnessInfoCount; i++) { | | |
| **loudnessInfoType**; | **2** | **uimsbf** |
| if (loudnessInfoType == 1 \|\| loudnessInfoType == 2) { | | |
| **mae_groupID**; | **7** | **uimsbf** |
| } else if (loudnessInfoType == 3) { | | |
| **mae_groupPresetID**; | **5** | **uimsbf** |
| } | | |
| loudnessInfo(); | | |
| } | | |
| **loudnessInfoAlbumPresent**; | **1** | **uimsbf** |
| if (loudnessInfoAlbumPresent) { | | |
| **loudnessInfoAlbumCount**; | **6** | **uimsbf** |
| for (i=0; i< loudnessInfoAlbumCount; i++) { | | |
| loudnessInfoType = 0; | | |
| loudnessInfo(); | | |
| } | | |
| } | | |
| **loudnessInfoSetExtensionPresent**; | **1** | **uimsbf** |
| if (loudnessInfoSetExtensionPresent == 1) { | | |
| loudnessInfoSetExtension(); | | |
| } | | |
| } | | |

### 6.3.2 Dynamic Range Control Metadata

The dynamic range control (DRC) metadata is located in an mpegh3daExtElementConfig and an mpegh3daExtElement as defined in Table 13 and Table 51. The static DRC metadata is defined by mpegh3daUniDrcConfig() listed in Table 62. The dynamic DRC metadata is defined by uniDrcGain() listed in ISO/IEC 23003-4. drcCoefficientsUniDrc(), drcInstructionsUniDrc(), uniDrcConfigExtension(), drcSampleRate, and baseChannelCount are defined in ISO/IEC 23003-4. Other syntax elements are either defined in 6.3.3 or in other parts of this standard. If loudnessInfoSetPresent == 1, the mpegh3daLoudnessInfoSet() in mpegh3daExtElementConfig() takes precedence over a present mpegh3daLoudnessInfoSet() in mpegh3daConfigExtension().

**Table 62 — Syntax of mpegh3daUniDrcConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daUniDrcConfig() | | |
| { | | |
| drcSampleRate = sampling frequency as determined from mpegh3daConfig(); | | |
| **drcCoefficientsUniDrcCount**; | **3** | **uimsbf** |
| **drcInstructionsUniDrcCount**; | **6** | **uimsbf** |
| mpegh3daUniDrcChannelLayout(); | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| for (i=0; i< drcCoefficientsUniDrcCount; i++) { | | |
| drcCoefficientsUniDrc(); | | |
| } | | **uimsbf** |
| for (i=0; i< drcInstructionsUniDrcCount; i++) { | | |
| **drcInstructionsType**; | **1..2** | **vlclbf** |
| if (drcInstructionsType == 2) { | | |
| **mae_groupID**; | **7** | **uimsbf** |
| } else if (drcInstructionsType == 3) { | | |
| **mae_groupPresetID**; | **5** | **uimsbf** |
| } | | |
| drcInstructionsUniDrc(); | | |
| } | | |
| **uniDrcConfigExtPresent**; | **1** | **uimsbf** |
| if (uniDrcConfigExtPresent == 1) { | | |
| uniDrcConfigExtension(); | | |
| } | | |
| **loudnessInfoSetPresent**; | **1** | **uimsbf** |
| if (loudnessInfoSetPresent == 1) { | | |
| mpegh3daLoudnessInfoSet(); | | |
| } | | |
| } | | |

**Table 63 — Syntax of mpegh3daUniDrcChannelLayout()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daUniDrcChannelLayout () | | |
| { | | |
| **baseChannelCount**; | **7** | **uimsbf** |
| } | | |

### 6.3.3 Data Elements

**loudnessInfoType** This field signals whether the following loudnessInfo() block refers to a fixed audio scene (default), to a specific audio element (mae_groupID), to an audio scene which includes a specific audio element (mae_groupID), or to an audio scene defined by a combination of audio elements (mae_groupPresetID).

**Table 64 — Coding and meaning of loudnessInfoType**

| loudnessInfoType (value) | binary encoding | codeword size [bits] | meaning |
|---|---|---|---|
| 0 | ‘00’ | 2 | loudnessInfo() for fixed audio scene (default) |
| 1 | ‘01’ | 2 | loudnessInfo() for single audio element defined by mae_groupID |
| 2 | ‘10’ | 2 | loudnessInfo() for audio scene which includes a specific mae_groupID |
| 3 | ‘11’ | 2 | loudnessInfo() for audio scene defined by mae_groupPresetID |

**drcInstructionsType** This field signals whether the following drcInstructionsUniDrc() block refers to a fixed audio scene (default), to an audio scene which includes a specific audio element (mae_groupID), or to an audio scene defined by a combination of audio elements (mae_groupPresetID).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 65 — Coding and meaning of drcInstructionsType**

| drcInstructionsType (value) | binary encoding | codeword size [bits] | meaning |
|---|---|---|---|
| 0 | '0' | 1 | drcInstructionsUniDrc() for fixed audio scene (default) |
| 2 | '10' | 2 | drcInstructionsUniDrc() for audio scene which includes a specific mae_groupID |
| 3 | '11' | 2 | drcInstructionsUniDrc() for audio scene defined by mae_groupPresetID |

**baseChannelCount** Sum of all present channels, objects, internal SAOC 3D channels/objects and HOA coefficient channels. baseChannelCount shall have a value as determined by the following pseudo code:

```
baseChannelCount = numAudioChannels; NOTE: from Signals3d()
baseChannelCount += numAudioObjects; NOTE: from Signals3d()
if (numSAOCTransportChannels > 0) {
    baseChannelCount += NumInputSignals; NOTE: from SAOC3DSpecificConfig()
}
if (numHOATransportChannels > 0) {
    baseChannelCount += NumOfHoaCoeffs   NOTE: from HoaConfig()
}
```

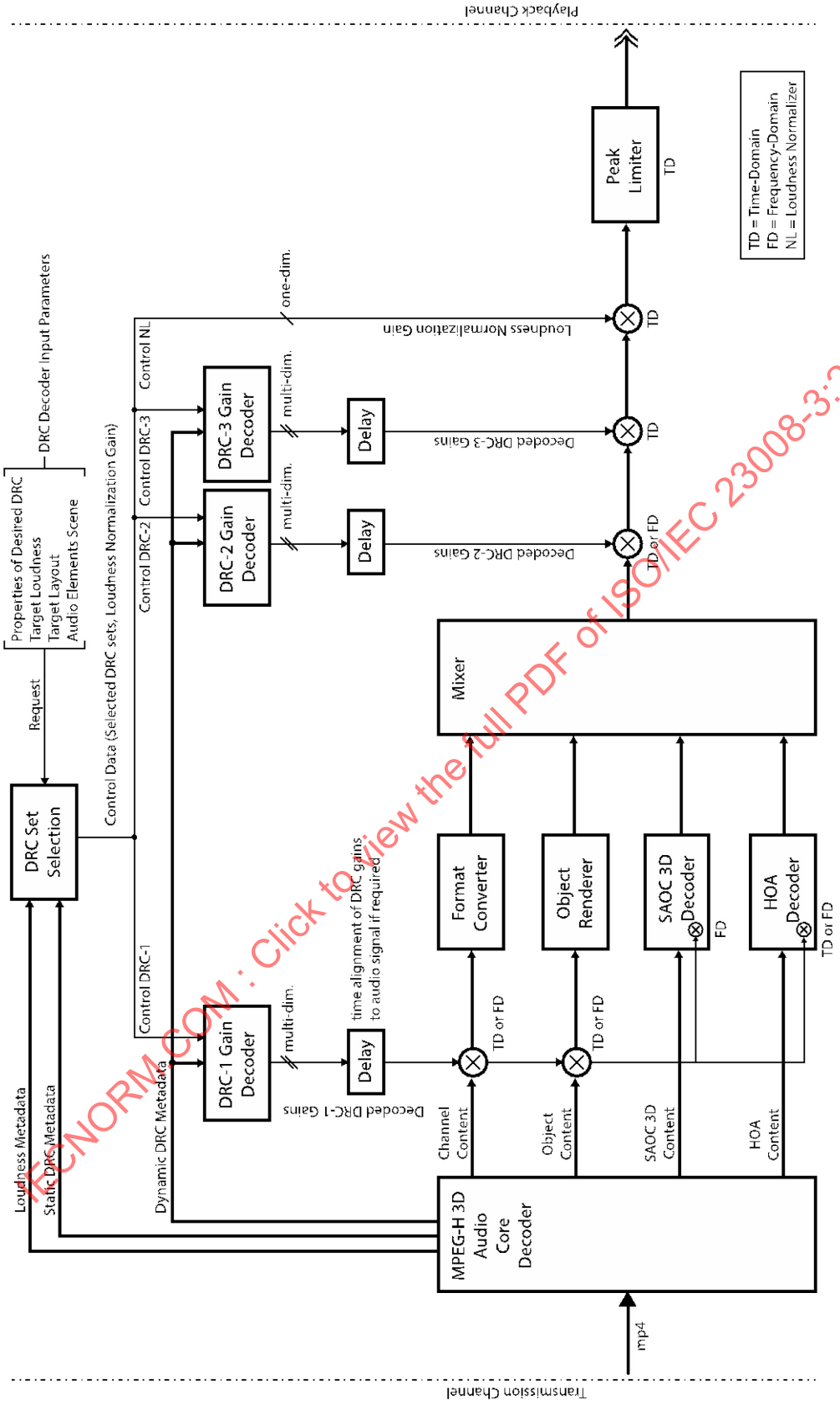
## 6.4 Decoding Process

### 6.4.1 General

The decoding and application of dynamic range control gains and loudness normalization gains is in general identical to the specification in ISO/IEC 23003-4. Therefore, the following subclauses only provide differences and specific MPEG-H related configuration details. A high-level block diagram of the complete MPEG-H decoder processing chain including blocks for dynamic range control, loudness normalization and peak limiting is depicted in Figure 16. Note that the definitions and descriptions for loudnessInfoSet() and uniDrcConfig() as specified in ISO/IEC 23003-4 equally hold for mpegh3daLoudnessInfoSet() and mpegh3daUniDrcConfig() in MPEG-H unless stated otherwise.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved




IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Figure 16 — MPEG-H decoder processing chain including blocks for dynamic range control, loudness normalization and peak limiting.**


© ISO/IEC 2015 – All rights reserved

### 6.4.2 Dynamic Range Control

There are three DRC gain decoders (denoted by DRC-1, DRC-2, and DRC-3) available in the MPEG-H decoder processing chain. All three decoders are identical and conform to ISO/IEC 23003-4. Each decoder addresses different tasks and scenarios. The decoded gains are applied by multiplication to the audio signal in the time- or frequency-domain as shown in Figure 16. Note that all three DRC gain decoders are driven by the same DRC metadata stream.

DRC-1 addresses dynamic range control for individual channels, individual objects, SAOC 3D content and HOA content. Additionally, DRC-1 shall support multi-band DRC in the 64-band QMF-domain. Dependent on the availability of the QMF representation of the audio signal, single-band DRC gains should be either applied in the time-domain or in the QMF-domain. Further details on the DRC application to SAOC 3D and HOA content are defined in 6.4.5 and 6.4.6, respectively.

DRC-2 addresses dynamic range control for the entire audio scene. DRC-2 shall also support multi-band DRC in the 64-band QMF-domain. Dependent on the availability of the QMF representation of the audio signal, single-band DRC gains should be either applied in the time-domain or in the QMF-domain. DRC sets designed for DRC-2 are restricted to one DRC channel group to be compatible to all target rendering layouts after the mixer module. Thus, DRC-2 is also best suited for simple DRC tasks especially in case of low-complexity requirements.

Note that in MPEG-H multi-band DRC gains shall be always applied in the QMF-domain or in the SAOC processing band domain. The usage of the multi-band DRC filterbank defined in ISO/IEC 23003-4 is not permitted.

DRC-3 addresses dynamic range control and/or playback level dependent guided clipping prevention for specific target layouts (channel configurations). DRC-3 shall always operate in the time-domain. Thus, DRC-3 only supports single-band DRC. However, it shall support multiple DRC channel groups. Note that the gain application of DRC-2 and DRC-3 can be only combined if both operate in the time-domain.

Due to general restrictions in ISO/IEC 23003-4, either one or two DRC decoders are active simultaneously to achieve one specific DRC effect. Note that the DRC-3 gain decoder is only suited for a restricted number of target layouts. In all cases, the DRC gain decoders shall provide gains suitable for the present audio signal representation according to the downsampling and frequency mapping rules defined in ISO/IEC 23003-4. Note that if one DRC gain decoder is used to apply a non-DRC effect such as fading or ducking, all three DRC gain decoders can be active simultaneously if the others additionally serve a specific DRC effect.

If binaural rendering is enabled, DRC-2 shall be applied to the output of the binaural renderer (not shown in Figure 16). The same holds for the loudness normalization gain. Note that DRC-3 gains are not supported in case of binaural rendering.

### 6.4.3 Usage of downmixId in MPEG-H

According to ISO/IEC 23003-4, a non-zero identifier termed downmixId can be used to externally refer to a specific downmix or rendering layout. In ISO/IEC 23003-4, downmixId is defined in the downmixInstructions() payload. If a DRC set contains a specific downmixId, the DRC set shall be applied to the output of the downmix or rendering which downmixId refers to. If loudness metadata contains a specific downmixId, this indicates that the loudness metadata was obtained from the output of the downmix or rendering which downmixId refers to.

In MPEG-H, downmixInstructions() is replaced by DownmixMatrixSet() as defined in Table 17, which allows to define a downmixId not only for transmitted downmix matrices, but also for default downmix matrices available on decoder side (format converter). At the decoder a downmixId matching algorithm determines which downmixId is present as outlined in 10.3.1.2.

According to ISO/IEC 23003-4, downmixId has two reserved values, namely 0x0 and 0x7F, which equally holds for MPEG-H.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

A downmixId of 0x0 indicates that a DRC set shall be applied to the unmodified MPEG-H 3D Audio Core Decoder output by DRC-1. On encoder side, the DRC gain sequences have to be sequentially assigned to the mixed-content multi-channel output of the MPEG-H 3D Audio Core Decoder. For combined channel-, object-, SAOC- and HOA-content, the size of the DRC assignment loop is defined by the baseChannelCount, which is the sum of all channels, all objects, all internal SAOC 3D channels/objects and the square of HoaOrder+1, NumOfHoaCoeffs (see definition in Table 63). In contrast to ISO/IEC 23003-4, DRC sets with ducking effect are only permitted for DRC sets with a downmixId of 0x0. Furthermore, in MPEG-H it is permitted to define more than one DRC set with ducking effect for the same downmixId if those DRC sets are differentiated by unique audio element group identifiers (mae_groupIDs/mae_groupPresetIDs) (see Table 62 and clause 15). In case of loudness metadata a downmixId of 0x0 shall refer to the reference layout (referenceLayout) specified in mpegh3daConfig().

A downmixId of 0x7F indicates that a DRC set shall be applied by DRC-2. In contrast to ISO/IEC 23003-4, it is not allowed to freely choose the application position of DRC sets coded with a downmixId of 0x7F. For example, the clipping behavior for one specific downmix configuration will be different if a DRC set is applied before or after the format converter in MPEG-H. Note that DRC-2 sets can be always applied independent of the actual target layout. In case of loudness metadata a downmixId of 0x7F is not permitted.

All other downmixId values (except 0x0 and 0x7F) indicate that a DRC set shall be applied to a specific target layout by DRC-3. In case of loudness metadata all other downmixId values shall refer to the target layout specified by downmixId.

The MPEG-H meaning of downmixId is again summarized in Table 66.

**Table 66 — Meaning of downmixId in MPEG-H**

| Metadata Type | downmixId == 0x0 | downmixId == 0x7F | downmixId != 0x0 && downmixId != 0x7F |
|---|---|---|---|
| drcInstructionsUniDrc() (DRC set) | Shall be applied by DRC-1. | Shall be applied by DRC-2. | Shall be applied by DRC-3. |
| loudnessInfo() (loudness metadata) | Measurement of reference layout. | Not permitted. | Measurement of target layout specified by downmixId. |

### 6.4.4 DRC Set Selection Process

#### 6.4.4.1 General

The DRC set selection process shall be implemented according to ISO/IEC 23003-4. Note that dmxInstructions() is replaced by DownmixMatrixSet() in MPEG-H. DownmixMatrixSet() is defined in Table 17 .

If information on present audio elements is available (mae_groupID/mae_groupPresetID) (see clause 15), the pre-selection process specified in ISO/IEC 23003-4 shall be extended by two additional requirements as listed in Table 67. Note that if in pre-selection step #8 the requirement of clipping prevention needs to be relaxed, at least one DRC set has to remain that either matches the present audio elements or has to be independent of any audio elements (drcInstructionsType==0). The final selection based on present audio elements shall be performed after the final selection based on output peak value (see 6.4.4.2, 6.4.4.3 and ISO/IEC 23003-4). If information on present audio elements is not available to the DRC selection process, DRC sets and loudness metadata with defined audio element group identifiers (mae_groupID/mae_groupPresetID) shall be ignored by the DRC selection process.

Note that the applicable loudnessInfo() block for loudness normalization according to ISO/IEC 23003-4 is also dependent on the present audio elements. If loudnessInfo() blocks with matching group identifiers for present audio elements are not available, loudnessInfo() blocks with undefined group identifiers shall be used instead (loudnessInfoType==0).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

A DRC set with ducking effect is automatically selected if the group identifiers of the present audio elements match the group identifier of the DRC set with ducking effect. Note that in MPEG-H a DRC set with ducking effect is always automatically selected if it doesn't define an audio element group identifier (drcInstructionsType==0).

The selection of DRC sets with fading effect shall be done according to ISO/IEC 23003-4.

**Table 67 — Requirements for DRC pre-selection (additional MPEG-H requirements).**

| # | Requirement | Applicability | Comment |
|---|---|---|---|
| 9 | mae_groupPresetID matches<br>or is undefined | If mae_groupPresetID is present. | See 6.4.4.2 |
| 10 | mae_groupID matches<br>or is undefined | If mae_groupID is present<br>and no match was found in step #9. | See 6.4.4.3 |

#### 6.4.4.2 Pre-selection based on present mae_groupPresetID (#9)

This pre-selection step addresses the case when a specific audio scene defined by a combination of audio elements (mae_groupPresetID) is present. All DRC sets, which define a matching mae_groupPresetID (see Table 62) shall be selected in this step. All other DRC sets with defined mae_groupPresetID shall be discarded. If at least one DRC set has a matching mae_groupPresetID, pre-selection step #10 shall be omitted and all DRC sets with defined mae_groupID shall be discarded. If no DRC set has a matching mae_groupPresetID, all DRC sets with defined mae_groupPresetID are discarded and the pre-selection shall continue with step #10. If several mae_groupPresetIDs are present, the matching shall be done separately for each identifier. Note that DRC sets which don't define a mae_groupPresetID shall also pass this pre-selection step.

If in the final selection according to ISO/IEC 23003-4 multiple DRC sets with different mae_groupPresetIDs are selected, the DRC set, which defines the largest number of members in its associated mae_groupPresetID, shall be selected. If there are still multiple DRC sets with different mae_groupPresetIDs selected, select the DRC set with the lowest mae_groupPresetID. Note that if the user has actively chosen a specific mae_groupPresetID, all corresponding DRC sets shall be preferred in this final selection step.

If in the final selection according to ISO/IEC 23003-4 multiple DRC sets with defined and undefined mae_groupPresetID are selected, all DRC sets with undefined mae_groupPresetID shall be discarded.

#### 6.4.4.3 Pre-selection based on present mae_groupID (#10)

This pre-selection step addresses the case when an audio scene is present which includes a specific audio element (mae_groupID). All DRC sets which define a matching mae_groupID (see Table 62) shall be selected in this step. All other DRC sets with defined mae_groupID shall be discarded. If no DRC set has a matching mae_groupID, all DRC sets with defined mae_groupID shall be discarded. If several mae_groupIDs are present, the matching shall be done separately for each identifier. Note that DRC sets which don't define a mae_groupID shall also pass this pre-selection step.

If in the final selection according to ISO/IEC 23003-4 multiple DRC sets with defined and undefined mae_groupID are selected, all DRC sets with undefined mae_groupID shall be discarded.

### 6.4.5 DRC-1 for SAOC 3D Content

For SAOC 3D, the DRC-1 decoder shall produce one DRC gain $\mathrm{drc}_{\mathrm{ch/obj}}(n,m,i)$ per QMF time slot *n*, per SAOC 3D processing band *m* and per internal SAOC 3D channel/object *i*. Appropriate down-sampling and frequency mapping rules are specified in ISO/IEC 23003-4.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

The decoded DRC gains are always applied to the SAOC 3D processing bands. More precisely, the gains are applied on the SAOC 3D rendering matrix. The corresponding SAOC 3D rendering coefficients are defined for each parameter time slot $l$ and processing band $m$. Note that the SAOC 3D decoder has to provide the number of processing bands to the DRC decoder. The corresponding center frequencies can be derived from ISO/IEC 23003-1.

The final SAOC 3D DRC rendering matrix is computed by replacing the SAOC 3D unmodified rendering matrix $\mathbf{R}$ defined in 9.5.3.4 by:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{\text{ch}} & \mathbf{R}_{\text{obj}} \end{bmatrix} \begin{bmatrix} \mathbf{DRC}_{\text{ch}} & 0 \\ 0 & \mathbf{DRC}_{\text{obj}} \end{bmatrix},$$

where $\mathbf{DRC}_{\text{ch}}$ of size $N_{\text{ch}} \times N_{\text{ch}}$ represents the DRC gain matrix associated with the channel-content input and $\mathbf{DRC}_{\text{obj}}$ of size $N_{obj} \times N_{obj}$ represents the DRC gain matrix associated with the object-content input.

The DRC gain matrices are defined for each parameter time slot $l$ and processing band $m$ and given by:

$$\mathbf{DRC}_{\text{ch}}^{l,m}(i,j) = \begin{cases} \text{drc}_{\text{ch}}(n_0, m, i), & \text{if } i = j \\ 0 & \text{, otherwise} \end{cases},$$

and

$$\mathbf{DRC}_{\text{obj}}^{l,m}(i,j) = \begin{cases} \text{drc}_{\text{obj}}(n_0, m, i), & \text{if } i = j \\ 0 & \text{, otherwise} \end{cases},$$

where $n_0$ represents the last time slot in parameter set $l$.

### 6.4.6 DRC-1 for HOA Content

#### 6.4.6.1 General

The DRC-1 gains for HOA are applied to the HOA signal before rendering and might be combined with rendering. The DRC-1 gains for HOA can be applied in the time-domain or in the QMF-domain.

#### 6.4.6.2 Application of DRC-1 Gains in the Time-Domain

The DRC decoder shall provide $(N+1)^2$ gain values $\boldsymbol{g}_{drc} = [g_1, \ldots, g_{(N+1)^2}]^T$ according to the number of HOA coefficient channels of the HOA signal $\boldsymbol{c}$. $N$ is the HOA order.

Application of DRC gains to the HOA signals:

$$\boldsymbol{c}_{drc} = \boldsymbol{D}_{DSHT}^{-1}\, diag(\boldsymbol{g}_{drc})\boldsymbol{D}_{DSHT}\, \boldsymbol{c},$$

where $\boldsymbol{c}$ is a vector of one time sample of HOA coefficients ($\boldsymbol{c} \in \mathbb{R}^{(N+1)^2 \times 1}$), and $\boldsymbol{D}_{DSHT} \in \mathbb{R}^{(N+1)^2 \times (N+1)^2}$ and its inverse $\boldsymbol{D}_{DSHT}^{-1}$ are matrices related to a Discrete Spherical Harmonics Transform (DSHT) optimized for DRC purposes.

Informative remark: To decrease the computational load by $(N+1)^4$ operations per sample, it can be advantageous to include the rendering step and calculate the loudspeaker signals directly by: $\boldsymbol{w}_{drc} = (\boldsymbol{D}\ \boldsymbol{D}_{DSHT}^{-1})\ (diag(\boldsymbol{g}_{drc})\boldsymbol{D}_{DSHT})\ \boldsymbol{c}$, where $\boldsymbol{D}$ is the rendering matrix and $(\boldsymbol{D}\ \boldsymbol{D}_{DSHT}^{-1})$ can be pre-computed.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

If all gains $g_1, \ldots, g_{(N+1)^2}$ have the same value of $g_{\mathrm{drc}}$, a single DRC channel group has been used. This case shall be flagged by the DRC-1 decoder because in this case the calculation of spatial filter is not needed and the calculation simplifies to $\boldsymbol{c}_{drc} = g_{\mathrm{drc}}\ \boldsymbol{c}$.

#### 6.4.6.3 Calculation of DSHT matrices for DRC-1 Gains

The matrices to determine the spatial filter $\boldsymbol{D}_{DSHT}$ and its inverse $\boldsymbol{D}_{DSHT}^{-1}$ are calculated as follows:
A set of spherical positions $\mathfrak{D}_{DSHT} = [\boldsymbol{\Omega}_1, \ldots, \boldsymbol{\Omega}_l, \ldots, \boldsymbol{\Omega}_{(N+1)^2}]$ with $\boldsymbol{\Omega}_l = [\theta_l, \phi_l]^T$ and related quadrature gains $\boldsymbol{q} \in \mathbb{R}^{(N+1)^2 \times 1}$ are selected indexed by the HOA order $N$ from Annex F.23. A mode matrix $\boldsymbol{\Psi}_{DSHT}$ related to these positions is calculated. For details see Annex F.1. A first prototype matrix is calculated by $\breve{\boldsymbol{D}}_1 = diag(\boldsymbol{q}) \frac{\boldsymbol{\Psi}_{DSHT}}{(N+1)^2}$. A compact singular value decomposition is performed $\breve{\boldsymbol{D}}_1 = \boldsymbol{USV}^T$ and a new prototype matrix is calculated by: $\hat{\breve{\boldsymbol{D}}}_2 = \boldsymbol{UV}^T$. This matrix is normalized by: $\breve{\boldsymbol{D}}_2 = \frac{\hat{\breve{\boldsymbol{D}}}_2}{\left\|\hat{\breve{\boldsymbol{D}}}_2\right\|_{fro}}$. A row-vector $\boldsymbol{e}$ is calculated by $= \frac{\mathbf{1}_L^T \breve{\boldsymbol{D}}_2 - [1,0,0,\ldots,0]}{(N+1)^2}$, where $[1,0,0,..,0]$ is a row vector of $(N+1)^2$ all zero elements except for the first element with a value of one. $\mathbf{1}_L^T \breve{\boldsymbol{D}}_2$ denotes the sum of rows of $\breve{\boldsymbol{D}}_2$. The optimized DSHT matrix $\boldsymbol{D}_{DSHT}$ is now derived by
$\boldsymbol{D}_{DSHT} = \breve{\boldsymbol{D}}_2 - [\boldsymbol{e}^T, \boldsymbol{e}^T, \boldsymbol{e}^T, \ldots]^T$.

#### 6.4.6.4 Application of DRC-1 Gains in the QMF-Domain

The DRC-1 decoder provides a gain value $g_{ch}(n,m)$ for every time frequency tile $(n,m)$ for $(N+1)^2$ spatial channels. The gains for time slot n and frequency band $m$ are arranged in $\boldsymbol{g}(n,m) \in \mathbb{R}^{(N+1)^2 \times 1}$.

Multi-band DRC gains are applied in the QMF-domain. The processing steps are shown in **Figure 17**. The reconstructed HOA signal is transformed into spatial domain by (inverse DSHT): $W_{DSHT} = D_{DSHT} C$, where $\boldsymbol{C} \in \mathbb{R}^{(N+1)^2 \times \tau}$ is a block of $\tau$ HOA samples and $\boldsymbol{W}_{DSHT} \in \mathbb{R}^{(N+1)^2 \times \tau}$ is a block of spatial samples matching the input time granularity of the QMF filter bank. Then the QMF analysis filter bank is applied. Let $\hat{\boldsymbol{w}}_{DSHT}(n,m) \in \mathbb{C}^{(N+1)^2 \times 1}$ denote the vector of spatial channels per time frequency tile $(n,m)$. Then the DRC gains are applied: $\breve{\boldsymbol{w}}_{DRC}(n,m) = diag(\boldsymbol{g}(n,m)) \hat{\boldsymbol{w}}_{DSHT}(n,m)$.To minimize the computational complexity the DSHT and rendering to loudspeaker channels are combined: $\boldsymbol{w}(n,m) = \boldsymbol{D}\ \boldsymbol{D}_{DSHT}^{-1}\ \breve{\boldsymbol{w}}_{DRC}(n,m)$, where $\boldsymbol{D}$ denotes the HOA rendering matrix. The QMF signals then can be fed to the mixer for further processing.

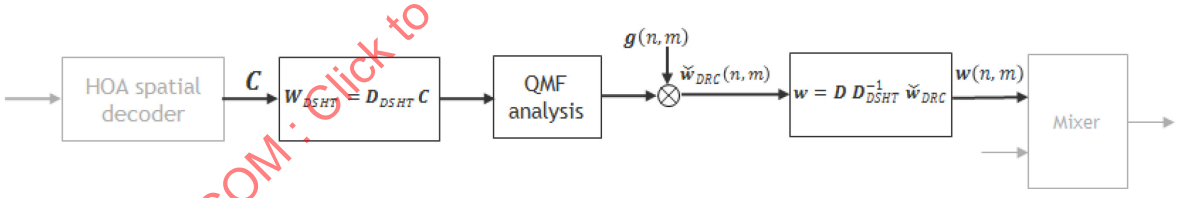



**Figure 17 — DRC-1 gain application for HOA in the QMF-domain combined with rendering step.**

If only a single DRC channel group is present, this should be flagged by the DRC-1 decoder because again computational simplifications are possible (Figure 18). In this case the gains in vector $g(n,m)$ all share the same value of $g_{drc}(n,m)$. The QMF filter bank can be directly applied to the HOA signal and the gain $g_{drc}(n,m)$ can be multiplied in the QMF-domain. The resulting loudspeaker channels in the QMF-domain after rendering are obtained by $\boldsymbol{w}(n,m) = g_{drc}(n,m) \boldsymbol{D} \boldsymbol{c}_{DRC}(n,m)$.

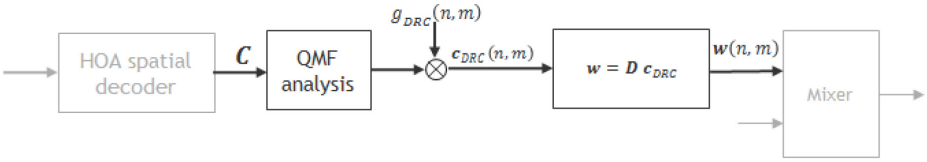



**Figure 18 — DRC-1 gain application for HOA in the QMF-domain combined with rendering step for the simple case of a single DRC channel group.**

Tables for the application DRC in HOA can be found in Annex F.23

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

### 6.4.7 Loudness Normalization

Loudness normalization shall be done in the time-domain after DRC-3. The processing shall be done according to ISO/IEC 23003-4. Note that the normalization gain also depends on the result of the DRC set selection process as shown in Figure 16.

Note that in case of binaural rendering loudness normalization shall be done after DRC-2.

### 6.4.8 Peak Limiter

The peak limiter is informative but strongly recommended. Further details are specified in the informative Annex D. If present, the peak limiter shall be placed at the very end of the audio processing chain. The processing should be done according to ISO/IEC 23003-4.

Note that a content dependent bypass of the peak limiter is only possible if a transmitted peak value fully matches the current MPEG-H playback configuration. A peak value might not fully match if, e.g. single-band DRC gains are applied in a different domain than on encoder side, the output setup has displaced loudspeakers, the metadata audio element parameters are different from the default setting (see clause 15), or fallback values were used to derive the peak value.

### 6.4.9 Time-Synchronization of DRC gains

According to ISO/IEC 23003-4, the decoded DRC gains are synchronized to the audio signal at the decoder.

In MPEG-H, additional audio delay can occur on decoder side due to additional QMF transforms or other optional tools like, e.g. format conversion or binaural rendering. Therefore, the decoded DRC gains shall be synchronized to the lowest possible audio signal delay at their application location. If additional audio delay is present at their application location on decoder side, the decoded DRC gains shall be delayed after the DRC gain decoder for correct time-synchronization to the audio signal.

# 7 Object Metadata Decoding

## 7.1 Introduction

This clause describes the decoding process of object metadata, i.e. geometrical data for audio objects. These are needed to apply the object rendering specified in clause 8.

## 7.2 Description

Metadata is conveyed for every audio object as given spatial positions (azimuth, elevation, and radius) and a linear gain at defined timestamps.

The units of decoded values for the components are given in Table 68.A visualization of the polar coordinate system for azimuth and elevation is shown in Figure 19.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Table 68 — Units of the decoded object metadata components**

| component | unit | value range |
|---|---|---|
| azimuth | ° (degree) | -180; 180 |
| elevation | ° (degree) | -90; 90 |
| radius | m (meter) | 0.5; 16 |
| gain | none (linear) | 0.004; 5.957 |
| Spread | ° (degree) | 0; 180 |


© ISO/IEC 2015 – All rights reserved

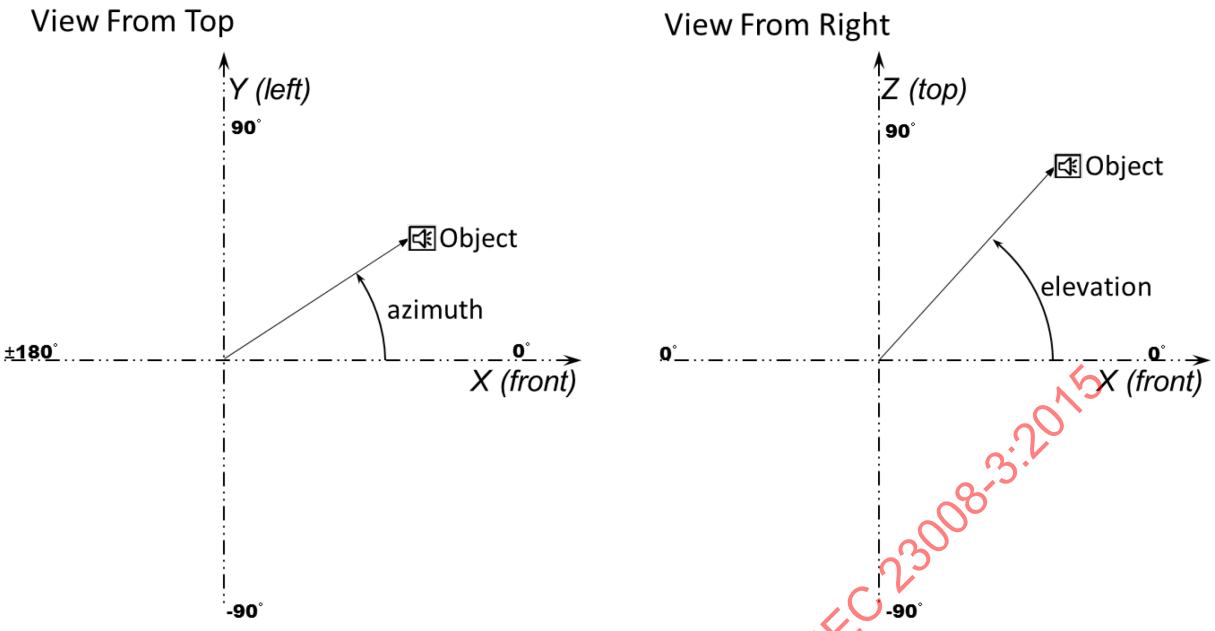



**Figure 19 — Visualization of polar coordinate system for signaling spatial positions of audio objects**

## 7.3 Syntax

### 7.3.1 Object Metadata Configuration

As mentioned in the extension payload (see Table 13), the ObjectMetadataConfig() specifies the metadata decoding method.

**Table 69 — Syntax of ObjectMetadataConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| ObjectMetadataConfig() | | |
| { | | |
| **lowDelayMetadataCoding;** | **1** | **bslbf** |
| **hasCoreLength;** | **1** | **bslbf** |
| if (!hasCoreLength) { | | |
| **frameLength;** | **6** | **uimsbf** |
| OAMFrameLength = frameLength<<6; | | |
| } else { | | |
| OAMFrameLength = outputFrameLength; | | |
| } | | |
| **hasScreenRelativeObjects;** | **1** | **bslbf** |
| if( hasScreenRelativeObjects ) { | | |
| for ( o = 0; o < num_objects; o++ ) { | | |
| **isScreenRelativeObject**[o]**;** | **1** | **bslbf** |
| } | | |
| } | | |
| **hasDynamicObjectPriority** | **1** | **bslbf** |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

If the flag **lowDelayMetadataCoding** is true, Low Delay object metadata are present in the bit stream and therefore, the Low Delay object metadata syntax must be used. Otherwise, efficient object metadata are present int the bit stream and the according syntax must be used.

### 7.3.2 Top level object metadata syntax

**Table 70 — Syntax of object_metadata()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| object_metadata()<br>{<br>  if (lowDelayMetadataCoding==0) {<br>  object_metadata_efficient()<br>  } else {<br>  object_metadata_low_delay()<br>  }<br>} | | |

### 7.3.3 Subsidiary payloads for efficient object metadata decoding

**Table 71 — Syntax of object_metadata_efficient()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| object_metadata_efficient()<br>{<br>  intracoded_object_metadata_efficient();<br>  **has_differential_metadata;**<br>  if (has_differential_metadata) {<br>  differential_object_metadata();<br>  }<br>} | **1** | **bslbf** |

**Table 72 — Syntax of intracoded_object_metadata_efficient()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| intracoded_object_metadata_efficient()<br>{ | | |
| **ifperiod;** | **6** | **uimsbf** |
| if (num_objects>1) {<br>**common_azimuth;** | **1** | **bslbf** |
| if (common_azimuth) {<br>**default_azimuth;** | **8** | **tcimsbf** |
| }<br>else {<br>for (o = 0; o < num_objects; o++) {<br>**position_azimuth[o];** | **8** | **tcimsbf** |
| }<br>}<br>**common_elevation;** | **1** | **bslbf** |
| if (common_elevation) {<br>**default_elevation;** | **6** | **tcimsbf** |
| }<br>else {<br>for (o = 0; o < num_objects; o++) { | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
                position_elevation[o];                          6       tcimsbf
            }
        }
        common_radius;                                          1       bslbf
        if (common_radius) {
            default_radius;                                     4       uimsbf
        }
        else {
            for (o = 0; o < num_objects; o++) {
                position_radius[o];                             4       uimsbf
            }
        }
        common_gain;                                            1       bslbf
        if (common_gain) {
            default_gain;                                       7       tcimsbf
        }
        else {
            for (o = 0; o < num_objects; o++) {
                gain_factor[o];                                 7       tcimsbf
            }
        }
        common_spread;                                          1       bslbf
        if (common_spread) {
            default_spread;                                     7       uimsbf
        }
        else {
            for (o = 0; o < num_objects; o++) {
                spread[o];                                      7       uimsbf
            }
        }
        if (hasDynamicObjectPriority) {
            common_dynamic_object_priority;                     1       bslbf
            if (common_dynamic_object_priority) {
                default_dynamic_object_priority;                3       uimsbf
            }
            else {
                for (o = 0; o < num_objects; o++) {
                    dynamic_object_priority[o];                 3       uimsbf
                }
            }
        }
    }
    else {
        position_azimuth;                                       8       tcimsbf
        position_elevation;                                     6       tcimsbf
        position_radius;                                        4       uimsbf
        gain_factor;                                            7       tcimsbf
        spread;                                                 7       uimsbf
        if (hasDynamicObjectPriority) {
            dynamic_object_priority;                            3       uimsbf
        }
    }
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 73 — Syntax of differential_object_metadata()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| differential_object_metadata() { | | |
| **bits_per_point;** | **4** | **uimsbf** |
| **fixed_azimuth;** | **1** | **bslbf** |
| if (!fixed_azimuth) { | | |
| for (o = 0; o < num_objects; o++) { | | |
| **flag_azimuth;** | **1** | **bslbf** |
| if (flag_azimuth) { | | |
| num_points_azimuth = offset_data(bits_per_point); | | |
| **nbits_azimuth;** | **3** | **uimsbf** |
| for (p = 0; p < num_points_azimuth; p++) { | | |
| **differential_azimuth[o][p];** | **nbits_azimuth + 2** | **tcimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| **fixed_elevation;** | **1** | **bslbf** |
| if (!fixed_elevation) { | | |
| for (o = 0; o < num_objects; o++) { | | |
| **flag_elevation;** | **1** | **bslbf** |
| if (flag_elevation) { | | |
| num_points_elevation = offset_data(bits_per_point); | | |
| **nbits_elevation;** | **3** | **uimsbf** |
| for (p = 0; p < num_points_elevation; p++) { | | |
| **differential_elevation[o][p];** | **nbits_elevation + 2** | **tcimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| **fixed_radius;** | **1** | **bslbf** |
| if (!fixed_radius) { | | |
| for (o = 0; o < num_objects; o++) { | | |
| **flag_radius;** | **1** | **bslbf** |
| if (flag_radius) { | | |
| num_points_radius = offset_data(bits_per_point); | | |
| **nbits_radius;** | **3** | **uimsbf** |
| for (p = 0; p < num_points_radius; p++) { | | |
| **differential_radius[o][p];** | **nbits_radius + 2** | **tcimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| **fixed_gain;** | **1** | **bslbf** |
| if (!fixed_gain) { | | |
| for (o = 0; o < num_objects; o++) { | | |
| **flag_gain;** | **1** | **bslbf** |
| if (flag_gain) { | | |
| num_points_gain = offset_data(bits_per_point); | | |
| **nbits_gain;** | **3** | **uimsbf** |
| for (p = 0; p < num_points_gain; p++) { | | |
| **differential_gain[o][p];** | **nbits_gain + 2** | **tcimsbf** |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
        }
    }
    fixed_spread;                                                   1               bslbf
    if (!fixed_spread) {
        for (o = 0; o < num_objects; o++) {
            flag_spread;                                            1               bslbf
            if (flag_spread) {
                num_points_spread = offset_data(bits_per_point);
                nbits_spread;                                       3               uimsbf
                for (p = 0; p < num_points_spread; p++) {
                    differential_spread[o][p];                      nbits_spread + 2  tcimsbf
                }
            }
        }
    }
    if (hasDynamicObjectPriority) {
        fixed_dynamic_object_priority;                              1               bslbf
        if (!fixed_dynamic_object_priority) {
            for (o = 0; o < num_objects; o++) {
                flag_dynamic_object_priority;                       1               bslbf
                if (flag_dynamic_object_priority) {
                    num_points_dynamic_object_priority = offset_data(bits_per_point);
                    nbits_dynamic_object_priority;                  2               uimsbf
                    for (p = 0; p < num_points_dynamic_object_priority; p++) {
                        differential_dynamic_object_priority[o][p]; nbits_dynamic_
                                                                    object_priority
                                                                    + 2             tcimsbf
                    }
                }
            }
        }
    }
}
```

**Table 74 — Syntax of offset_data()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| int offset_data(bits_per_point) { | | |
| **bitfield_syntax;** | **1** | **bslbf** |
| if (bitfield_syntax) { | | |
| **offset_bitfield;** | **ifperiod*** | **bslbf** |
| num_points = sum(offset_bitfield); | | |
| } | | |
| else { | | |
| **npoints;** | **bits_per_point** | **uimsbf** |
| num_points = npoints + 1; | | |
| for (p = 0; p < num_points; p++) { | | |
| **foffset[p];** | **ceil(log2(ifperiod*))** | **uimsbf** |
| } | | |
| } | | |
| return num_points; | | |
| } | | |
| * Known from the intracoded object metadata for this current iframe_period | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

### 7.3.4 Subsidiary payloads for object metadata decoding with low delay

**Table 75 — Syntax of object_metadata_low_delay ()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| object_metadata_low_delay ()<br>{<br>  **has_intracoded_object_metadata;**<br>  if (has_intracoded_object_metadata) {<br>  intracoded_object_metadata_low_delay();<br>  }<br>  else {<br>  dynamic_object_metadata();<br>  }<br>} | **1** | **bslbf** |

**Table 76 — Syntax of intracoded_object_metadata_low_delay()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| intracoded_object_metadata_low_delay()<br>{<br>  if (num_objects>1) {<br>  **fixed_azimuth;**<br>  if (fixed_azimuth) {<br>  **default_azimuth;**<br>  }<br>  else {<br>  **common_azimuth;**<br>  if (common_azimuth) {<br>  **default_azimuth;**<br>  }<br>  else {<br>  for (o = 0; o < num_objects; o++) {<br>  **position_azimuth[o];**<br>  }<br>  }<br>  }<br>  **fixed_elevation;**<br>  if (fixed_elevation) {<br>  **default_elevation;**<br>  }<br>  else {<br>  **common_ elevation;**<br>  if (common_elevation) {<br>  **default_elevation;**<br>  }<br>  else {<br>  for (o = 0; o < num_objects; o++) {<br>  **position_elevation[o];**<br>  }<br>  }<br>  }<br>  **fixed_radius;** | **1**<br>**8**<br>**1**<br>**8**<br>**8**<br>**1**<br>**6**<br>**1**<br>**6**<br>**6**<br>**1** | **bslbf**<br>**tcimsbf**<br>**bslbf**<br>**tcimsbf**<br>**tcimsbf**<br>**bslbf**<br>**tcimsbf**<br>**bslbf**<br>**tcimsbf**<br>**tcimsbf**<br>**bslbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
if (fixed_radius) {
    default_radius;                                  4        uimsbf
}
else {
    common_radius;                                   1        bslbf
    if (common_radius) {
        default_radius;                              4        uimsbf
    }
    else {
        for (o = 0; o < num_objects; o++) {
            position_ radius[o];                     4        uimsbf
        }
    }
}
fixed_gain;                                          1        bslbf
if (fixed_gain) {
    default_gain;                                    7        tcimsbf
}
else {
    common_gain;                                     1        bslbf
    if (common_gain) {
        default_gain;                                7        tcimsbf
    }
    else {
        for (o = 0; o < num_objects; o++) {
            gain_factor[o];                          7        tcimsbf
        }
    }
}
fixed_spread;                                        1        bslbf
if (fixed_spread) {
    default_spread;                                  7        uimsbf
}
else {
    common_spread;                                   1        bslbf
    if (common_spread) {
        default_spread;                              7        uimsbf
    }
    else {
        for (o = 0; o < num_objects; o++) {
            spread[o];                               7        uimsbf
        }
    }
}
if (hasDynamicObjectPriority) {
    fixed_dynamic_object_priority;                   1        bslbf
    if (fixed_dynamic_object_priority) {
        default_dynamic_object_priority;             3        uimsbf
    }
    else {
        common_dynamic_object_priority;              1        bslbf
        if (common_dynamic_object_priority) {
            default_dynamic_object_priority;         3        uimsbf
        }
        else {
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| for (o = 0; o < num_objects; o++) { | | |
| **dynamic_object_priority[o];** | **3** | **uimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| } | | |
| else { | | |
| **position_azimuth;** | **8** | **tcimsbf** |
| **position_elevation;** | **6** | **tcimsbf** |
| **position_radius;** | **4** | **uimsbf** |
| **gain_factor;** | **7** | **tcimsbf** |
| **spread;** | **7** | **uimsbf** |
| if (hasDynamicObjectPriority) { | | |
| **dynamic_object_priority;** | **3** | **uimsbf** |
| } | | |
| } | | |
| } | | |

**Table 77 — Syntax of dynamic_object_metadata()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| dynamic_object_metadata() { | | |
| **flag_absolute;** | **1** | **bslbf** |
| for (o = 0; o < num_objects; o++) { | | |
| **has_object_metadata;** | **1** | **bslbf** |
| if (has_object_metadata) { | | |
| single_dynamic_object_metadata( flag_absolute ); | | |
| } | | |
| } | | |
| } | | |

**Table 78 — Syntax of single_dynamic_object_metadata()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| single_dynamic_object_metadata ( flag_absolute ) { | | |
| if ( flag_absolute ) { | | |
| if (!fixed_azimuth*) { | | |
| **position_azimuth;** | **8** | **tcimsbf** |
| } | | |
| if (!fixed_elevation*) { | | |
| **position_elevation;** | **6** | **tcimsbf** |
| } | | |
| if (!fixed_radius*) { | | |
| **position_radius;** | **4** | **uimsbf** |
| } | | |
| if (!fixed_gain*) { | | |
| **gain_ factor;** | **7** | **tcimsbf** |
| } | | |
| if (!fixed_spread*) { | | |
| **spread;** | **7** | **uimsbf** |
| } | | |
| if (hasDynamicObjectPriority) { | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| if (!fixed_dynamic_object_priority*) { | | |
| **dynamic_object_priority;** | **3** | **uimsbf** |
| } | | |
| } | | |
| } | | |
| else { | | |
| **nbits;** | **3** | **uimsbf** |
| num_bits = nbits +2; | | |
| if (!fixed_azimuth*) { | | |
| **flag_azimuth;** | **1** | **bslbf** |
| if (flag_azimuth) { | | |
| **position_azimuth_difference;** | **num_bits** | **tcimsbf** |
| } | | |
| } | | |
| if (!fixed_elevation*) { | | |
| **flag_elevation;** | **1** | **bslbf** |
| if (flag_elevation) { | | |
| **position_elevation_difference;** | **min(num_bits,7)** | **tcimsbf** |
| } | | |
| } | | |
| if (!fixed_radius*) { | | |
| **flag_radius;** | **1** | **bslbf** |
| if (flag_radius) { | | |
| **position_radius_difference;** | **min(num_bits,5)** | **tcimsbf** |
| } | | |
| } | | |
| if (!fixed_gain*) { | | |
| **flag_gain;** | **1** | **bslbf** |
| if (flag_gain) { | | |
| **gain_factor_difference ;** | **min(num_bits,8)** | **tcimsbf** |
| } | | |
| } | | |
| if (!fixed_spread*) { | | |
| **flag_spread;** | **1** | **bslbf** |
| if (flag_spread) { | | |
| **spread_difference ;** | **min(num_bits,8)** | **tcimsbf** |
| } | | |
| } | | |
| if (hasDynamicObjectPriority) { | | |
| if (!fixed_dynamic_object_priority*) { | | |
| **flag_dynamic_object_priority;** | **1** | **bslbf** |
| if (flag_dynamic_object_priority) { | | |
| **dynamic_object_priority_difference;** | **min(num_bits,4)** | **tcimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| } | | |
| * Given by the preceding intracoded_object_metadata_low_delay() frame | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

## 7.4 Data Structure

### 7.4.1 Definition of ObjectMetadataConfig() payloads

**lowDelayMetadataCoding** indicates whether Object Metadata Decoding with Low Delay is used. Otherwise, Efficient Object Metadata are present in the bit stream.

**hasCoreLength** indicates whether the core coder works with the same number of audio samples per channel as the object metadata coder.

**OAMFrameLength** if the frame length of the core coder does not match the time resolution of the object metadata coder, the **OAMFrameLength** defines the size (in multiples of 64 audio samples) of a time slice for which a set of object metadata is available.

**hasScreenRelativeObjects** This flag specifies whether screen-relative objects are present.

**isScreenRelativeObject** This flag defines whether an object position is screen-relative. Its position shall be rendered, such that their position is remapped according to clause 18. It can still contain all valid angular values.

**hasDynamicObjectPriority** This flag indicates whether there is dynamic object priority data present in the bit stream or not. If there is no dynamic object priority data in the bit stream, mae_groupPriority is used as dynamic object priority.

num_objects Is equal to the number of objects in the associated signal group.

### 7.4.2 Efficient Object Metadata Decoding

#### 7.4.2.1 Definition of object metadata payloads

7.4.2.1.1 **Definition of object_metadata_efficient() payloads**

**has_differential_metadata** indicates whether differential object metadata are present in the bit stream.

7.4.2.1.2 **Definition of intracoded_object_metadata_efficient() payloads**

**ifperiod** defines the number of audio frames for which OAM data are processed in intracoded_object_metadata_efficient() minus 1 (iframe_period = **ifperiod** + 1).

**common_azimuth** indicates whether a common azimuth angle is used for all objects.

**default_azimuth** defines the value of the common azimuth angle.

**position_azimuth** if there is no common azimuth value, a value for each object is transmitted. If there is only one object, this is its azimuth angle.

**common_elevation** indicates whether a common elevation angle is used for all objects.

**default_elevation** defines the value of the common elevation angle.

**position_elevation** if there is no common elevation value, a value for each object is transmitted. iIf there is only one object, this is its elevation angle.

**common_radius** indicates whether a common radius value is used for all objects.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**default_radius** defines the value of the common radius.

**position_radius** if there is no common radius value, a value for each object is transmitted. If there is only one object, this is its radius.

**common_gain** indicates whether a common gain value is used for all objects.

**default_gain** defines the value of the common gain factor.

**gain_factor** if there is no common gain value, a value for each object is transmitted. If there is only one object, this is its gain factor.

**common_spread** indicates whether a common spread parameter is used for all objects.

**default_spread** defines the value of the common spread parameter.

**spread** if there is no common spread parameter, a value for each object is transmitted. If there is only one object, this is its spread parameter.

**common_dynamic_object_priority**
indicates whether a common dynamic_object_priority is used for all objects.

**default_dynamic_object_priority**
defines the value of the common dynamic_object_priority.

**dynamic_object_priority** This field defines the priority of the object.
This field can take integer values between 0 and 7. The object may be discarded from rendering and decoding if the prioity is lower than 7. If objects are discarded, the objects with lowest priority should be discarded first. If there is only one object, this is its dynamic_object_priority.

7.4.2.1.3 **Definition of differential_object_metadata() payloads**

**bits_per_point** number of bits required to represent number of polygon points (used in offset_data()).

**fixed_azimuth** flag indicating whether the azimuth value is fixed for all objects.

**flag_azimuth** flag per object indicating whether the azimuth value changes for this iframe_period.

**nbits_azimuth** how many bits are required to represent the differential value minus 2.

**differential_azimuth** value of the difference between the linearly interpolated and the actual value.

**fixed_elevation** flag indicating whether the elevation value is fixed for all objects.

**flag_elevation** flag per object indicating whether the elevation value changes for this iframe_period.

**nbits_elevation** how many bits are required to represent the differential value minus 2.

**differential_elevation** value of the difference between the linearly interpolated and the actual value.

**fixed_radius** flag indicating whether the radius is fixed for all objects.

**flag_radius** flag per object indicating whether the radius changes for this iframe_period.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**nbits_radius** how many bits are required to represent the differential value minus 2.

**differential_radius** value of the difference between the linearly interpolated and the actual value.

**fixed_gain** flag indicating whether the gain factor is fixed for all objects.

**flag_gain** flag per object indicating whether the gain radius changes for this iframe_period.

**nbits_gain** how many bits are required to represent the differential value minus 2.

**differential_gain** value of the difference between the linearly interpolated and the actual value.

**fixed_spread** flag indicating whether the spread parameter is fixed for all objects.

**flag_spread** flag per object indicating whether the spread parameter changes for this iframe_period.

**nbits_spread** how many bits are required to represent the differential value minus 2.

**differential_spread** value of the difference between the linearly interpolated and the actual value.

**fixed_dynamic_object_priority** flag indicating whether the dynamic_object_priority is fixed for all objects.

**flag_dynamic_object_priority** flag per object indicating whether the dynamic_object_priority changes for this iframe_period.

**nbits_dynamic_object_priority** how many bits are required to represent the differential value minus 2.

**differential_dynamic_object_priority**
value of the difference between the linearly interpolated and the actual value.

7.4.2.1.4 **Definition of offset_data() payloads**

**bitfield_syntax** flag indicating whether a vector with polygon time slice indices is present in the bit stream.

**offset_bitfield** bool array containing ifperiod flags for each time slice whether they are polygon points or not.

**npoints** number of polygon points minus 1 (num_points = **npoints** + 1).

**foffset** time slice indices of the polygon points within iframe_period (frame_offset = **foffset**+1).

### 7.4.2.2 Decoding process

Each audio object has associated object metadata that describe the temporal change of its position, given in spherical coordinates [azimuth, elevation, radius], and a linear gain that is applied by the object renderer. Each of these four components is generally described by a discrete time signal y(n). In the following, the discrete time index n is denoted as time slice to avoid confusion with the sample index of the audio signal that is sampled at a much greater sampling rate.

The general idea of encoding the object metadata is to segment y(n) into periods of a fixed length and to approximate these periods by means of polygons. Each transmitted frame contains at least one new polygon point, i.e. the final value. The oam decoder always stores the last polygon point of a frame and uses it as a starting point for the consecutive frame. This is important for the first frame since there is no previous value in this case.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

 © ISO/IEC 2015 – All rights reserved

Each OAM frame consists of iframe_period time slices which correspond to the audio frame length (per channel) of the core coder each (in case, **hasCoreLength** is true).

#### 7.4.2.3 Decoding the transmitted Data

First, as shown in Figure 20 by the dashed line, the values in between the last intracoded object metadata value $y_{last_iframe}$ and the value of the current intracoded object metadata $y_{current_iframe}$ are calculated for each component (azimuth, elevation, radius, and gain) using linear interpolation.

In case differential object metadata values are present in the bit stream, these are also used to calculate interpolated values. Those describe the difference between the interpolated intracoded object metadata values and the desired object metadata values. Hence, the interpolated output values $yi$ are retrieved by summing up the components of the interpolated intracoded object metadata and – if present in the bit stream - the interpolated differential object metadata for each time slice $xi$ (depicted in Figure 20 as the dotted line).

In Figure 20, there are num_points = 2 new polygon points given by differential_object_metadata() and the iframe_period is 14. The given polygon points are denoted as $y_{last_iframe} = y_0$, $y_1$, $y_2$, and $y_{current_iframe} = y_3$ which can be values for azimuth, elevation, radius, and gain. Each of these has a distinct time slice index which is denoted as $x_{last_iframe} = x_0$, $x_1$, $x_2$ and $x_{current_iframe} = x_3$. The indices of these given values are referred to as p = {0,1,2,3}.

The corresponding payload element name for time indices in case of differential object metadata is frame_offset[p] = **foffset**[p] +1 for p = {1, 2}. The indices $xi_1$ to $xi_{14}$ represent the scale of the output time axis. For each $xi$ an interpolated $yi$ has to be calculated, as shown in Figure 21.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

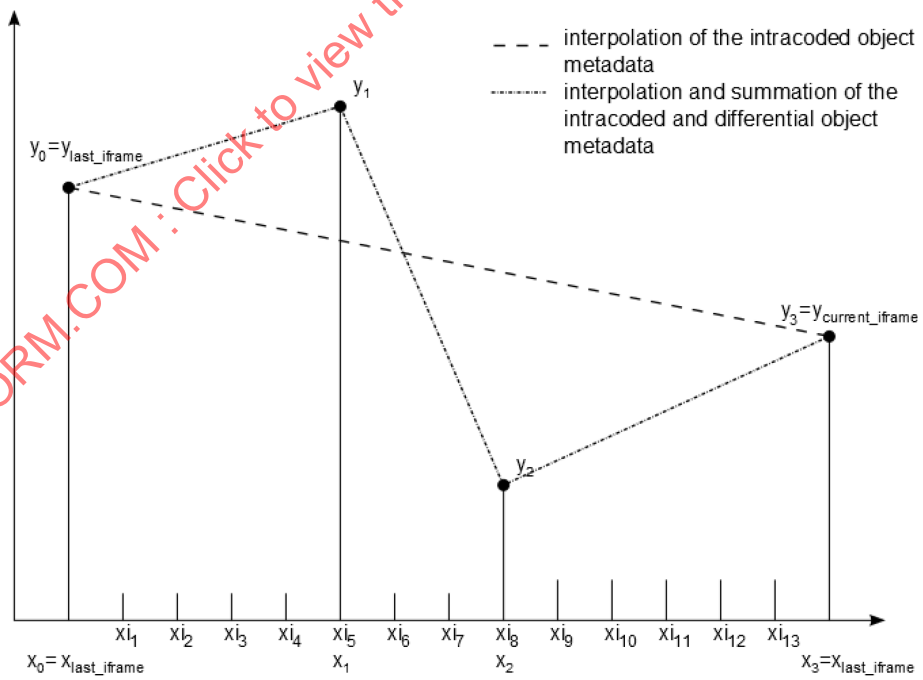



**Figure 20 — Interpolation in between the given polygon points. The dashed line depicts the first step (interpolation of intracoded object metadata). The dotted line shows the result of both interpolation steps.**

© ISO/IEC 2015 – All rights reserved

The positions (x-values) of the polygon points can be either given in form of a sequence of integer values of the size num_points, indicated by **bitfield_syntax** = 0. But they can also be given by the boolean array **offset_bitfield**, indicated by **bitfield_syntax** = 1. In the latter case, **offset_bitfield**[n] = 1 indicates the presence of a polygon point at time slice n. An **offset_bitfield** has the length of **ifperiod** time slices since the first polygon point is always known from the previous frame and polygon point for the last time slice is given by the intracoded object metadata.

The formula for the linear interpolation process is:

$$\alpha = \frac{xi_n - x_p}{x_{p+1} - x_p}$$

$$yi_n = (1-\alpha)\cdot y_p + \alpha \cdot y_{p+1}$$

where $xi_n$ is the current time index within the period $[x_p, x_{p+1}]$ that is specified by the frame_offset values. This yields the output depicted in Figure 21.

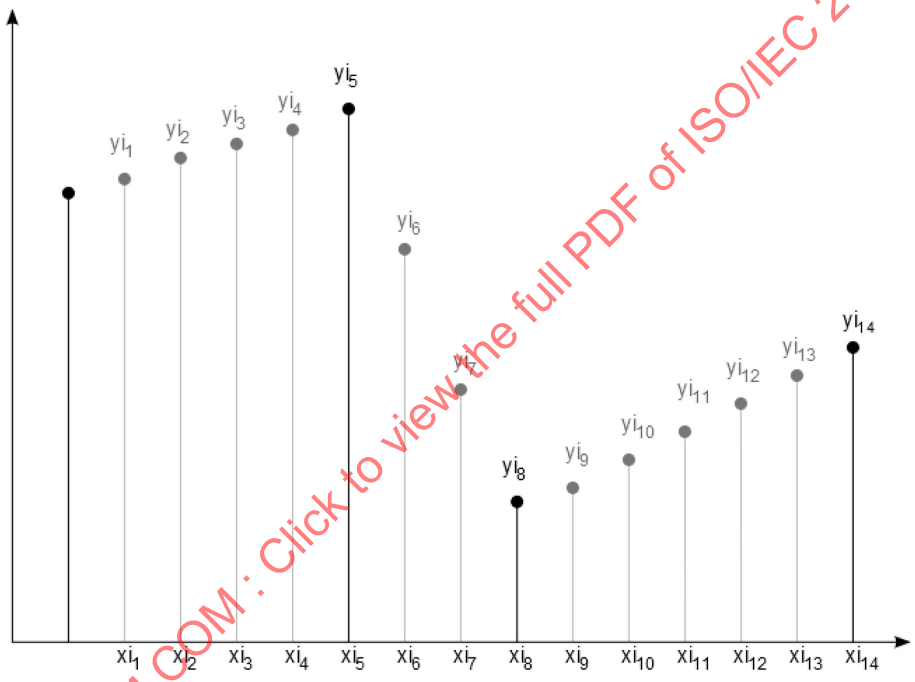



**Figure 21 — Interpolated output values yi for each time slice xi.**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

#### 7.4.2.4 Post Processing of Object Metadata

##### 7.4.2.4.1 Scaling of Object Metadata

Apply scaling factors to revert encoder scaling of the input data for each component of every object.

```
descale_multidata()
{
    for (o = 0; o < num_objects; o++)
        azimuth[o] = azimuth[o] * 1.5;

    for (o = 0; o < num_objects; o++)
        elevation[o] = elevation[o] * 3.0;

    for (o = 0; o < num_objects; o++)
```


© ISO/IEC 2015 – All rights reserved

```
        radius[o] = pow(2.0, (radius[o] / 3.0)) / 2.0;

    for (o = 0; o < num_objects; o++)
        gain[o] = pow(10.0, (gain[o] - 32.0) / 40.0);

    for (o = 0; o < num_objects; o++)
        spread[o] = spread[o] * 1.5;
}
```

#### 7.4.2.4.2 Limiting the Object Metadata

Apply limiting to the decoded values to keep the values within a valid range.

```
limit_range()
{
    minval = -180;
    maxval =  180;
    for (o = 0; o < num_objects; o++)
        azimuth[o] = MIN(MAX(azimuth[o], minval), maxval);

    minval = -90;
    maxval =  90;
    for (o = 0; o < num_objects; o++)
        elevation[o] = MIN(MAX(elevation[o], minval), maxval);

    minval =    0.5;
     maxval = 16;
    for (o = 0; o < num_objects; o++)
        radius[o] = MIN(MAX(radius[o], minval), maxval);

    minval =    0.004;
    maxval = 5.957;
    for (o = 0; o < num_objects; o++)
        gain[o] = MIN(MAX(gain[o], minval), maxval);

    minval =    0;
    maxval =  180;
    for (o = 0; o < num_objects; o++)
        spread[o] = MIN(MAX(spread[o], minval), maxval);

}
```

### 7.4.3 Object Metadata Decoding with Low Delay

#### 7.4.3.1 Definition of object metadata payloads

#### 7.4.3.1.1 Definition of object_metadata_low_delay() payloads

**has_intracoded_object_metadata** indicates whether the current frame is intracoded or differentially coded.

#### 7.4.3.1.2 Definition of intracoded_object_metadata_low_delay() payloads

**fixed_azimuth** flag indicating whether the azimuth value is fixed for all object and not transmitted in case of dynamic_object_metadata().

**default_azimuth** defines the value of the fixed or common azimuth angle.

**common_azimuth** indicates whether a common azimuth angle is used is used for all objects.

**position_azimuth** if there is no common azimuth value, a value for each object is transmitted. If there is only one object, this is its azimuth angle.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**fixed_elevation** flag indicating whether the elevation value is fixed for all object and not transmitted in case of dynamic_object_metadata().

**default_elevation** defines the value of the fixed or common elevation angle.

**common_elevation** indicates whether a common elevation angle is used for all objects.

**position_elevation** if there is no common elevation value, a value for each object is transmitted. If there is only one object, this is its elevation angle.

**fixed_radius** flag indicating whether the radius is fixed for all object and not transmitted in case of dynamic_object_metadata().

**default_radius** defines the value of the fixed or common radius.

**common_radius** indicates whether a common radius value is used for all objects.

**position_radius** if there is no common radius value, a value for each object is transmitted. If there is only one object, this is its radius.

**fixed_gain** flag indicating whether the gain factor is fixed for all object and not transmitted in case of dynamic_object_metadata().

**default_gain** defines the value of the fixed or common gain factor.

**common_gain** indicates whether a common gain value is used for all objects.

**gain_factor** if there is no common gain value, a value for each object is transmitted. If there is only one object, this is its gain factor.

**fixed_spread** flag indicating whether the spread parameter is fixed for all object and not transmitted in case of dynamic_object_metadata().

**default_spread** defines the value of the fixed or common spread parameter.

**common_spread** indicates whether a common spread parameter is used for all objects.

**spread** if there is no common spread parameter, a value for each object is transmitted. If there is only one object, this is its spread parameter.

**fixed_dynamic_object_priority** flag indicating whether the dynamic_object_priority is fixed for all objects and not transmitted in case of dynamic_object_metadata().

**default_dynamic_object_priority**
defines the value of the fixed or common dynamic_object_priority.

**common_dynamic_object_priority**
indicates whether a common dynamic_object_priority value is used for all objects.

**dynamic_object_priority** This field defines the priority of the object.
This field can take integer values between 0 and 7. The object can be discarded from rendering and decoding if the prioity is lower than 7. If objects are discarded, the objects with lowest priority should be discarded first.
If there is only one object, this is its dynamic_object_priority.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

7.4.3.1.3 **Definition of dynamic_object_metadata() payloads**

| | |
|---|---|
| **flag_absolute** | indicates whether the values of the components are transmitted differentially or in absolute values. |
| **has_object_metadata** | indicates whether there are object metadata present in the bit stream or not. |

7.4.3.1.4 **Definition of single_dynamic_object_metadata() payloads**

| | |
|---|---|
| **position_azimuth** | the absolute value of the azimuth angle if the value is not fixed. |
| **position_elevation** | the absolute value of the elevation angle if the value is not fixed. |
| **position_radius** | the absolute value of the radius if the value is not fixed. |
| **gain_factor** | the absolute value of the gain factor if the value is not fixed. |
| **spread** | the absolute value of the spread parameter if the value is not fixed. |
| **dynamic_object_priority** | the absolute value of the dynamic_object_priority if the value is not fixed. |
| **nbits** | how many bits are required to represent the differential values. |
| **flag_azimuth** | flag per object indicating whether the azimuth value changes. |
| **position_azimuth_difference** | difference between the previous and the active value. |
| **flag_elevation** | flag per object indicating whether the elevation value changes. |
| **position_elevation_difference** | value of the difference between the previous and the active value. |
| **flag_radius** | flag per object indicating whether the radius changes. |
| **position_radius_difference** | difference between the previous and the active value. |
| **flag_gain** | flag per object indicating whether the gain radius changes. |
| **gain_factor_difference** | difference between the previous and the active value. |
| **flag_spread** | flag per object indicating whether the spread parameter changes. |
| **spread_difference** | difference between the previous and the active value. |
| **flag_dynamic_object_priority** | flag per object indicating whether the dynamic_object_priority changes. |
| **dynamic_object_priority_difference** | difference between the previous and the active value. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

### 7.4.3.2 Decoding process

7.4.3.2.1 **General**

Object Metadata with Low Delay is a modified DPCM procedure. It allows switching between differentially and absolutely coded values.

© ISO/IEC 2015 – All rights reserved

#### 7.4.3.2.2 Decoding the transmitted Data

In case intracoded object metadata are present in the bit stream, the object metadata values can be read from the bit stream directly. If dynamic object metadata are in the bit stream, the object metadata values can be read from the bit stream as well when **flag_absolute** is true. Otherwise, the following equation is to be used

$$y[n] = y[n-1] + d[n]$$

where y denotes the output value, d is the difference value read from the bit stream and n is the active time slice for each component.

#### 7.4.3.2.3 Post Processing of Object Metadata

##### 7.4.3.2.3.1 Scaling of Object Metadata

Apply scaling factor to revert encoder scaling of the input data for each component of every object.

```
descale_multidata()
{
    for (o = 0; o < num_objects; o++)
        azimuth[o] = azimuth[o] * 1.5;

    for (o = 0; o < num_objects; o++)
        elevation[o] = elevation[o] * 3.0;

    for (o = 0; o < num_objects; o++)
        radius[o] = pow(2.0, (radius[o] / 3.0)) / 2.0;

    for (o = 0; o < num_objects; o++)
        gain[o] = pow(10.0, (gain[o] - 32.0) / 40.0);

    for (o = 0; o < num_objects; o++)
        spread[o] = spread[o] * 1.5;
}
```

##### 7.4.3.2.3.2 Limiting the Object Metadata

Apply limiting to the decoded values for each component of every object to keep the values within a valid range.

```
limit_range()
{
    minval = -180;
    maxval =  180;
    for (o = 0; o < num_objects; o++)
        azimuth[o] = MIN(MAX(azimuth[o], minval), maxval);

    minval =  -90;
    maxval =   90;
    for (o = 0; o < num_objects; o++)
        elevation[o] = MIN(MAX(elevation[o], minval), maxval);

    minval =    0.5;
    maxval = 16;
    for (o = 0; o < num_objects; o++)
        radius[o] = MIN(MAX(radius[o], minval), maxval);

    minval =    0.004;
    maxval = 5.957;
    for (o = 0; o < num_objects; o++)
        gain[o] = MIN(MAX(gain[o], minval), maxval);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
    minval =    0;
    maxval =  180;
    for (o = 0; o < num_objects; o++)
        spread[o] = MIN(MAX(azimuth[o], minval), maxval);

}
```

# 8 Object Rendering

## 8.1 Description

The audio object rendering is carried out as a process converting/rendering the object based audio signals into a channel based representation.

## 8.2 Terms and Definitions

| | |
|---|---|
| $\hat{\mathbf{p}}$ | Unit length vector representing the direction of the audio object. |
| $\hat{\mathbf{l}}_n$ | Unit length vector representing the direction of loudspeaker *n*. |
| $\mathbf{L}_{abc}$ | Triplet of unit length loudspeaker vectors $\hat{\mathbf{l}}_a, \hat{\mathbf{l}}_b, \hat{\mathbf{l}}_c$ in matrix form. |
| $r$ | Object position: radius in [m]. |
| $\theta$ | Object position: elevation angle in [°]. |
| $\varphi$ | Object position: azimuth angle in [°]. |
| $\alpha$ | Spread parameter in [°]. |
| $a$ | Audio object linear gain factor. |
| $\mathbf{X}$ | Matrix with audio samples of all objects (rows: objects, columns: samples). |
| $\mathbf{Y}$ | Matrix with speaker samples (rows: speakers, columns: samples). |
| Audio object | Audio + Metadata. |
| VBAP | Vector Base Amplitude Panning [1]. |

## 8.3 Input data

Audio objects consist of audio data $\mathbf{x}$ and metadata. Metadata is conveyed for every audio object at defined timestamps. The Metadata consists of the following data per audio object:

— Spherical coordinates with radius $r$ [m], elevation angle $\theta$ [°] and azimuth angle $\varphi$ [°]:

$$\mathbf{s} = \begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix}$$ (see Figure 22).

— Linear gain factor $a$.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

— Spread parameter $\alpha \in [0°, 180°]$.

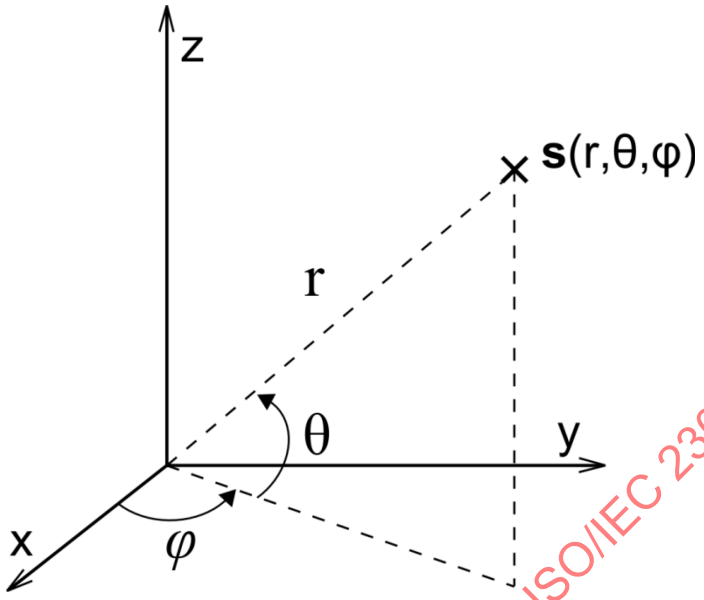



**Figure 22 — Polar coordinate system used to specify the object position**

Additionally the following definitions apply:

— Forward looking (frontal direction) is along the x-axis: $\theta = 0^\circ, \varphi = 0^\circ$.

— Azimuth ( $\varphi$ ) angle increases with counter clockwise rotation.

— $0^\circ \le \varphi \le 180^\circ$ Left hemisphere.

— $-180^\circ \le \varphi \le 0^\circ$ Right hemisphere.

— $0^\circ \le \theta \le 90^\circ$ Upper hemisphere.

— $-90^\circ \le \theta \le 0^\circ$ Lower hemisphere.

The calculations described here require Cartesian coordinates. For this purpose, the coordinates are converted according to:

$$x = r\cos(\theta)\cos(\varphi)$$
$$y = r\cos(\theta)\sin(\varphi)$$
$$z = r\sin(\theta)$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

## 8.4 Processing

### 8.4.1 Imaginary Loudspeakers

Not all considered loudspeaker setups are complete 3D setups that cover all solid angles. Hence, with regard to the triangulation that is needed for the VBAP rendering algorithm, there are voids in the triangulation surface which can be seen as invalid solid angles. These voids are filled by adding imaginary loudspeakers. The object renderer first computes the panning gains for the extended set of loudspeakers which also includes these imaginary loudspeakers (see 8.4.3). In a second step, a downmix matrix is applied to the gain vector which equally distributes the sound energy of each imaginary loudspeaker among his neighbors by applying a weighting factor of 1/sqrt(N) where N is the number of neighbors. Finally the down-mixed gain vector is power normalized.

Imaginary loudspeakers are added according to the following rules where the tolerances defined in Table 94 are taken into account to match actual speaker positions (i.e. actual azimuth, elevation angles) to the channel labels used in the following rules to define sub-sets of loudspeakers:

1. If no loudspeaker exists above 45° elevation, add an imaginary loudspeaker at [0°,90°].

2. If no loudspeaker exists below -45° elevation, add an imaginary loudspeaker at [0°,-90°].

3. If exactly one of the loudspeaker sub-sets

   - sub-set A: {CH_M_L030, CH_M_R030, CH_U_L030, CH_U_R030}

   or

   - sub-set B: {CH_M_L045, CH_M_R045, CH_U_L045, CH_U_R045}

   exists, but no other loudspeaker exists within the quadrilateral defined by the actual positions of the 4 sub-set loudspeakers, then add an imaginary loudspeaker at the mean azimuth angle of those 4 sub-set loudspeakers and the mean elevation angle of those 4 sub-set loudspeakers, where the mean angles shall be derived from the actual azimuth and elevation angles of those 4 sub-set loudspeakers.

4. If exactly one of the sub-sets

   - sub-set C: {CH_M_L110, CH_M_R110, CH_U_L110, CH_U_R110}

   or

   - sub-set D: {CH_M_L135, CH_M_R135, CH_U_L135, CH_U_R135}

   exists, but no other loudspeaker exists within the quadrilateral defined by the actual positions of the 4 sub-set loudspeakers, then add an imaginary loudspeaker at the mean azimuth angle of those 4 sub-set loudspeakers plus 180° and the mean elevation angle of those 4 sub-set loudspeakers, where the mean angles shall be derived from the actual azimuth and elevation angles of those 4 sub-set loudspeakers.

5. Sort all loudspeakers with an absolute elevation angle smaller than or equal to 45° according to their azimuth angle and fill gaps greater than 160° by the minimum number of equally spaced imaginary speakers with 0° elevation.

The neighborhood of an imaginary loudspeaker is defined by the edges of the triangulation mesh specified in 8.4.2.2. If an edge exists between a loudspeaker and the considered imaginary loudspeaker, then this loudspeaker is a neighbor.

The purpose of the downmix matrix is to eliminate the added imaginary speakers and to restrict the calculated gains to the existing speakers. If an imaginary loudspeaker is the neighbor of another imaginary loudspeaker,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

then the gains cannot be reduced to the existing loudspeakers. This problem is solved by the following approach: First, an energy distribution matrix $\boldsymbol{D}$ is constructed where the elements $d_{ij} = 1/N_i$ specifies the amount of energy which is re-distributed from the imaginary loudspeaker $i$ to loudspeaker $j$ where $N_i$ denotes the number of neighbors. Column vectors which belong to an existing loudspeaker have only one non-zero value $d_{ii} = 1$. In each iteration step we then multiply this matrix by $\boldsymbol{D}$. This corresponds to re-distributing energy portions to the neighbors according to the given weights. This is repeated until all elements which belong to the imaginary loudspeakers are less than or equal to $10^{-4}$ (≤-40dB). The element-wise square root finally yields the elements of the downmix matrix. Mathematically, the whole process is given as follows:

$$\boldsymbol{M} = \mathrm{sqrt}(\boldsymbol{D}^n)$$

where $n$ denotes the number of iterations, sqrt(•) denotes the element-wise square root, and $\boldsymbol{M}$ denotes the resulting downmix matrix.

NOTE Faster convergence is achieved by computing $\boldsymbol{D}^{(2^n)}$ instead of $\boldsymbol{D}^n$. This means that the renderer does not multiply the iteration result by $\boldsymbol{D}$, but by itself.

### 8.4.2 Dividing the Loudspeaker Setup into a Triangle Mesh

#### 8.4.2.1 General

To calculate 3D VBAP, a triangulation of the convex hull around the given loudspeaker setup is required. Within this hull loudspeaker triplets are defined that consist of three adjacent speaker positions in 3D space. Each loudspeaker triplet must fulfil the following requirements in order to generate useful results:

— The vectors from the listener position to the triangle corners must be linearly independent.

— Must not intersect with any other loudspeaker triplet.

The triangulation in 3D space can be done in several different ways and there are many ways to divide a loudspeaker setup into triangles. Please note that the additional triangles for the areas covered by the imaginary loudspeakers are defined by the imaginary loudspeaker and two successive neighbors.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

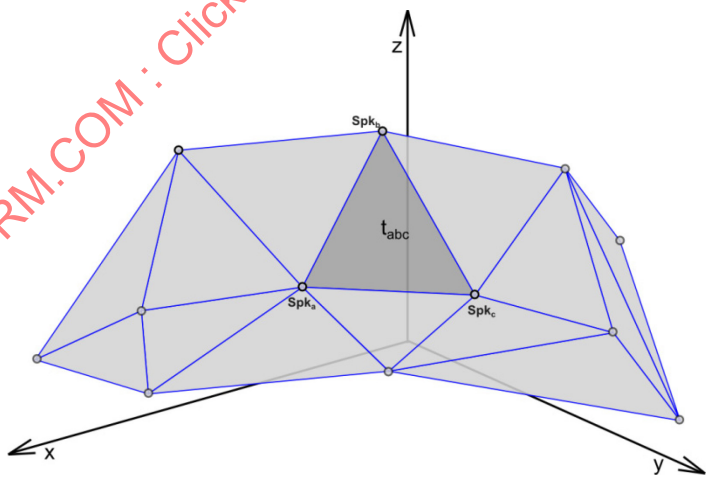



**Figure 23 —Triangulation example –**

**Triplet $t_{abc}$ defined by $Spk_a$, $Spk_b$, $Spk_c$, vectors.**


© ISO/IEC 2015 – All rights reserved

#### 8.4.2.2 Automatic Triangulation

The triangulation mesh is determined by means of a Delaunay triangulation algorithm. As all vertices, i.e. the loudspeaker positions, are located on a sphere surface, the Delaunay solution can be found by calculating the convex hull of the given vertex set. The automatic triangulation algorithm uses the QuickHull algorithm which was extended to yield left-right and front-back symmetric triangulation meshes:

1. Sort all vertices in ascending order according to the vertex index specified in the following pseudo code.

```
function index = vertex_index(azimuth, elevation)
{
    azimuth   = 180 - mod(180 - azimuth, 360);
    elevation = max(-90, min(90, elevation));
    idx_azi   = round(abs(90 - abs(azimuth)));
    idx_ele   = round(abs(elevation));
    index     = idx_azi + 181 * idx_ele;
}
```

2. Choose any sub-set of the vertices with a convex hull as initial polyhedron.

3. Extend the initial polyhedron sequentially by the sorted list of vertices. If the new vertex is located outside of the polyhedron or on its surface, replace those surface patches by the triangles that are defined by the new vertex and the border (the outer edges) of those surface patches. As a result, the most recently added vertex defines the sub-division of a planar polyeder into triangles.

### 8.4.3 Rendering Algorithm

#### 8.4.3.1 General

The rendering is based on VBAP. The algorithm and descriptions are based on [1]. As all setups are extended by imaginary loudspeakers to a complete 3D setup, 3D VBAP is used for rendering.

#### 8.4.3.2 3D VBAP

A triplet-wise panning is used for 3D setups and for all other setups, as all setups are extended by imaginary loudspeakers, if necessary. For this, the audio object is applied to a maximum of three loudspeakers. All calculations are performed for each loudspeaker triplet. The triplets are defined by the loudspeaker triangulation (see 8.4.2).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

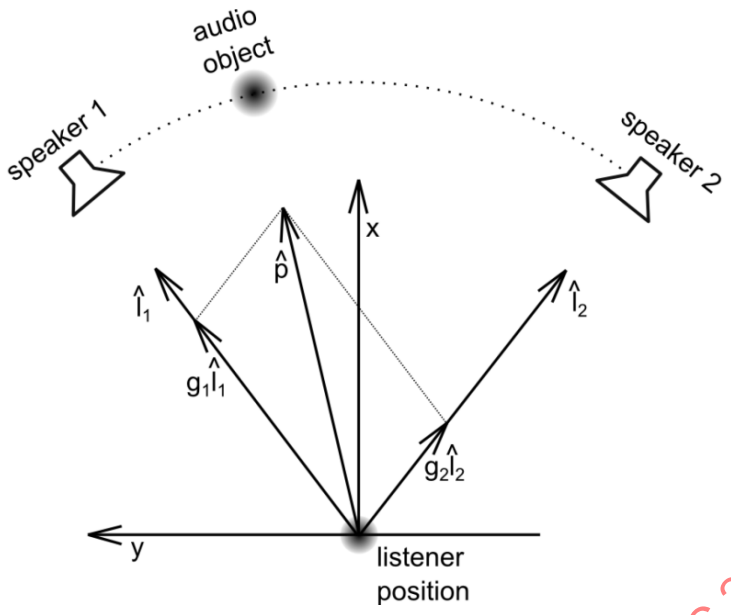



**Figure 24 — Audio object described as linear combination of speaker vectors. For the sake of simplicity, here the 2D VBAP case is shown with two instead of 3 base vectors.**

The audio object vector $\hat{\mathbf{p}}$ is expressed as a linear combination of the loudspeaker triplet $\hat{\mathbf{l}}_1, \hat{\mathbf{l}}_2, \hat{\mathbf{l}}_3$:

$$\hat{\mathbf{p}} = g_1\hat{\mathbf{l}}_1 + g_2\hat{\mathbf{l}}_2 + g_3\hat{\mathbf{l}}_3$$

In matrix form and transposed, gain factors can be calculated using the formula:

$$g = \hat{\mathbf{p}}^{\mathrm{T}}\mathbf{L}_{\mathbf{123}}^{-1} = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix}^{-1}$$

The loudspeaker triangle which contains the panning direction is found by sequentially calculating $g_1, g_2, g_3$ for the list of triplets until all gain factors are greater than or equal to zero. This set yields the vector

$$\boldsymbol{g} = [g_1, g_2, g_3].$$

Finally, the audio object's gain factor $a$ is applied and power normalization is performed:

$$\boldsymbol{g}_{\mathrm{scaled}} = \frac{a\boldsymbol{g}}{\|\boldsymbol{g}\|}$$

This is the vector which is then used to construct the rendering matrix $\boldsymbol{G}$.

#### 8.4.3.3 Audio Processing

The audio processing function transforms audio objects into channel based speaker signals.

Let $\mathbf{X}$ be the audio samples of all objects in matrix representation with dimension $O$ - number of objects, $N$ - number of audio samples per audio processing block. Each row contains the audio samples of one object.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{O,1} & x_{O,2} & \cdots & x_{O,N} \end{bmatrix}$$

Let $\mathbf{G}$ be the gain factors of all audio objects and loudspeakers in matrix representation with dimension $O$ - number of objects, $S$ - number of speakers. Each column contains all gain factors for one audio object.

$$\mathbf{G} = \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,O} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,O} \\ \vdots & \vdots & \ddots & \vdots \\ g_{S,1} & g_{S,2} & \cdots & g_{S,O} \end{bmatrix}$$

Note: There will be a maximum of three non-zero values per column.

Matrix $\mathbf{Y}$ contains the output samples that can be calculated with the formula:

$$\mathbf{Y} = \mathbf{G}\mathbf{X}$$

Each row in $\mathbf{Y}$ contains the output audio samples for one speaker.

#### 8.4.3.4 Gain Factor Crossfading

Metadata is conveyed for every audio object at defined timestamps. To get a smooth transition when metadata changes, the $\mathbf{G}$ matrices are interpolated linear between adjacent timestamps and applied on a per sample basis.

#### 8.4.3.5 Audio Processing in QMF-Domain

Alternatively, object rendering can be performed in QMF-domain using QMF-transformed audio signals. In QMF-domain, the input data is given by a three-dimensional matrix with complex elements $x^k_{o,n}$ where $o$ denotes the object index, $n$ denotes the slot index, and $k$ denotes the band index. As the VBAP panning gains are frequency independent, object rendering in QMF-domain is realized analogously to the time domain processing,

$$\boldsymbol{Y}_k = \boldsymbol{G}\boldsymbol{X}_k \ ,$$

where

$$\boldsymbol{X}_k = \begin{bmatrix} x^k_{1,1} & x^k_{1,2} & \cdots & x^k_{1,N} \\ x^k_{2,1} & x^k_{2,2} & \cdots & x^k_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x^k_{O,1} & x^k_{O,2} & \cdots & x^k_{O,N} \end{bmatrix}$$

denotes the two-dimensional sub-set of the input data matrix that belongs to QMF band $k$. The matrix

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$Y_k = \begin{bmatrix} y_{1,1}^k & y_{1,2}^k & \cdots & y_{1,N}^k \\ y_{2,1}^k & y_{2,2}^k & \cdots & y_{2,N}^k \\ \vdots & \vdots & \ddots & \vdots \\ y_{S,1}^k & y_{S,2}^k & \cdots & y_{S,N}^k \end{bmatrix}$$

denotes the two-dimensional sub-set of the output data matrix with complex elements $y_{s,n}^k$ where $s$ denotes the speaker index, $n$ denotes the slot index, and $k$ denotes the band index. The gain factor cross-fading is realized analogously to the time domain processing with the only difference that the cross-fading relates to slot indices instead of sample indices.

#### 8.4.3.6 Time-Alignment

Object rendering requires object metadata side information embedded into a MPEG-H 3D audio bitstream (see Clause 15) in combination with the audio signals which are related to each audio object. Both data streams must be time-aligned for proper rendering. This time-alignment is realized by the encoder such that the first frame of the current decoded object metadata, which specifies the metadata values over a period of *iframe_period* frames, is applied to the current audio data frame.

#### 8.4.3.7 Spreading

The object spreading specified in the following section is applied if all of the following conditions are fulfilled:

— the local loudspeaker setup is signalled in the LoudspeakerRendering(), and

— the speakerLayoutType is 0, and

— the CICPspeakerLayoutIdx being one of the following values:
4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

If at least one of the conditions is not met, the spread processing specified in the following text is not applied.

Audio objects with a spatial extent are processed by means of the Multiple Direction Amplitude Panning (MDAP) method[1]. This method involves the computation of a set of panning gains $g_{\text{scaled,m}}$ for $M = 18$ MDAP directions $p_m$ around the panning direction $p_0 = \hat{p}$.

The determination of the MDAP directions requires the computation of two base vectors, $u$ and $v$. If the loudspeaker setup contains height speakers, these vectors are computed as follows,

$$v = \begin{cases} \text{cart}(\phi, \theta + 90^\circ, 1), & \theta < 0^\circ \\ \text{cart}(\phi, \theta - 90^\circ, 1), & \theta \geq 0^\circ \end{cases}$$

$$u = v \times p_0$$

where cart(·) denotes the transform from spherical coordinates to Cartesian coordinates and $\times$ denotes the cross product. Otherwise, $u$ and $v$ are computed as follows:

$$w = \begin{cases} \text{cart}(\phi, \theta + 90^\circ, 1), & \theta < 0^\circ \\ \text{cart}(\phi, \theta - 90^\circ, 1), & \theta \geq 0^\circ \end{cases}$$

$$u = w \times p_0$$

1) Pulkki, Ville, “Generic panning tools for MAX/MSP”, Proceedings of International Computer Music Conference, 2000

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$v = [0, \ldots, 0]^T$$

The MDAP directions are then computed from 18 pattern vectors $p_m'$ :

$$\begin{aligned}
p_1' &= u \\
p_2' &= 0.75\,u + 0.25\,p_0 \\
p_3' &= 0.375\,u + 0.625\,p_0 \\
p_4' &= -u \\
p_5' &= -0.75\,u + 0.25\,p_0 \\
p_6' &= -0.375\,u + 0.625\,p_0 \\
p_7' &= 0.5\,u + 0.866\,v + p_0/3 \\
p_8' &= 0.5\,p_7' + 0.5\,p_0 \\
p_9' &= 0.25\,p_7' + 0.75\,p_0 \\
p_{10}' &= -0.5\,u + 0.866\,v + p_0/3 \\
p_{11}' &= 0.5\,p_{10}' + 0.5\,p_0 \\
p_{12}' &= 0.25\,p_{10}' + 0.75\,p_0 \\
p_{13}' &= -0.5\,u - 0.866\,v + p_0/3 \\
p_{14}' &= 0.5\,p_{13}' + 0.5\,p_0 \\
p_{15}' &= 0.25\,p_{13}' + 0.75\,p_0 \\
p_{16}' &= 0.5\,u - 0.866\,v + p_0/3 \\
p_{17}' &= 0.5\,p_{16}' + 0.5\,p_0 \\
p_{18}' &= 0.25\,p_{16}' + 0.75\,p_0
\end{aligned}$$

Together with the spread parameter $\alpha$, this yields the MDAP directions $p_m$:

$$p_m = p_m' + \frac{p_0}{\tan(\alpha')}$$

Where $\alpha'$ is equal to $\alpha$ limited to $[0.001°, 90°]$. The normalization to unit norm is not necessary since the normalization to unit norm is performed when $g_{scaled,m}$ is computed for the MDAP directions $p_m$.

If the spread parameter $\alpha$ exceeds 90°, then a cross-fading towards power-normalized unit gain $1/\sqrt{N}$ for all of the $N$ loudspeakers is applied to yield the final panning gains:

$$\lambda = \frac{\alpha - 90°}{90°}$$

$$g_{MDAP}' = (1 - \lambda)\frac{\sum_{m=0}^{M} g_{scaled,m}}{\left\|\sum_{m=0}^{M} g_{scaled,m}\right\|} + \lambda\,[1, \ldots, 1]^T/\sqrt{N}$$

$$g_{MDAP} = a\frac{g_{MDAP}'}{\|g_{MDAP}'\|}$$

Otherwise, the final panning gains are computed as follows:

$$g_{MDAP} = a\frac{\sum_{m=0}^{M} g_{scaled,m}}{\left\|\sum_{m=0}^{M} g_{scaled,m}\right\|}$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

# 9 SAOC 3D

## 9.1 Description

Spatial Audio Object Coding 3D Audio reproduction (SAOC 3D) is based on MPEG SAOC technology ISO/IEC 23003-2:2010. The SAOC 3D technology is used to enable interactive rendering functionality for audio object-based content.

## 9.2 Definitions

| | |
|---|---|
| $K$ | is the number of hybrid subbands. |
| $L$ | is the number of parameter sets. |
| $N_{\text{dec}}$ | is the number of used decorrelators. |
| $M_{\text{proc}}$ | is the number of processing bands. |
| $M_{\text{QMF}}$ | is the number of QMF subbands depending on sampling frequency. |
| $N$ | is the number of SAOC 3D input signals (channels and objects). |
| $N_{\text{ch}}$ | is the number of SAOC 3D input channels. |
| $N_{\text{obj}}$ | is the number of SAOC 3D input audio objects. |
| $N_{\text{dmx}}$ | is the number of SAOC 3D downmix signals (channel and object signals). |
| $N_{\text{ch}}^{\text{dmx}}$ | is the number of SAOC 3D downmix signals for channel inputs. |
| $N_{\text{obj}}^{\text{dmx}}$ | is the number of SAOC 3D downmix signals object inputs. |
| $N_{\text{premix}}$ | is the number of premix channels. |
| $N_{\text{out}}$ | is the number of SAOC 3D output channels. |
| $N_{\text{g}}^{q}$ | is the number of downmix signals assigned to group $\mathbf{g}_q$, defined for all group indices $q$. |
| $\mathbf{g}_q$ | is a vector with the indices of the downmix signals assigned to the same group, defined for all group indices $q$. |
| $\mathbf{R}^{l,m}$ | is the time and frequency variant rendering matrix, defined for all parameter time slots $l$ and all processing bands $m$. |
| $\mathbf{s}^{n,k}$ | is a vector with the hybrid subband (encoder) input channels, defined for all time slots $n$ and all hybrid subbands $k$. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $\mathbf{x}^{n,k}$ | is a vector with the hybrid subband (decoder) input signals (downmix), defined for all time slots $n$ and all hybrid subbands $k$. |
| $\hat{\mathbf{y}}^{n,k}$ | is a vector with the (decoder) output hybrid subband signals, which are fed into the hybrid synthesis filter banks, defined for all time slots $n$ and all hybrid subbands $k$. |
| $\mathbf{y}_{\text{dry}}^{n,k}$ | is a vector with the parametrically estimated hybrid subband signals, defined for all time slots $n$ and all hybrid subbands $k$. |
| $\mathbf{y}_{\text{wet}}^{n,k}$ | is a vector with the decorrelated hybrid subband signals, defined for all time slots $n$ and all hybrid subbands $k$. |
| $\mathbf{D}$ | is the downmixing matrix. |
| $\mathbf{D}_{\text{DMG}}$ | is the three dimensional matrix holding the dequantized, and mapped DMG data for every input signal, downmix channel, and parameter set. |
| $\mathbf{D}_{\text{IOC}}$ | is the four dimensional matrix holding the dequantized, and mapped IOC data for every input channel pair, every parameter set, and $M_{proc}$ bands. |
| $\mathbf{D}_{\text{OLD}}$ | is the three dimensional matrix holding the dequantized, and mapped OLD data for every input channel, every parameter set, and $M_{proc}$ bands. |
| $\mathbf{m} = diag(\mathbf{M})$ | is the main diagonal of matrix $\mathbf{M}$. |
| $\mathbf{H} = matdiag(\mathbf{M})$ | is a matrix containing the elements from the main diagonal of matrix $\mathbf{M}$ on the main diagonal and zero values on the off-diagonal positions. |
| $\varepsilon$ | is a constant used to avoid division by zero $\varepsilon = 10^{-9}$. |
| **SAOC 3D** | Spatial Audio Object Coding for 3D Audio reproduction. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

## 9.3 Delay and synchronization

The SAOC 3D decoder introduces a delay when processing the time domain signal coming from a downmix decoder. The transmission of the SAOC 3D side information with respect to the transmission of the coded downmix signal is done in such a manner that there is no need to further delay the downmix signal before the SAOC processing.

## 9.4 Syntax

### 9.4.1 Payloads for SAOC 3D

**Table 79 — Syntax of SAOC3DSpecificConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| SAOC3DSpecificConfig()<br>{ | | |

© ISO/IEC 2015 – All rights reserved

| | | | |
|---|---|---|---|
| **bsSamplingFrequencyIndex**; | **4** | **uimsbf** |
| if ( bsSamplingFrequencyIndex == 15 ) { | | |
| **bsSamplingFrequency**; | **24** | **uimsbf** |
| } | | |
| **bsFreqRes**; | **3** | **uimsbf** |
| **bsDoubleFrameLengthFlag**; | **1** | **uimsbf** |
| **bsNumSaocDmxChannels**; | **5** | **uimsbf** |
| **bsNumSaocDmxObjects**; | **5** | **uimsbf** |
| **bsDecorrelationMethod;** | **1** | **uimsbf** |
| NumInputSignals = 0; | | |
| if (bsNumSaocDmxChannels > 0) { | | |
| saocChannelLayout = SpeakerConfig3d(); | | |
| NumSaocChannels = SAOC3DgetNumChannels(saocChannelLayout); | | Note 1 |
| NumInputSignals += NumSaocChannels**;** | | |
| } | | |
| **bsNumSaocObjects**; | **8** | **uimsbf** |
| NumInputSignals += bsNumSaocObjects**;** | | |
| for ( i=0; i<NumSaocChannels; i++ ) { | | |
| bsRelatedTo[i][i] = 1; | | |
| for( j=i+1; j< NumSaocChannels; j++ ) { | | |
| **bsRelatedTo**[i][j]; | **1** | **uimsbf** |
| bsRelatedTo[j][i] = bsRelatedTo[i][j]; | | |
| } | | |
| } | | |
| for ( i=NumSaocChannels; i<NumInputSignals; i++ ) { | | |
| for( j=0; j<NumSaocChannels; j++ ) { | | |
| bsRelatedTo[i][j] = 0; | | |
| bsRelatedTo[j][i] = 0; | | |
| } | | |
| } | | |
| for ( i=NumSaocChannels; i<NumInputSignals; i++ ) { | | |
| bsRelatedTo[i][i] = 1; | | |
| for( j=i+1; j<NumInputSignals; j++ ) { | | |
| **bsRelatedTo**[i][j]; | **1** | **uimsbf** |
| bsRelatedTo[j][i] = bsRelatedTo[i][j]; | | |
| } | | |
| } | | |
| **bsOneIOC;** | **1** | **uimsbf** |
| **bsSaocDmxMethod;** | **1** | **uimsbf** |
| if (bsSaocDmxMethod == 1) { | | |
| NumPremixedChannels = SAOC3DgetNumChannels(referenceLayout); | | Note 1,2 |
| } | | |
| **bsDualMode**; | **1** | **uimsbf** |
| if (bsDualMode) { | | |
| **bsBandsLow;** | **5** | **uimsbf** |
| bsBandsHigh = numBands; | | Note 3 |
| } else { | | |
| bsBandsLow = numBands; | | |
| } | | |
| **bsDcuFlag;** | **1** | **uimsbf** |
| if ( bsDcuFlag == 1 ) { | | |
| **bsDcuMandatory;** | **1** | **uimsbf** |
| **bsDcuDynamic;** | **1** | **uimsbf** |
| if ( bsDcuDynamic == 0 ) { | | |
| **bsDcuMode;** | **1** | **uimsbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| **bsDcuParam;** | **4** | **uimsbf** |
| **}** | | |
| } else { | | |
| bsDcuMandatory = 0; | | |
| bsDcuDynamic = 0; | | |
| bsDcuMode = 0; | | |
| bsDcuParam = 0; | | |
| } | | |
| ByteAlign(); | | |
| SAOC3DExtensionConfig(); | | |
| } | | |

Note 1: SAOC3DgetNumChannels() defines the number of SAOC 3D input channels from data obtained by the bitstream syntax element SpeakerConfig3d().
Note 2: referenceLayout is defined in 5.3.2.
Note 3: numBands is defined in Table 33 in ISO/IEC 23003-2:2010.

**Table 80 — Syntax of SAOC3DgetNumChannels()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| SAOC3DgetNumChannels(Layout) | | Note 1 |
| { | | |
| numChannels = numSpeakers; | | Note 2 |
| for (i = 0; i < numSpeakers; i++) { | | |
| if (Layout.isLFE[i] == 1) { | | |
| numChannels = numChannels - 1; | | |
| } | | |
| } | | |
| return numChannels; | | |
| } | | |

Note 1: The function SAOC3DgetNumChannels() returns the number of available non-LFE channels numChannels.
Note 2: numSpeakers is defined in Syntax of SpeakerConfig3d(). If speakerLayoutType == 0 numSpeakers represents the number of loudspeakers corresponding to the ChannelConfiguration value, CICPspeakerLayoutIdx, as defined in ISO/IEC 23001-8.

**Table 81 — Syntax of Saoc3DFrame()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| Saoc3DFrame() | | |
| { | | |
| SAOC3DFramingInfo(); | | |
| **bsIndependencyFlag**; | **1** | **uimsbf** |
| for( i=0; i<NumInputSignals; i++ ) { | | |
| idxOLD[i] = EcDataSaoc(OLD, i, numBands); | | |
| } | | |
| k=0; | | |
| iocIdx1=0; | | |
| iocIdx2=0; | | |
| for( i=0; i<NumInputSignals; i++ ) { | | |
| idxIOC[i][i] = 0; | | |
| for( j=i+1; j<NumInputSignals; j++ ) { | | |
| if ( bsRelatedTo[i][j] != 0 ) { | | |
| if ( bsOneIOC == 0 ) { | | |
| idxIOC[i][j] = EcDataSaoc(IOC, k, numBands); | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
                    k++;
                } else {
                    if ( k == 0 )
                        idxIOC[i][j] = EcDataSaoc(IOC, k, numBands);
                        k++;
                        iocIdx1=i;
                        iocIdx2=j;
                    } else {
                        idxIOC[i][j] = idxIOC[iocIdx1][iocIdx2];
                    }
                }
            } else {
                idxIOC[i][j] = 5;
            }
        idxIOC[j][i] = idxIOC[i][j];
        }
    }
    if (bsNumSaocDmxObjects==0)  {
        for( i=0; i< bsNumSaocDmxChannels; i++ ) {
            idxDMG[i] = EcDataSaoc(DMG, 0,  NumInputSignals);
        }
    }  else {
        dmgIdx = 0;
        for( i=0; i<bsNumSaocDmxChannels; i++ ) {
            idxDMG[i] = EcDataSaoc(DMG, 0, NumSaocChannels);
        }
        dmgIdx = bsNumSaocDmxChannels;
        if (bsSaocDmxMethod == 0) {
            for( i=dmgIdx; i<dmgIdx + bsNumSaocDmxObjects; i++ ) {
                idxDMG[i] = EcDataSaoc(DMG, 0,  bsNumSaocObjects);
            }
        }  else {
            for( i=dmgIdx; i<dmgIdx + bsNumSaocDmxObjects; i++ ) {
                idxDMG[i] = EcDataSaoc(DMG, 0, NumPremixedChannels);
            }
        }
    }
    if ( bsDcuFlag == 1 ) && ( bsDcuDynamic == 1 ) {
        if ( bsIndependencyFlag == 1 ) {
            bsDcuDynamicUpdate = 1;
        } else {
            bsDcuDynamicUpdate;                                  1          uimsbf
        }
        if ( bsDcuDynamicUpdate == 1 ) {
            bsDcuMode;                                           1          uimsbf
            bsDcuParam;                                          4          uimsbf
        }
    }
    ByteAlign();
    SAOC3DExtensionFrame();
}
```

Note 1: numBands is defined in Table 33 in ISO/IEC 23003-2:2010.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 82 — Syntax of SAOC3DFramingInfo()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| SAOC3DFramingInfo()<br>{<br>**bsFramingType**;<br>**bsNumParamSets**;<br>if (bsFramingType) {<br>for (ps=0; ps<numParamSets; ps++) {<br>**bsParamSlot**[ps];<br>}<br>}<br>} | <br><br>**1**<br>**3**<br><br><br>**nBitsParamSlot** | <br><br>**uimsbf**<br>**uimsbf**<br><br>Note 1<br>**uimsbf**<br>Note 2 |
| Note 1: numParamSets is defined by numParamSets = bsNumParamSets + 1.<br>Note 2: nBitsParamSlot is defined according to nBitsParamSlot = ceil(log2(numSlots)). | | |

### 9.4.2 Definition of SAOC 3D payloads

The following tables contain definitions of used SAOC 3D bitstream syntactic elements and variables:

**Table 83 — Definitions of SAOC 3D syntactic elements**

| SAOC 3D syntactic elements | Definition*) | Reference |
|---|---|---|
| EcDataSaoc() | EcDataSaoc() | Table 22 |
| SAOC3DExtensionConfig() | SAOCExtensionConfig() | Table 6 |
| SAOC3DExtensionConfigData(bsSaoc3DExtType) | N/A | |
| SAOC3DExtensionFrame() | SAOCExtensionFrame() | Table 27 |
| SAOC3DExtensionFrameData(bsSaoc3DExtType) | N/A | |
| ByteAlign() | ByteAlign() | |
| *) defined in ISO/IEC 23003-2:2010 | | |

**Table 84 — Definitions of SAOC 3D bitstream variables**

| SAOC 3D bitstream variables | Definition*) |
|---|---|
| bsSamplingFrequencyIndex | bsSamplingFrequencyIndex |
| bsSamplingFrequency | bsSamplingFrequency |
| bsDcuFlag | bsDcuFlag |
| bsOneIOC | bsOneIOC |
| bsDcuMode | bsDcuMode |
| bsDcuMandatory | bsDcuMandatory |
| bsDcuDynamic | bsDcuDynamic |
| bsDcuDynamicUpdate | bsDcuDynamicUpdate |
| bsDcuParam | bsDcuParam |
| bsIndependencyFlag | bsIndependencyFlag |
| bsRelatedTo | bsRelatedTo |
| *) defined in ISO/IEC 23003-2:2010 | |

saocChannelLayout — Defines the input channel layout for which SAOC 3D parameters are transmitted.

NumSaocChannels — Defines the number of input channels for which SAOC 3D parameters are transmitted.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**bsNumSaocObjects** Defines the number of input objects for which SAOC 3D parameters are transmitted.

**bsDoubleFrameLengthFlag**

Indicates whether the SAOC 3D frame length is equal to or double of the core coder output frame length. The number of time slots is given by:

numSlots = ((bsDoubleFrameLengthFlag + 1) · outputFrameLength) / 64,

where outputFrameLength is defined in ISO/IEC 23003-3:2012, 6.1, Table 70.

**bsNumSaocDmxChannels**

Defines the number of SAOC 3D downmix channels for channel based content.

**bsNumSaocDmxObjects**

Defines the number of SAOC 3D downmix channels for object based content according to:

**Table 85 — Definition of decoding mode**

| bsNumSaocDmxObjects | Mode | Meaning |
|---|---|---|
| 0 | "Combined" | All input signals are combined into $N_{ch}$ channels |
| 1 … 31 | "Independent" | The channel-based and object-based signals are downmixed independently into $N_{ch}$ and $N_{obj}$ channels |

**bsSaocDmxMethod** Defines the downmix matrix mode and number of premixing channels according to:

**Table 86 — bsSaocDmxMethod**

| bsSaocDmxMethod | Mode | Meaning |
|---|---|---|
| 0 | "Direct" | Downmix matrix is defined directly by DMGs. |
| 1 | "Premixing" | Downmix matrix is defined as a product of the matrix obtained from the dequantized DMGs and a premixing matrix obtained from the spatial information of the input audio objects and the reference layout. The Premixing mode can be used only if the reference layout is defined in mpegh3daConfig(). |

**NumPremixedChannels**

Defines the number of premixing channels ( $N_{premix}$ ) for the input audio objects.

**bsDecorrelationMethod**

Defines the decorrelation method according to:

**Table 87 — bsDecorrelationMethod**

| bsDecorrelationMethod | Meaning |
|---|---|
| 0 | "Energy compensation method" |
| 1 | "Covariance adjustment method" |

**bsDualMode** Indicates if decoding operates in different modes for a low and high band range according to:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 88 — bsDualMode**

| bsDualMode | Meaning |
|---|---|
| 0 | Same decoding mode for full band range (i.e., no separate high band range) |
| 1 | Different decoding modes for low and high band ranges |

**bsBandsLow** Defines the number of parameter bands for which decoding should be processed according to prediction based scheme.

Prediction based scheme should be used for the parameter band range: 0 <= pb < bsBandsLow.

Energy based scheme should be used for the parameter band range: bsBandsLow <= pb < numBands.

**bsSaocExtType** Indicates type of the SAOC 3D extension data according to:

**Table 89 — bsSaoc3DExtType**

| bsSaoc3DExtType | SAOC3DExtensionFrameData() |
|---|---|
| 0 ... 7 | present |
| 8 ... 15 | not present |

## 9.5 SAOC 3D processing

### 9.5.1 Compressed data stream decoding and dequantization of SAOC 3D data

The process for dequantization of the DMG, OLD, IOC parameters is specified in ISO/IEC 23003-2:2010.

### 9.5.2 Time/frequency tranforms

The hybrid filterbank specified in ISO/IEC 23003-1:2007 is applied.

### 9.5.3 Signals and parameters

#### 9.5.3.1 Dimensionality of signals and parameters

The audio signals are defined for every time slot $n$ and every hybrid subband $k$. The corresponding SAOC 3D parameters are defined for each parameter time slot $l$ and processing band $m$. The subsequent mapping between the hybrid and parameter domain is specified by Table A.31 of ISO/IEC 23003-1:2007. Hence, all calculations are performed with respect to the certain time/band indices and the corresponding dimensionalities are implied for each introduced variable.

The data available at the SAOC 3D decoder consists of the multi-channel downmix signal $\mathbf{X}$, the covariance matrix $\mathbf{E}$, the rendering matrix $\mathbf{R}$ and downmix matrix $\mathbf{D}$.

#### 9.5.3.2 Object parameters

The covariance matrix $\mathbf{E}$ of size $N \times N$ with elements $e_{i,j}$ represents an approximation of the original signal covariance matrix $\mathbf{E} \approx \mathbf{SS}^*$ and is obtained from the OLD and IOC parameters as

$$e_{i,j} = \sqrt{OLD_i OLD_j}\, IOC_{i,j}\,.$$

Here, the dequantized object parameters are obtained as

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$OLD_i = \mathbf{D}_{\mathrm{OLD}}(i,l,m), \quad IOC_{i,j} = \mathbf{D}_{\mathrm{IOC}}(i,j,l,m).$$

#### 9.5.3.3 Downmix matrix

The downmix matrix $\mathbf{D}$ applied to the input audio signals $\mathbf{S}$ determines the downmix signal as $\mathbf{X} = \mathbf{DS}$. The downmix matrix $\mathbf{D}$ of size $N_{\mathrm{dmx}} \times N$ is obtained as:

$$\mathbf{D} = \mathbf{D}_{\mathrm{dmx}} \mathbf{D}_{\mathrm{premix}}.$$

- "Direct mode" (bsSaocDmxMethod == 0):

The matrix $\mathbf{D}_{\mathrm{premix}}$ of size $N \times N$ is defined as:

$$\mathbf{D}_{\mathrm{premix}} = \mathbf{I}.$$

- "Premixing mode" (bsSaocDmxMethod == 1):

The matrix $\mathbf{D}_{\mathrm{premix}}$ of size $(N_{\mathrm{ch}} + N_{\mathrm{premix}}) \times N$ is defined as:

$$\mathbf{D}_{\mathrm{premix}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}.$$

The premixing matrix $\mathbf{A}$ of size $N_{\mathrm{premix}} \times N_{\mathrm{obj}}$ is received as an input to the SAOC 3D decoder from the object renderer as a function of decoded object metadata and channel configuration of the reference output format (number and geometric positions of premixed channels, "referenceLayout").

The matrix $\mathbf{D}_{\mathrm{dmx}}$ of size $N_{\mathrm{dmx}} \times \bar{N}$ is obtained from the DMG parameters as:

— "Combined decoding mode" (bsNumSaocDmxObjects == 0):

$$d_{i,j} = 10^{0.05 DMG_{i,j}},$$

— "Independent decoding mode" (bsNumSaocDmxObjects >= 1):

$$d_{i,j} = \begin{cases} 10^{0.05 DMG_{i,j}}, & 0 \le i < N_{\mathrm{ch}}^{\mathrm{dmx}}, 0 \le j < N_{\mathrm{ch}} \\ 10^{0.05 DMG_{i,j-N_{\mathrm{ch}}}}, & N_{\mathrm{ch}}^{\mathrm{dmx}} \le i < N_{\mathrm{dmx}}, N_{\mathrm{ch}} \le j < \bar{N} \\ 0, & otherwise \end{cases},$$

where $\bar{N} = N$ for the "direct mode" or $\bar{N} = N_{\mathrm{ch}} + N_{\mathrm{premix}}$ for the "premixing mode".

Here, the dequantized downmix parameters are obtained as:

$$DMG_{i,j} = \mathbf{D}_{\mathrm{DMG}}(i,j,l).$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 9.5.3.4 Rendering matrix

The rendering matrix $\mathbf{R}$ applied to the input audio signals $\mathbf{S}$ determines the target rendered output $\hat{\mathbf{Y}} = \mathbf{RS}$.

The rendering matrix $\mathbf{R}$ of size $N_{\text{out}} \times N$ is given by:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{\text{ch}} & \mathbf{R}_{\text{obj}} \end{bmatrix},$$

where the matrix $\mathbf{R}_{\text{ch}}$ of size $N_{\text{out}} \times N_{\text{ch}}$ is associated with input channels and matrix $\mathbf{R}_{\text{obj}}$ of size $N_{\text{out}} \times N_{\text{obj}}$ is associated with input objects.

The rendering matrix $\mathbf{R}_{\text{ch}}$ is received as an input to the SAOC 3D decoder from the format converter as a function of the: channel configuration of the channels for which SAOC 3D parameters are transmitted (number and geometric positions of SAOC 3D input channels, saocChannelLayout) and the reproduction layout which is received as input to the SAOC 3D decoder:

$$\mathbf{R}_{\text{ch}} = \mathbf{M}_{\text{DMX}},$$

where matrix $\mathbf{M}_{\text{DMX}}$ is defined in 10.2.2. The downmix matrix $\mathbf{M}_{\text{DMX}}$ is different to the one which is used for regular audio channels, but is just computed in the same way by the format converter

The rendering matrix $\mathbf{R}_{\text{obj}}$ is received as an input to the SAOC 3D decoder from the object renderer as a function of decoded object metadata and the reproduction layout which is received as input to the SAOC 3D decoder:

$$\mathbf{R}_{\text{obj}} = \mathbf{G},$$

where matrix $\mathbf{G}$ is defined in 8.4.3.3 for the specified timestamps. The rendering matrix $\mathbf{R}_{\text{obj}}$ is determined for each parameter time-slot $l$ by linear interpolation of the gain factors $\mathbf{G}$, between the timestamps preceding and following the parameter time-slot $l$.

#### 9.5.3.5 Target output covariance matrix

The covariance matrix $\mathbf{C}$ of size $N_{\text{out}} \times N_{\text{out}}$ representing an approximation of the target output signal covariance $\mathbf{C} \approx \mathbf{YY}^*$ is obtained as:

$$\mathbf{C} = \mathbf{RER}^*.$$

### 9.5.4 SAOC 3D decoding

#### 9.5.4.1 Overview

The method for obtaining an output signal using SAOC 3D parameters and rendering information is described. The basic structure of the SAOC 3D decoder consisting of the parameter processor and the downmix processor is depicted in Figure 25.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

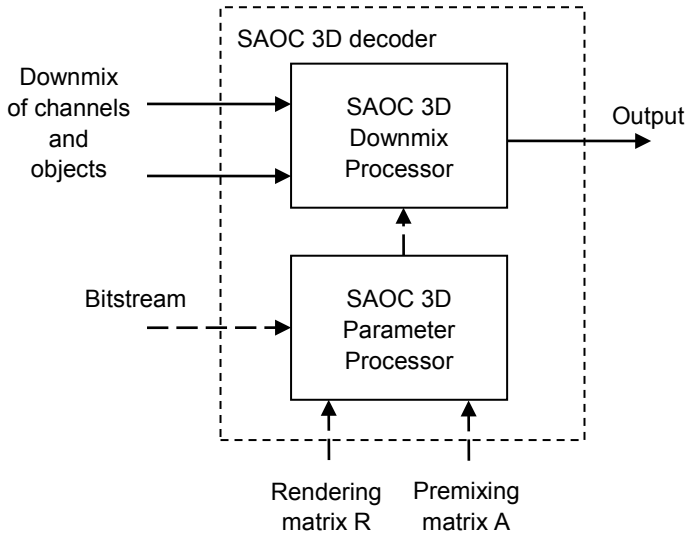



**Figure 25 — Structure of the SAOC 3D decoder**

### 9.5.4.2 SAOC 3D downmix processor

#### 9.5.4.2.1 General

The detailed structure of the SAOC 3D downmix processor is depicted in Figure 26:

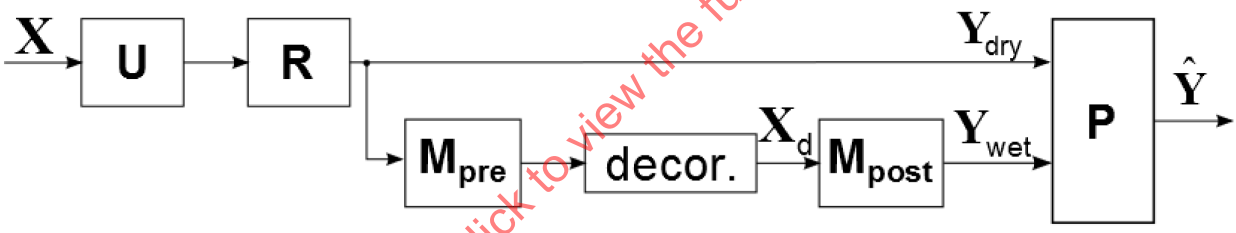



**Figure 26 — Structure of the downmix processor**

The output signal $\hat{\mathbf{Y}}$ is computed from the signals $\mathbf{Y}_{dry}$ and $\mathbf{Y}_{wet}$ as:

$$\hat{\mathbf{Y}} = \mathbf{P}\begin{bmatrix}\mathbf{Y}_{dry}\\ \mathbf{Y}_{wet}\end{bmatrix}.$$

The signals $\mathbf{Y}_{dry}$ and $\mathbf{Y}_{wet}$ are calculated from the downmix signal $\mathbf{X}$ and the decorrelated signal $\mathbf{X}_d$ as:

$$\mathbf{Y}_{dry} = \mathbf{RUX},$$

$$\mathbf{Y}_{wet} = \mathbf{M}_{post}\mathbf{X}_d.$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

The matrix product $(\mathbf{RU})^{l,k}$ and the matrix $(\mathbf{P})^{l,k}$, computed for every parameter time-slot $l$, are interpolated over all time-slots $n$ as described in 6.5.2.1 of ISO/IEC 23003-1:2007, considering:

$\mathbf{W}_1^{l,k} = (\mathbf{RU})^{l,k}$, for computing $(\mathbf{RU})^{n,k} = \mathbf{M}_1^{n,k}$,

and

$\mathbf{W}_1^{l,k} = \mathbf{P}^{l,k}$, for computing $(\mathbf{P})^{n,k} = \mathbf{M}_1^{n,k}$,

where $\mathbf{W}_1^{l,k}$ and $\mathbf{M}_1^{n,k}$ are variables specified in ISO/IEC 23003-1:2007.

#### 9.5.4.2.2 Parametric unmixing matrix

The parametric unmixing matrix $\mathbf{U}$ is obtained according to the "decoding mode" as:

- "Combined decoding mode" (bsNumSaocDmxObjects == 0):

$\mathbf{U} = \mathbf{E}\mathbf{D}^*\mathbf{J}$ .

The matrix $\mathbf{J} \approx \mathbf{\Delta}^{-1}$ of size $N_{dmx} \times N_{dmx}$ for $\mathbf{\Delta} = \mathbf{D}\mathbf{E}\mathbf{D}^*$ is derived according to 9.5.4.2.5.

- "Independent decoding mode" (bsNumSaocDmxObjects >= 1):

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{\text{ch}} & 0 \\ 0 & \mathbf{U}_{\text{obj}} \end{bmatrix}, \quad \mathbf{U}_{\text{ch}} = \mathbf{E}_{\text{ch}}\mathbf{D}_{\text{ch}}^*\mathbf{J}_{\text{ch}}, \quad \mathbf{U}_{\text{obj}} = \mathbf{E}_{\text{obj}}\mathbf{D}_{\text{obj}}^*\mathbf{J}_{\text{obj}}.$$

The channel based covariance matrix $\mathbf{E}_{ch}$ of size $N_{ch} \times N_{ch}$ and the object based covariance matrix $\mathbf{E}_{\text{obj}}$ of size $N_{\text{obj}} \times N_{\text{obj}}$ are obtained from the covariance matrix $\mathbf{E}$ by selecting only the corresponding main diagonal blocks:

$$\mathbf{E} = \begin{bmatrix} \mathbf{E}_{\text{ch}} & \mathbf{E}_{\text{ch,obj}} \\ \mathbf{E}_{\text{obj,ch}} & \mathbf{E}_{\text{obj}} \end{bmatrix}.$$

The object-channel cross-covariance matrix $\mathbf{E}_{\text{ch,obj}} = \mathbf{E}_{\text{obj,ch}}^*$ is not used in the computations.

The channel based downmix matrix $\mathbf{D}_{\text{ch}}$ of size $N_{\text{ch}}^{\text{dmx}} \times N_{\text{ch}}$ and the object based downmix matrix $\mathbf{D}_{\text{obj}}$ of size $N_{\text{obj}}^{\text{dmx}} \times N_{\text{obj}}$ are obtained from the downmix matrix $\mathbf{D}$ by selecting only the corresponding main diagonal blocks:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_{\text{ch}} & 0 \\ 0 & \mathbf{D}_{\text{obj}} \end{bmatrix}.$$

The matrices $\mathbf{J}_{\text{ch}} \approx \mathbf{\Delta}^{-1}$ of size $N_{\text{ch}}^{\text{dmx}} \times N_{\text{ch}}^{\text{dmx}}$ for $\mathbf{\Delta} = \mathbf{D}_{\text{ch}}\mathbf{E}_{\text{ch}}\mathbf{D}_{\text{ch}}^*$ and $\mathbf{J}_{\text{obj}} \approx \mathbf{\Delta}^{-1}$ of size $N_{\text{obj}}^{\text{dmx}} \times N_{\text{obj}}^{\text{dmx}}$ for $\mathbf{\Delta} = \mathbf{D}_{\text{obj}}\mathbf{E}_{\text{obj}}\mathbf{D}_{\text{obj}}^*$ are derived according to 9.5.4.2.5.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

#### 9.5.4.2.3 Decorrelation

The pre-processing matrix $\mathbf{M}_{\text{pre}}$ is defined for different output configurations in Annex B.

The post-processing matrix $\mathbf{M}_{\text{post}}$ is obtained as:

$$\mathbf{M}_{\text{post}} = \mathbf{M}_{\text{pre}}^{*}\mathbf{J}_{\text{pre}}.$$

The matrix $\mathbf{J}_{\text{pre}} \approx \left(\mathbf{M}_{\text{pre}}\mathbf{M}_{\text{pre}}^{*}\right)^{-1}$ is derived according to 9.5.4.2.5.

The decorrelated signals $\mathbf{X}_{\text{d}}$ are created from the decorrelator specified in 6.6.2 of ISO/IEC 23003-1:2007, with bsDecorrConfig == 0 and a decorrelator index $X$ according to Table B.1 and Table B.2 in Annex B. Hence, the function $decorrFunc(\cdot)$ denotes the decorrelation process:

$$\mathbf{X}_{\text{d}} = decorrFunc\left(\mathbf{M}_{\text{pre}}\mathbf{Y}_{\text{dry}}\right).$$

#### 9.5.4.2.4 Mixing matrix $\mathbf{P}$

The following covariance matrices notation is introduced:

- The matrix $\mathbf{E}_{\mathbf{Y}}^{\text{dry}}$ representing the covariance of the parametrically estimated signal $\mathbf{E}_{\mathbf{Y}}^{\text{dry}} \approx \mathbf{Y}_{\text{dry}}\mathbf{Y}_{\text{dry}}^{*}$ and defined as:

$$\mathbf{E}_{\mathbf{Y}}^{\text{dry}} = \mathbf{RU}\left(\mathbf{DED}^{*}\right)\mathbf{U}^{*}\mathbf{R}^{*}.$$

- The matrix $\mathbf{E}_{\mathbf{Y}}^{\text{wet}}$ representing the covariance matrix of the decorrelated signal $\mathbf{E}_{\mathbf{Y}}^{\text{wet}} \approx \mathbf{Y}_{\text{wet}}\mathbf{Y}_{\text{wet}}^{*}$ and defined as:

$$\mathbf{E}_{\mathbf{Y}}^{\text{wet}} = \mathbf{M}_{\text{post}}\left[matdiag(\mathbf{M}_{\text{pre}}\mathbf{E}_{\mathbf{Y}}^{\text{dry}}\mathbf{M}_{\text{pre}}^{*})\right]\mathbf{M}_{\text{post}}^{*}.$$

- The matrix $\mathbf{\Delta}_{E}$ describing difference between the target output covariance and the covariance of the parametrically estimated signals and computed as:

$$\mathbf{\Delta}_{E} = \mathbf{C} - \mathbf{E}_{\mathbf{Y}}^{\text{dry}}.$$

The mixing matrix $\mathbf{P}$ of size $N_{\text{out}} \times 2N_{\text{out}}$ is given by:

$$\mathbf{P} = \left[\mathbf{P}_{\text{dry}} \quad \mathbf{A}_{\text{wet}}\mathbf{P}_{\text{wet}}\right].$$

The limitation matrix $\mathbf{A}_{\text{wet}}$ of size $N_{\text{out}} \times N_{\text{out}}$ is given by:

$$\mathbf{A}_{\text{wet}} = \text{matdiag}\left(\min\left(1, \sqrt{\max\left(0, \lambda_{Dec}\frac{\mathbf{E}_{\mathbf{Y}}^{\text{dry}}(i,i)}{\max\left(\varepsilon, \hat{\mathbf{E}}_{\mathbf{Y}}^{\text{wet}}(i,i)\right)}\right)}\right)\right),$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

where $\lambda_{Dec} = 4$ is a constant used to limit the amount of decorrelated component added to the output signals. The matrix $\hat{\mathbf{E}}_{\mathbf{Y}}^{\text{wet}}$ representing the estimated covariance matrix of the decorrelated signals after the mixing matrix $\mathbf{P}_{\text{wet}}$ has been applied, and defined as:

$$\hat{\mathbf{E}}_{\mathbf{Y}}^{\text{wet}} = \mathbf{P}_{\text{wet}} \mathbf{E}_{\mathbf{Y}}^{\text{wet}} \mathbf{P}_{\text{wet}}^{*}.$$

The mixing matrices $\mathbf{P}_{\text{dry}}$, $\mathbf{P}_{\text{wet}}$ of size $N_{\text{out}} \times N_{\text{out}}$ are obtained according to the "decorrelation method" as:

- "Energy compensation method" (bsDecorrelationMethod == 0):

$$\mathbf{P}_{\text{dry}} = \mathbf{I},$$

$$\mathbf{P}_{\text{wet}}(i,j) = \begin{cases} \sqrt{\max\left(0, \dfrac{\mathbf{C}(i,i) - \mathbf{E}_{\mathbf{Y}}^{\text{dry}}(i,i)}{\max\left(\varepsilon, \mathbf{E}_{\mathbf{Y}}^{\text{wet}}(i,i)\right)}\right)} & i = j, \\ 0 & i \neq j. \end{cases}$$

- "Covariance adjustment method" (bsDecorrelationMethod == 1):

$$\mathbf{P}_{\text{dry}} = \mathbf{I},$$

$$\mathbf{P}_{\text{wet}} = \left(\mathbf{V}_1 \sqrt{\mathbf{Q}_1} \mathbf{V}_1^{*}\right)\left(\mathbf{V}_2 \sqrt{\mathbf{Q}_2^{inv}} \mathbf{V}_2^{*}\right).$$

The matrices $\mathbf{V}_1$ and $\mathbf{Q}_1$ are determined as the singular value decomposition of the matrix $\mathbf{\Delta}_E$ as:

$$\mathbf{\Delta}_E = \mathbf{V}_1 \mathbf{Q}_1 \mathbf{V}_1^{*}.$$

The matrices $\mathbf{V}_2$ and $\mathbf{Q}_2$ are determined as the singular value decomposition of the matrix $\mathbf{E}_{\mathbf{Y}}^{\text{wet}}$ as:

$$\mathbf{E}_{\mathbf{Y}}^{\text{wet}} = \mathbf{V}_2 \mathbf{Q}_2 \mathbf{V}_2^{*}.$$

9.5.4.2.5 **Regularized inverse operation**

The regularized inverse matrix $\mathbf{J}$ approximating $\mathbf{J} \approx \mathbf{\Delta}^{-1}$ is calculated as:

$$\mathbf{J} = \mathbf{V} \mathbf{\Lambda}^{inv} \mathbf{V}^{*}.$$

The matrices $\mathbf{V}$ and $\mathbf{\Lambda}$ are determined as the singular value decomposition of the matrix $\mathbf{\Delta}$ as:

$$\mathbf{V} \mathbf{\Lambda} \mathbf{V}^{*} = \mathbf{\Delta}.$$

The regularized inverse $\mathbf{\Lambda}^{inv}$ of the diagonal singular value matrix $\mathbf{\Lambda}$ is computed according to 9.5.4.2.6.

In the case the matrix $\mathbf{\Delta}$ is used in the calculation of the parametric un-mixing matrix $\mathbf{U}$, the operations described are applied for all sub-matrices $\mathbf{\Delta}_q$. A sub-matrix $\mathbf{\Delta}_q$ of size $N_{\text{g}}^{q} \times N_{\text{g}}^{q}$, with elements $\mathbf{\Delta}_q\left(idx_1, idx_2\right)$,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

is obtained by selecting the elements $\mathbf{\Delta}(ch_1, ch_2)$ corresponding to the downmix channels $ch_1$ and $ch_2$ assigned to the group $\mathbf{g}_q$ (i.e., $\mathbf{g}_q(idx_1) = ch_1$ and $\mathbf{g}_q(idx_2) = ch_2$).

The group $\mathbf{g}_q$ of size $1 \times N_{\mathrm{g}}^q$ is defined by the smallest set of downmix channels with the following properties:

— The input signals contained in the downmix channels of group $\mathbf{g}_q$ are not contained in any other downmix channel. An input signal is not contained in a downmix channel if the corresponding downmix gain is given by the smallest quantization index (Table 49 of ISO/IEC 23003-2:2010).

— All input signals $i$ contained in the downmix channels of group $\mathbf{g}_q$ are not related to any input signal $j$ contained in any downmix channel of any other group (i.e., **bsRelatedTo**[i][j] == 0).

The results of the independent regularized inversion operations $\mathbf{J}_q \approx \mathbf{\Delta}_q^{-1}$ are combined for obtaining the matrix $\mathbf{J}$ as:

$$\mathbf{J}(ch_1, ch_2) = \begin{cases} \mathbf{J}_q(idx_1, idx_2), & \text{if } \mathbf{g}_q(idx_1) = ch_1 \text{ and } \mathbf{g}_q(idx_2) = ch_2, \\ 0, & \text{otherwise.} \end{cases}$$

#### 9.5.4.2.6 Regularization of singular values

The regularized inverse operation $(\cdot)^{inv}$ used for the diagonal singular value matrix $\mathbf{\Lambda}$ is determined as:

$$\mathbf{\Lambda}^{inv} = \lambda_{i,j}^{-1} = \begin{cases} \dfrac{1}{\lambda_{i,i}} & \text{, if } i = j \text{ and } abs(\lambda_{i,i}) \geq T_{reg}^{\Lambda}, \\ 0 & \text{, otherwise.} \end{cases}$$

The relative regularization scalar $T_{reg}^{\Lambda}$ is determined using absolute threshold $T_{reg}$ and maximal value of $\mathbf{\Lambda}$ as follows:

$$T_{reg}^{\Lambda} = \max_i \left( \mathrm{abs}(\lambda_{i,i}) \right) T_{reg}, \text{ with } T_{reg} = 10^{-2}.$$

### 9.5.5 Dual mode

The SAOC 3D decoder can use an alternative scheme for calculation of the parameters $\mathbf{U}$ and $\mathbf{P}$ for the upper frequency range, defined by parameter bands $\mathbf{bsBandsLow} \leq pb < numBands$. This scheme is particularly useful for downmix signals where the upper frequency range is coded by a non-waveform preserving coding algorithm e.g. SBR in High Efficiency AAC. The matrices $\mathbf{U}$, $\mathbf{P}_{\mathrm{dry}}$ and $\mathbf{P}_{\mathrm{wet}}$ are determined as:

$$\mathbf{U} = \mathbf{G}\mathbf{T},$$

$$\mathbf{P}_{\mathrm{dry}} = \mathbf{I},$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$\mathbf{P}_{\text{wet}} = \mathbf{0}\,.$$

The matrix $\mathbf{T}$ is defined as:

$$\mathbf{T} = \mathbf{R}\mathbf{D}^{*}\mathbf{J}\,.$$

The matrix $\mathbf{J} \approx \mathbf{\Delta}^{-1}$ of size $N_{dmx} \times N_{dmx}$ for $\mathbf{\Delta} = \mathbf{D}\mathbf{D}^{*}$ is derived according to 9.5.4.2.5.

The gain matrix $\mathbf{G}$ of size $N_{\text{out}} \times N_{\text{out}}$ is given by:

$$\mathbf{G} = g_{i,j} = \begin{cases} \sqrt{\dfrac{c_{i,i}}{e_i^{upmix}}} & \text{, if } i = j, \\ 0 & \text{, otherwise.} \end{cases}$$

The energy upmix vector $e_i^{upmix}$ of size $1 \times N_{out}$ is given by:

$$e_i^{upmix} = diag\left(\mathbf{T}\left[matdiag\left(\mathbf{D}\mathbf{E}\mathbf{D}^{*}\right)\right]\mathbf{T}^{*} + \varepsilon\mathbf{I}\right).$$

The unmixing matrix $\mathbf{U}$ contains the rendering matrix $\mathbf{R}$ thus the rendering block in Figure 26 is omitted for the upper frequency range.

The matrix $(\mathbf{U})^{l,k}$, computed for every parameter time-slot $l$, is interpolated over all time-slots $n$ as described in 6.5.2.1 of ISO/IEC 23003-1:2007, considering:

$$\mathbf{W}_1^{l,k} = \mathbf{U}^{l,k} \quad \text{for computing} \quad \mathbf{U}^{n,k} = \mathbf{M}_1^{n,k}.$$

where $\mathbf{W}_1^{l,k}$ and $\mathbf{M}_1^{n,k}$ are variables specified in ISO/IEC 23003-1:2007.

# 10 Generic Loudspeaker Rendering/Format Conversion

## 10.1 Description

The loudspeaker renderer converts multichannel signals from transmitted channel configurations to desired reproduction formats. It is thus also called 'format converter'. If the channel configuration of the channels routed to the format converter exactly matches the signaled reproduction layout (i.e. the target channel configuration), the format converter shall be bypassed. The system consists of two major building blocks:

— An initialization algorithm that takes into account static parameters like the input and output format,

— a signal adaptive downmixing process that operates in a subband domain.

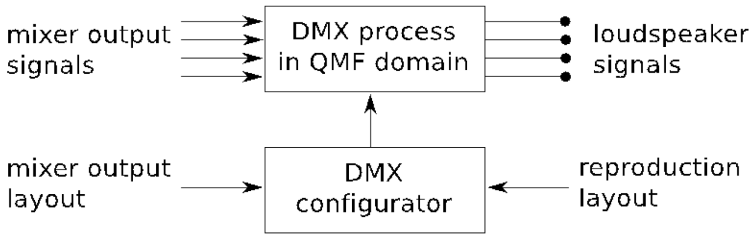



**Figure 27 — Main building blocks of the generic format converter**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

In the initialization phase the format converter automatically generates optimized downmixing parameters (like the downmixing matrix) for the given combination of input and output formats: It applies an algorithm that selects for each input loudspeaker the most appropriate mapping rule from a list of rules that has been designed to incorporate psychoacoustic considerations. Each rule describes the mapping from one input channel to one or several output loudspeaker channels.

Input channels are

- either mapped to a single output channel,
- or panned to two output channels,
- or (in case of the 'Voice of God' channel) distributed over a larger number of output channels.

The optimal mapping for each input channel is selected depending on the list of output loudspeakers that are available in the desired output format. Each mapping defines downmix gains for the input channel under consideration as well as potentially also an equalizer that is applied to the input channel under consideration.

Output setups with non-standard loudspeaker positions can be signaled to the system by providing the azimuth and elevation deviations from a regular loudspeaker setup.

The actual downmixing of the audio signals is performed on a hybrid QMF subband representation of the signals. The algorithm makes use of two mechanisms to avoid signal deteriorations like comb-filtering, coloration, or modulation artifacts:

- Phase-alignment of the multichannel input signals: Correlated input signals that differ in phase are aligned prior to downmixing them. The alignment process makes use of an attraction measure to only align the relevant channels for the relevant time-frequency tiles and to avoid modifications of other parts of the input signal. The alignment is further regularized to avoid artifacts due to rapid changes of the alignment phase modification terms. The phase-alignment improves the output signal quality by avoiding narrow spectral notches due to out-of-phase signal cancellations that could not be compensated for by energy normalization because of a limited frequency resolution. It further reduces the need of boosting signals in the energy preserving normalization, thus minimizes modulation artifacts.
- Normalization of the downmix process to preserve the input energies (except for the desired energy scaling that may be inherent in the downmix matrix).

## 10.2 Definitions

### 10.2.1 General remarks

Audio signals that are fed into the format converter are referred to as *input signals* in the following. Audio signals that are the result of the format conversion process are referred to as *output signals*. Note that the audio input signals of the format converter are audio output signals of the core decoder.

Vectors and matrices are denoted by bold-faced symbols. Vector elements or matrix elements are denoted as italic variables supplemented by indices indicating the row/column of the vector/matrix element in the vector/matrix, e.g. $\left[y_1 \cdots y_a \cdots y_N\right] = \mathbf{y}$ denotes a vector and its elements. Similarly, $M_{a,b}$ denotes the element in the $a$ th row and $b$ th column of a matrix $\mathbf{M}$.

### 10.2.2 Variable definitions

| | |
|---|---|
| $N_{\text{in}}$ | Number of channels in the input channel configuration. |
| $N_{\text{out}}$ | Number of channels in the output channel configuration. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $\mathbf{M}_{DMX}$ | Downmix matrix containing real-valued non-negative downmix coefficients (downmix gains). $\mathbf{M}_{DMX}$ is of dimension ( $N_{out}$ x $N_{in}$ ). |
| $\mathbf{G}_{EQ}$ | Matrix consisting of gain values per processing band determining frequency responses of equalizing filters. |
| $\mathbf{I}_{EQ}$ | Vector signalling which equalizer filters to apply to the input channels (if any). |
| $L$ | Frame length measured in time domain audio samples. |
| $\nu$ | Time domain sample index. |
| $n$ | QMF time slot index (= subband sample index). |
| $L_n$ | Frame length measured in QMF slots. |
| $F$ | Frame index (frame number). |
| $K$ | Number of hybrid QMF frequency bands, $K$ = 71. |
| $k$ | QMF band index (1..64) or hybrid QMF band index (1.. $K$ ). |
| $A, B$ | Channel indices. |
| eps | Numerical constant, $\text{eps} = 10^{-35}$ . |

## 10.3 Processing

### 10.3.1 Application of transmitted downmix matrices

#### 10.3.1.1 Introduction

MPEG-H 3D audio allows transmission of downmix specifications for specific target channel configurations. downmixIds are assigned to the transmitted downmix specifications, allowing DRC to adapt to the downmix specification applied in the MPEG-H decoder (e.g. to select an appropriate DRC gain sequence). Further, loudness metadata values may be coupled with downmixIds. downmixIds are transmitted in the bitstream together with the downmixType as well as the nominal loudspeaker layouts the embedded downmix matrices (and/or DRC and loudness data) have been designed for.

In the MPEG-H 3D audio decoder the selection of a downmixIds thus

— determines whether transmitted downmix coefficients (downmixType=1) or decoder side generated downmix coefficients (downmixType=0) are applied in the downmix process;

— influences DRC/loudness processing.

Two arguments speak in favor of applying transmitted downmix coefficients also for moderately displaced reproduction layouts:

— The selection of a significantly different downmix matrix (transmitted vs. decoder generated) results in large perceptual changes of the downmix result.

— The artistic intent of transmitted downmix coefficients would be lost if a transmitted downmix matrix is not applied.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

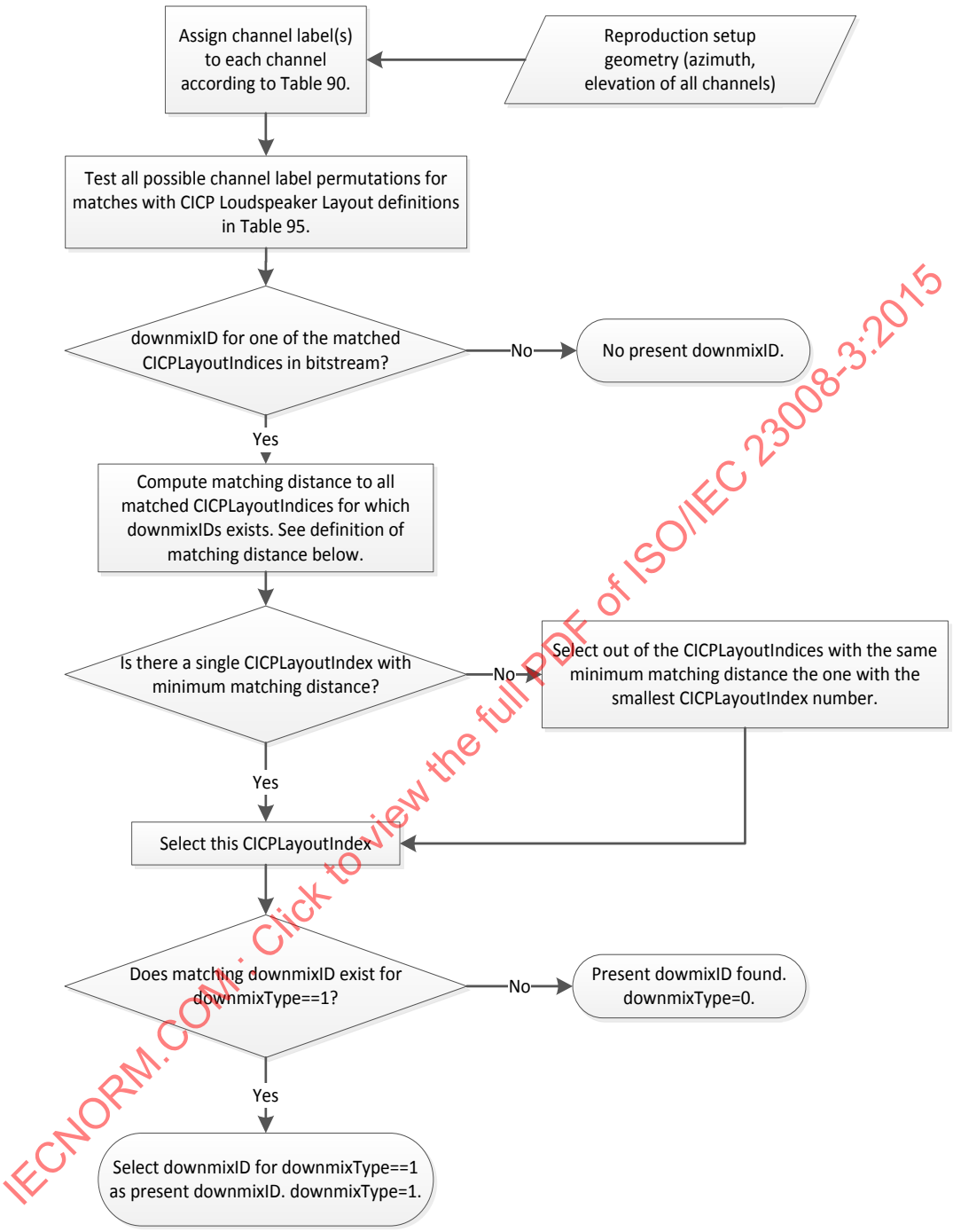
Of course, in case the loudspeaker displacements of the reproduction setup are too large, the application of the transmitted downmix coefficients may result in a larger perceptual distance from the intended reproduction than the application of decoder generated downmix coefficients. As a consequence, the allowed loudspeaker displacement values are restricted in the following matching scheme.

#### 10.3.1.2 Loudspeaker layout matching scheme

The downmixId matching algorithm is specified by the flow chart of Figure 28 below. It takes as input the geometry of the actual reproduction setup and outputs as result the present downmixId (if any). The present downmixId determines the downmix processing in the decoder as shown in 10.3.1.4. Further, the present downmixId may affect DRC and loudness processing.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved




IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Figure 28 — Flow diagram for loudspeaker leayout matching and to determine downmixId**

The *matching distance* is defined as the sum of all absolute azimuth and elevation angle differences between the channel positions of the reproduction layout and the tested CICP loudspeaker layout, summed over all channels of the reproduction setup, excluding the LFE channels:

© ISO/IEC 2015 – All rights reserved

$$d_{\text{matching}} = \sum_{ch \in \left\{\begin{matrix}\text{all channels} \\ \text{except LFEs}\end{matrix}\right\}} \left|\varphi_{\text{CICPLayout},ch} - \varphi_{\text{reprodcutionLayout},ch}\right| + \left|\vartheta_{\text{CICPLayout},ch} - \vartheta_{\text{reprodcutionLayout},ch}\right| ,$$

where $\varphi$ denotes azimuth angles and $\vartheta$ denotes elevation angles.

**Table 90 — Channel matching tolerances for matching downmixIds to reproduction layouts. Azimuth and elevation tolerance intervals are defined as sectors, where azimuth start and end values are connected in counterclockwise direction and elevation start and end values are connected in ascending elevations. Start and end values of the sectors are considered part of the sectors.**

| LoudspeakerGeometry as defined in ISO/IEC 23001-8) | Channel | Azimuth [deg] | Elevation [deg] | Azimuth start angle of sector [deg] | Azimuth end angle of sector [deg] | Elevation start angle of sector [deg] | Elevation end angle of sector [deg] | Ch. is LFE | Position is relative |
|---|---|---|---|---|---|---|---|---|---|
| | CH_EMPTY | n/a | n/a | n/a | n/a | n/a | n/a | 0 | 0 |
| 0 | CH_M_L030 | 30 | 0 | 15 | 45 | -15 | 15 | 0 | 0 |
| 1 | CH_M_R030 | -30 | 0 | -45 | -15 | -15 | 15 | 0 | 0 |
| 2 | CH_M_000 | 0 | 0 | -10 | 10 | -15 | 15 | 0 | 0 |
| 3 | CH_LFE1 | 0 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 4 | CH_M_L110 | 110 | 0 | 90 | 130 | -15 | 15 | 0 | 0 |
| 5 | CH_M_R110 | -110 | 0 | -130 | -90 | -15 | 15 | 0 | 0 |
| 6 | CH_M_L022 | 22 | 0 | 7 | 37 | -15 | 15 | 0 | 0 |
| 7 | CH_M_R022 | -22 | 0 | -37 | -7 | -15 | 15 | 0 | 0 |
| 8 | CH_M_L135 | 135 | 0 | 120 | 150 | -15 | 15 | 0 | 0 |
| 9 | CH_M_R135 | -135 | 0 | -150 | -120 | -15 | 15 | 0 | 0 |
| 10 | CH_M_180 | 180 | 0 | 170 | 190 | -15 | 15 | 0 | 0 |
| 13 | CH_M_L090 | 90 | 0 | 70 | 110 | -15 | 15 | 0 | 0 |
| 14 | CH_M_R090 | -90 | 0 | -110 | -70 | -15 | 15 | 0 | 0 |
| 15 | CH_M_L060 | 60 | 0 | 40 | 80 | -15 | 15 | 0 | 0 |
| 16 | CH_M_R060 | -60 | 0 | -80 | -40 | -15 | 15 | 0 | 0 |
| 17 | CH_U_L030 | 30 | 35 | 15 | 45 | 15 | 55 | 0 | 0 |
| 18 | CH_U_R030 | -30 | 35 | -45 | -15 | 15 | 55 | 0 | 0 |
| 19 | CH_U_000 | 0 | 35 | -15 | 15 | 15 | 55 | 0 | 0 |
| 20 | CH_U_L135 | 135 | 35 | 115 | 155 | 15 | 55 | 0 | 0 |
| 21 | CH_U_R135 | -135 | 35 | -155 | -115 | 15 | 55 | 0 | 0 |
| 22 | CH_U_180 | 180 | 35 | 165 | 195 | 15 | 55 | 0 | 0 |
| 23 | CH_U_L090 | 90 | 35 | 70 | 110 | 15 | 55 | 0 | 0 |
| 24 | CH_U_R090 | -90 | 35 | -110 | -70 | 15 | 55 | 0 | 0 |
| 25 | CH_T_000 | 0 | 90 | -180 | 180 | 60 | 90 | 0 | 0 |
| 26 | CH_LFE2 | 45 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 27 | CH_L_L045 | 45 | -15 | 25 | 65 | -40 | 0 | 0 | 0 |
| 28 | CH_L_R045 | -45 | -15 | -65 | -25 | -40 | 0 | 0 | 0 |
| 29 | CH_L_000 | 0 | -15 | -15 | 15 | -40 | 0 | 0 | 0 |
| 30 | CH_U_L110 | 110 | 35 | 90 | 130 | 15 | 55 | 0 | 0 |
| 31 | CH_U_R110 | -110 | 35 | -130 | -90 | 15 | 55 | 0 | 0 |
| 32 | CH_U_L045 | 45 | 35 | 30 | 60 | 15 | 55 | 0 | 0 |
| 33 | CH_U_R045 | -45 | 35 | -60 | -30 | 15 | 55 | 0 | 0 |
| 34 | CH_M_L045 | 45 | 0 | 30 | 60 | -15 | 15 | 0 | 0 |
| 35 | CH_M_R045 | -45 | 0 | -60 | -30 | -15 | 15 | 0 | 0 |
| 36 | CH_LFE3 | -45 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |

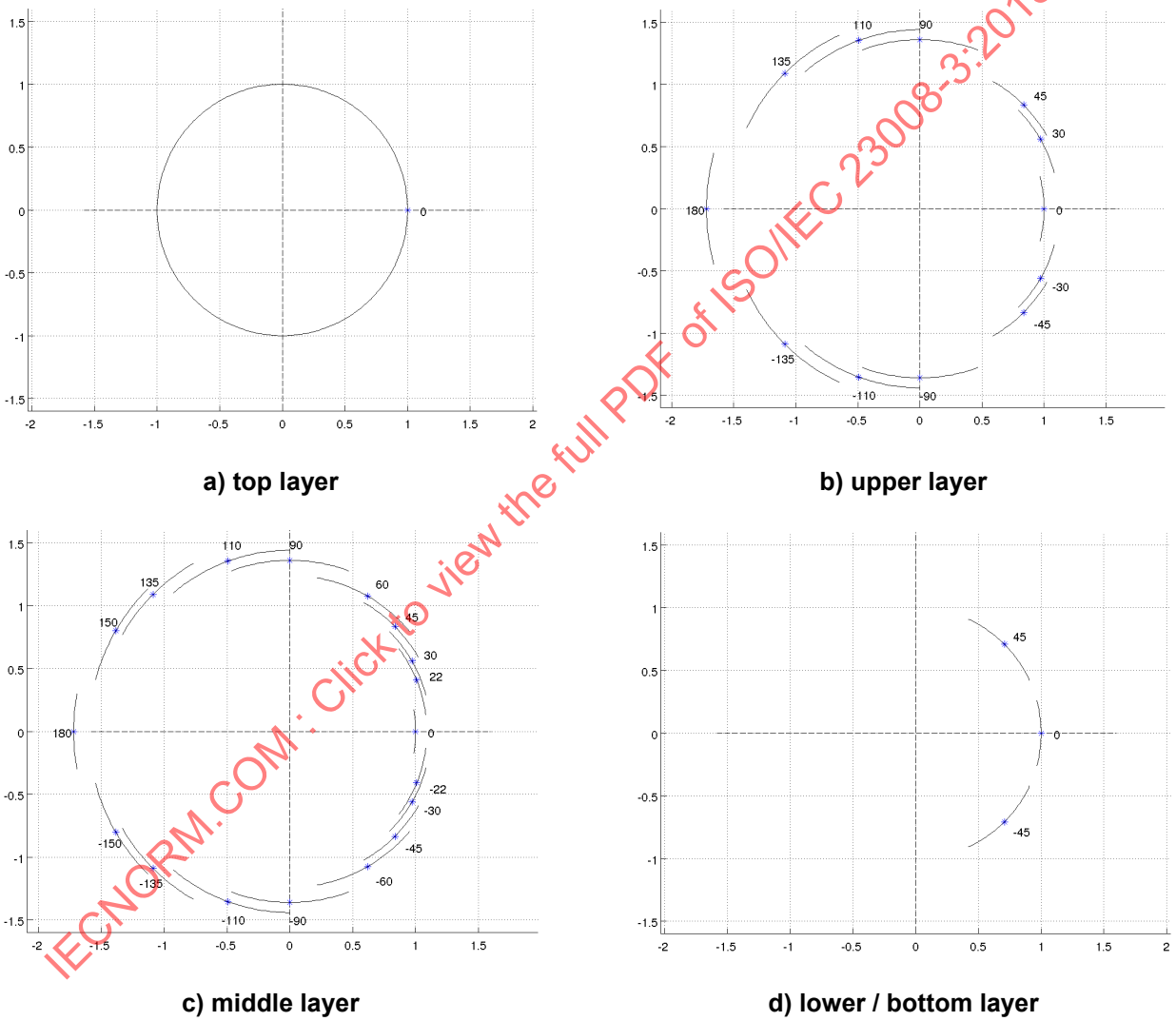
IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 37 | CH_M_LSCR | 60 | 0 | 15 | 80 | -15 | 15 | 0 | 1 |
| 38 | CH_M_RSCR | -60 | 0 | -80 | -15 | -15 | 15 | 0 | 1 |
| 39 | CH_M_LSCH | 30 | 0 | 7 | 40 | -15 | 15 | 0 | 1 |
| 40 | CH_M_RSCH | -30 | 0 | -40 | -7 | -15 | 15 | 0 | 1 |
| 41 | CH_M_L150 | 150 | 0 | 135 | 165 | -15 | 15 | 0 | 0 |
| 42 | CH_M_R150 | -150 | 0 | -165 | -135 | -15 | 15 | 0 | 0 |

#### 10.3.1.3 Visualization of azimuth tolerances

The following figures depict the azimuth matching sectors for each loudspeaker. The tolerances reflect the non-isotropic sound localization performance of the human auditory system.




IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Figure 29 — Visualization of azimuth tolerances.**
**Arcs have been plotted on different radii just for clarity of presentation. Nominal positions of the channels have been marked with asterisks and labels indicating the nominal azimuth angle in degrees.**

#### 10.3.1.4 Determination of downmix processing depending on present downmixId

The downmix processing in the MPEG-H decoder is determined by the present downmixId as follows:

© ISO/IEC 2015 – All rights reserved

— If there is no present downmixId for the current reproduction setup, the downmix coefficients shall be derived as specified in the format converter initialization.

— If downmixType==0 for the present downmixId, the downmix coefficients shall be derived as specified in the format converter initialization.

— If downmixType==1 for the present downmixId, the downmix coefficients transmitted with the downmixId shall be applied in the downmix.

### 10.3.2 Application of transmitted equalizer settings

Equalizer settings may be transmitted together with downmix matrices, as indicated by the **EqualizerPresent** bitstream element. In case equalizer settings have been transmitted together with a downmix matrix that is applied in the format converter, the equalizers shall be applied to this downmix matrix as follows:

The transmitted equalizer parameters shall be decoded into frequency dependent gains (i.e. into equalizer frequency responses) according to 10.3.4.6.4. Next, the transmitted downmix matrix gains shall be multiplied by the frequency dependent equalizer gains to arrive at the final frequency dependent downmix matrix that shall be applied in the downmix. Note that the assignment of equalizer gains to downmix matrix elements is given by the vector **equalizerIndex** that is derived according to Table 21 - Syntax of EqualizerConfig: **equalizerIndex** tells for each input channel whether an equalizer (and if any: which) shall be applied to an input channel by applying the corresponding equalizer gains to all downmix matrix coefficients associated with the respective input channel.

### 10.3.3 Downmix processing involving multiple channel groups

#### 10.3.3.1 General

In case multiple channel groups are transmitted in the MPEG-H 3D Audio bitstream and routed to the format converter, one instance of the format converter shall perform a downmix of all input channels routed to the format converter to the desired target channel configuration. Therefore all channels routed to the format converter are compiled in a group of channels that constitutes the input to the format conversion process.

If downmix matrices to one or more target channel configurations are transmitted in the bitstream, downmix matrices for those target channel configurations shall be transmitted for all channel groups/channel elements present in the bitstream.

#### 10.3.3.2 Downmix processing with decoder generated downmix gains

In case no appropriate downmix matrices have been transmitted for the signaled target loudspeaker configuration, the downmix gains are generated during the initialization of the format converter according to 10.3.4. The channels are fed to the format converter as a group of all input channels routed to the format converter. The input channel configuration signaled to the format converter shall reflect the channel geometry of this group of channels.

#### 10.3.3.3 Downmix processing with transmitted downmix gains

In case downmix matrices have been transmitted that are applicable to the desired target channel configuration, those downmix matrices shall be applied in the format converter instead of generating downmix gains in the format converter initialization process. Whether a downmix matrix is applicable for a desired target setup, or not, is determined according to 10.3.1.

All channels (and/or groups of channels) shall be compiled in one group of channels, where this group of channels consists of blocks of channels that are the channels (and/or groups of channels) routed to the format converter.

The transmitted downmix matrices assigned to the channels routed to the format converter shall be concatenated according to the order of the blocks of channels in the group of channels that forms the input to

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

the format converter. The concatenated downmix matrix shall then be applied in the format converter to derive the downmixed signal in the desired target channel configuration.

### 10.3.4 Initialization of the format converter

#### 10.3.4.1 General description of the initialization

The initialization of the format converter is carried out before processing of the audio samples delivered by the core decoder takes place.

The initialization takes into account as input parameters:

- The sampling rate of the audio data to process.
- The channel configuration of the audio data to process with the format converter (number and geometric positions of input channels).
- The channel configuration of the desired output format (number and geometric positions of output channels).
- Optional: Parameters signaling the deviation of the output loudspeaker positions from a standard loudspeaker setup (random setup functionality).

It returns

- A frequency dependent downmix matrix $\mathbf{M}_{\mathrm{DMX}}$ that is applied in the audio signal processing of the format converter. $\mathbf{M}_{\mathrm{DMX}}$ is also taken into account in the core decoding process, see 5.5.3.1.2.

The input parameters to the initialization algorithm are listed in Table 91.

**Table 91 — Format converter initialization input parameters**

| | |
|---|---|
| | Input format: number of channels and nominal channel setup geometry |
| | Output format: number of channels and nominal channel setup geometry |
| $f_s$ | Sampling frequency in Hertz. |
| $r_{\mathrm{azi},A}$ | For each output channel *A*, an azimuth angle is specified, determining the deviation from the standard format loudspeaker azimuth. |
| $r_{\mathrm{ele},A}$ | For each output channel *A*, an elevation angle is specified, determining the deviation from the standard format loudspeaker elevation. |

Table 92 lists the output parameters that are derived during the initialization of the format converter.

**Table 92 — Format converter initialization output parameters**

| | |
|---|---|
| $\mathbf{M}_{\mathrm{DMX}}$ | Downmix matrix [linear gains] |

#### 10.3.4.2 Assignment of format converter channel labels to input/output format channels

The format converter initialization is based on a system of rules that are defined in terms of *format converter channel labels*, see **Table 94**. To allow the application of the initialization rules, the channel labels have to be assigned to the channels of the input and output formats. Each format converter channel label is associated

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

with a segment of the surface of the unit sphere, as defined in Table 94. The segments are designed non-overlapping.

The assignment of channel labels to channels is done by geometrically matching the segments to the position data associated with the channels of the input and output formats. The azimuth and elevation angles in degrees of the position data associated with the channels shall be rounded towards the nearest integer number before performing the channel label assignment. Note that the *nominal* channel positions shall be applied in the following matching to channel label sectors, i.e. the azimuth and elevation angles *without* taking into account potential angle deviations signalled in $r_{\mathrm{azi},A}$ and/or $r_{\mathrm{ele},A}$ .

For each channel that is not an LFE (low-frequency effects) channel:

If the nominal position of the current channel, defined by its azimuth angle and elevation angle, is within or on the border of one of the segments defined in Table 94 then:

> Assign the corresponding channel label (e.g. CH_M_L030) associated with the matching segment.
>
> Add the angle differences between the nominal position of the current channel and the nominal position associated with the matching segment (i.e. the angles in the second and third column of Table 94) to the angle deviations stored in $r_{\mathrm{azi},A}$ and $r_{\mathrm{ele},A}$.

Else (i.e. no matching sector found), then:

> Assign the CH_EMPTY label.

If an input or output format contains exactly one LFE channel, then the label CH_LFE2 shall be assigned to this channel.

If an input or output format contains exactly two LFE channels, then the labels CH_LFE2 and CH_LFE3 shall be assigned to the two LFE channels in the order that minimizes the maximum azimuth distance from the channels to the assigned CH_LFE2 and CH_LFE3 nominal azimuth positions.

If an input or output format contains more than 2 LFE channels, then those 2 LFE channels out of the considered setup shall be selected that minimize the maximum azimuth distance to the CH_LFE2 and CH_LFE3 nominal azimuth positions. The labels CH_LFE2 and CH_LFE3 shall be assigned as in the case of two LFE channels. The remaining LFE channels shall not be considered further in the calculation of downmix coefficients, i.e. the corresponding lines/columns of the downmix matrix shall remain filled with zeros.

#### 10.3.4.3 Handling for unknown input channels

If the label CH_EMPTY is assigned to an input channel, this channel shall be considered unknown to the rules-based initialization and the downmix coefficients for mapping this input channel to the output channels shall be derived as specified in 10.3.4.6.7.

#### 10.3.4.4 Handling for unknown output formats

If the output format contains at least one channel with the label CH_EMPTY assigned to it, or if at least one channel label is assigned to more than one channel of the output format, the output format shall be considered unknown and the derivation of the downmixing coefficients shall be carried out as specified in 10.3.4.6.7. The rules-based derivation of downmix coefficients shall not be applied for unknown output formats.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 10.3.4.5 Handling of deviations from standard loudspeaker positions

If the below conditions are not met, the rules-based initialization is considered to have failed, the ouput format shall be considered to be unknown, and the downmixing gains shall be obtained as defined in 10.3.4.6.7

The absolute values of $r_{\mathrm{azi},A}$ and $r_{\mathrm{ele},A}$ shall not exceed 35 and 55 degrees, respectively. The minimum angle between any loudspeaker pair (without LFE channels) shall not be smaller than 15 degrees.

The values of $r_{\mathrm{azi},A}$ shall be such that the ordering by azimuth angles of the horizontal loudspeakers does not change. Likewise, the ordering of the height and low loudspeakers shall not change.

The values of $r_{\mathrm{ele},A}$ shall be such that the ordering by elevation angles of loudspeakers which are (approximately) above/below each other does not change. To verify this, the following procedure is applied:

For each row of **Table 99** which contains two or three channels of the output format, do:

— Order the channels by elevation without randomization

— Order the channels by elevation with considering randomization

— If the two orderings differ, return an initialization error

#### 10.3.4.6 Rules-Based Initialization algorithm

##### 10.3.4.6.1 General

The rules-based initialization algorithm is defined in the following. It shall not be applied if the output format is considered unknown as defined above. Note that the following description makes use of the intermediate parameters defined in Table 93 only for clarity reasons. An implementation may omit the introduction of these intermediate parameters.

**Table 93 — Format converter initialization intermediate parameters**

| | |
|---|---|
| $\mathbf{S}$ | Vector of converter source channels [input channel indices] |
| $\mathbf{D}$ | Vector of converter destination channels [output channel indices] |
| $\mathbf{G}$ | Vector of converter gains [linear] |
| $\mathbf{E}$ | Vector of converter EQ indices |
| $\mathbf{G}_{\mathrm{EQ}}$ | Matrix containing equalizer gain values for all EQ indices and frequency bands |

The intermediate parameters describe the dowmixing parameters in a mapping-oriented way, i.e. as sets of parameters $S_i, D_i, G_i, E_i$ per mapping *i*.

The format converter initialization output parameters are derived as described in the following steps:

##### 10.3.4.6.2 Random setups Pre-Processing:

Random output loudspeaker setups, i.e. output setups that contain loudspeakers at positions deviating from the positions defined for the desired output format are signalled by specifying the loudspeaker position deviation angles as input parameters $r_{\mathrm{azi},A}$ and $r_{\mathrm{ele},A}$. The angle deviations are taken into account as a pre-processing step:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

Modify in Table 94 the channels' azimuth and elevation angles by adding $r_{\mathrm{azi},A}$ and $r_{\mathrm{ele},A}$ to the corresponding channels' azimuth and elevation angles.

10.3.4.6.3 **Derivation of input channel/output channel mapping parameters:**

The parameters vectors $\mathbf{S}$, $\mathbf{D}$, $\mathbf{G}$, $\mathbf{E}$ define the mapping of input channels to output channels. For each mapping *i* from an input channel to an output channel with non-zero downmix gain they define the downmix gain as well as an equalizer index that indicates which equalizer curve has to be applied to the input channel under consideration in mapping *i*.

The elements of the parameter vectors $\mathbf{S}$, $\mathbf{D}$, $\mathbf{G}$, $\mathbf{E}$ are derived by the following algorithm:

Initialize the mapping counter *i*: $i = 1$;

For each input channel, ignoring channels with label CH_EMPTY assigned to them:

If the input channel also exists in the output format (e.g. input channel under consideration is CH_M_R030 and channel CH_M_R030 exists in the output format), then:

> $S_i$ = index of source channel in input (***Example:*** *channel CH_M_R030 in ChannelConfiguration 6 is at second place according to Table 95, i.e. has index 2 in this format)*
>
> $D_i$ = index of same channel in output
>
> $G_i$ = 1.0
>
> $E_i$ = 0
>
> $i = i + 1$

Else (i.e. if the input channel does not exist in the output format)

> search the first entry of this channel in the **Source** column of Table 96, for which the channels in the corresponding row of the **Destination** column exist. The ALL_U destination shall be considered valid (i.e. the relevant output channels exist) if the output format contains at least one "CH_U_" channel. The ALL_M destination shall be considered valid (i.e. the relevant output channels exist) if the output format contains at least one "CH_M_" channel. If for no entry in Table 96 corresponding to the input channel the channels in the **Destination** column exist, the rules-based initialization shall terminate and the downmix gains shall be derived according to 10.3.4.6.7.
>
> If **Destination** column contains ALL_U, then:
>
> > For each output channel x with "CH_U_" in its name, do:
> >
> > $S_i$ = index of source channel in input
> >
> > $D_i$ = index of channel x in output
> >
> > $G_i$ = (value of Gain column) / sqrt(number of "CH_U_" output channels)
> >
> > $E_i$ = value of EQ column

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$i = i + 1$

Else if **Destination** column contains ALL_M, then:

For each output channel x with “CH_M_” in its name, do:

$S_i$ = index of source channel in input

$D_i$ = index of channel x in output

$G_i$ = (value of Gain column) / sqrt(number of “CH_M_” output channels)

$E_i$ = value of EQ column

$i = i + 1$

Else If there is one channel in the **Destination** column, then:

$S_i$ = index of source channel in input

$D_i$ = index of destination channel in output

$G_i$ = value of Gain column

$E_i$ = value of EQ column

$i = i + 1$

Else (two channels in **Destination** column)

$S_i$ = index of source channel in input

$D_i$ = index of first destination channel in output

$G_i$ = (value of Gain column) * $g_1$

$E_i$ = value of EQ column

$i = i + 1$

$S_i$ = $S_{i-1}$

$D_i$ = index of second destination channel in output

$G_i$ = (value of Gain column) * $g_2$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$E_i = E_{i-1}$

$i = i + 1$

The gains $g_1$ and $g_2$ are computed by applying tangent law amplitude panning in the following way:

— Unwrap source destination channel azimuth angles to be positive.

— The azimuth angles of the destination channels are $\alpha_1$ and $\alpha_2$ (see **Table 94**).

— The azimuth angle of the source channel ( = panning target) is $\alpha_{\text{src}}$.

— $\alpha_0 = \dfrac{|\alpha_1 - \alpha_2|}{2}$

— $\alpha_{\text{center}} = \dfrac{\alpha_1 + \alpha_2}{2}$

— $\alpha = (\alpha_{\text{center}} - \alpha_{\text{src}}) \cdot \text{sgn}(\alpha_2 - \alpha_1)$

— $g_1 = \dfrac{g}{\sqrt{1+g^2}}, \quad g_2 = \dfrac{1}{\sqrt{1+g^2}}$ with $g = \dfrac{\tan\alpha_0 - \tan\alpha + 10^{-10}}{\tan\alpha_0 + \tan\alpha + 10^{-10}}$

10.3.4.6.4 **Derivation of equalizer gains $\mathbf{G}_{\text{EQ}}$:**

$\mathbf{G}_{\text{EQ}}$ consists of gain values per frequency band *k* and equalizer index *e*. The 5 predefined equalizers are combinations of different peak filters. Each equalizer is a serial cascade of one or more peak filters and a gain:

$$G_{\text{EQ},e}^{k} = 10^{\frac{g}{20}} \prod_{n=1}^{N} \text{peak}\left(band(k)\frac{f_s}{2}, P_{f,n}, P_{Q,n}, P_{g,n}\right)$$

where *band*(*k*) is the normalized center frequency of frequency band *k*, specified in Table 97, $f_s$ is the sampling frequency, and function *peak()* is for negative *G*

$$\text{peak}(b, f, Q, G) = \sqrt{\frac{b^4 + \left(\frac{1}{Q^2} - 2\right) f^2 b^2 + f^4}{b^4 + \left(\frac{10^{\frac{-G}{10}}}{Q^2} - 2\right) f^2 b^2 + f^4}}$$

and otherwise

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$\text{peak}(b,f,Q,G)=\sqrt{\frac{b^4+\left(\frac{10^{\frac{G}{10}}}{Q^2}-2\right)f^2b^2+f^4}{b^4+\left(\frac{1}{Q^2}-2\right)f^2b^2+f^4}}$$

The parameters for the equalizers are specified in Table 98.

10.3.4.6.5 **Post-Processing for Random Setups**

Once the output parameters are computed, they are modified related to the specific random azimuth and elevations angles. This step has only to be carried out, if not all $r_{\text{ele},A}$ are zero. Definition of the post-processing algorithm:

For each element *i* in $D_i$, do:

**if** output channel with index $D_i$ is a horizontal channel by definition (i.e. output channel label contains the label '_M_'), **and**

**if** this output channel is now a height channel (elevation in range 0..60 degrees), **and**

**if** input channel with index $S_i$ is a height channel (i.e. label contains '_U_'), **then**

—*h* = min(elevation of randomized output channel, 35) / 35

— $G_{\text{comp}} = h \cdot \frac{1}{0.85} + (1-h)$

—Apply compensation gain to DMX gain: $G_i = G_i \cdot G_{\text{comp}}$

— Define new equalizer with a new index *e*, where $G^k_{\text{EQ},e} = h + (1-h) \cdot G^k_{\text{EQ},E_i}$

— $E_i = e$

**else if** input channel with index $S_i$ is a horizontal channel (label contains '_M_')

— *h* = min(elevation of randomized output channel, 35) / 35

— Define new equalizer with a new index *e*, where $G^k_{\text{EQ},e} = h \cdot G^k_{\text{EQ},5} + (1-h) \cdot G^k_{\text{EQ},E_i}$

— $E_i = e$

*Explanation of the post-processing steps defined above:*

*h* is a normalized elevation parameter indicating the elevation of a nominally horizontal output channel ('_M_') due to a random setup elevation offset $r_{\text{ele},A}$. For zero elevation offset *h*=0 follows and effectively no post-processing is applied.

The rules table (Table 96) in general applies a gain of 0.85 when mapping an upper input channel ('_U_' in channel label) to one or several horizontal output channels ('_M_' in channel label(s)). In case the output channel gets elevated due to a random setup elevation offset $r_{\text{ele},A}$, the gain of 0.85 is partially (0<*h*<1) or fully

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

(*h*=1) compensated for. Similarly the equalizer definitions fade towards a flat EQ-curve ( $G^k_{\mathrm{EQ},e} = const. = 1.0$ ) for *h* approaching *h* =1.

In case a horizontal input channel gets mapped to an output channel that gets elevated due to a random setup elevation offset $r_{\mathrm{ele},A}$, the equalizer $G^k_{\mathrm{EQ},5}$ is partially (0<*h*<1) or fully (*h*=1) applied.

10.3.4.6.6 **Derivation of rules-based initialization downmix matrix:**

$\mathbf{M}_{\mathrm{DMX}}$ is derived by rearranging the temporary parameters from the mapping-oriented representation (enumerated by mapping counter *i*) to a channel-oriented representation as defined in the following:

Initialize $\mathbf{M}^k_{\mathrm{DMX}}$ as an $N_{\mathrm{out}}$ x $N_{\mathrm{in}}$ zero matrix for all processing bands *k*.

For each *i* do:

If ( $E_i = 0$ )

$$M^k_{\mathrm{DMX},A,B} = G_i \quad \text{for} \quad A = D_i,\ B = S_i,\ 1 \le k \le K$$

Else

$$M^k_{\mathrm{DMX},A,B} = G_i \cdot G^k_{\mathrm{EQ},E_i} \quad \text{for} \quad A = D_i,\ B = S_i,\ 1 \le k \le K$$

where $M^k_{\mathrm{DMX},A,B}$ denotes the matrix element in the $A$th row and $B$th column of $\mathbf{M}^k_{\mathrm{DMX}}$. Note that after the rules-based initialization this matrix of downmix coefficients will contain columns of zeros, if unknown channels are present in the input format. Those columns are filled with downmix gains as specified in 10.3.4.6.7.

10.3.4.6.7 **VBAP-based downmix coefficients derivation**

Handling of unknown output formats:

In case the output format is considered unknown, the downmix coefficients for all input channels shall be derived as follows:

Each channel of the input setup is regarded as a static audio object at the position defined by the azimuth and elevation angles associated with the input channel. For each input channel the mixing gains to all output loudspeakers are calculated as VBAP panning gains $\mathbf{g}_{\mathbf{scaled}}$ according to 8.4.3, where the same output format shall be signaled to the VBAP algorithm as to the format converter. The panning gain vectors $\mathbf{g}_{\mathbf{scaled}}$ shall be post-processed according to 10.3.4.6.8.

The downmix matrix $\mathbf{M}^k_{\mathrm{DMX}}$ is finally derived by filling each matrix column with the post-processed panning gain vector elements of the corresponding input channel, independently of the processing band index k.

Handling of unknown input channels:

In case the input format contains unknown input channels, the downmix coefficients for these channels shall be derived as follows:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

Each unknown channel of the input setup is regarded as a static audio object at the position defined by the azimuth and elevation angles associated with the input channel. For each unknown input channel the mixing gains to all output loudspeakers are calculated as VBAP panning gains $\mathbf{g}_{\mathbf{scaled}}$ according to 8.4.3, where the same output format shall be signaled to the VBAP algorithm as to the format converter. The panning gain vectors $\mathbf{g}_{\mathbf{scaled}}$ shall be post-processed according to 10.3.4.6.8.

The downmix matrix $\mathbf{M}_{\mathrm{DMX}}^{k}$ is finally derived by filling each matrix column corresponding to an unknown input channel with the post-processed panning gain vector elements of the corresponding unknown input channel, independently of the processing band index k.

10.3.4.6.8 **VBAP gains post-processing**

The mixing gains obtained from the VBAP rendering algorithm shall be post-processed to avoid excessive use of phantom sources. Therefore, small matrix gains are set to zero, followed by a renormalization of the panning gains to ensure energy-preservation:

For each panning gain vector $\mathbf{g}_{\mathbf{scaled}}$ do:

If the vector contains at least one panning gain that exceeds the threshold value 0.3, then

Set all vector elements smaller or equal to 0.3 to the value 0.0

Normalize the gain vector such that the sum of squares of the vector elements remains the same as before the post-processing

### 10.3.4.7 Format converter initialization tables

Table 94 lists channel labels, corresponding azimuth and elevation angles, and associated sectors. The sectors are defined as points on the unit sphere, whose azimuth/elevation angles are within or on the borders of the intervals given by the azimuth/elevation start and end values in the table, connecting azimuth start and end values in counter-clockwise direction and connecting elevation start and end values in direction of increasing elevation angles.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 94 — Channels definitions: Channel labels, corresponding azimuth and elevation angles, and associated sectors**

| LoudspeakerGeometry as defined in ISO/IEC 23001-8) | Channel | Azimuth [deg] | Elevation [deg] | Azimuth start angle of sector [deg] | Azimuth end angle of sector [deg] | Elevation start angle of sector [deg] | Elevation end angle of sector [deg] | Ch. is LFE | Position is relative |
|---|---|---|---|---|---|---|---|---|---|
| | CH_EMPTY | n/a | n/a | n/a | n/a | n/a | n/a | 0 | 0 |
| 0 | CH_M_L030 | +30 | 0 | +23 | +37 | -9 | +20 | 0 | 0 |
| 1 | CH_M_R030 | -30 | 0 | -37 | -23 | -9 | +20 | 0 | 0 |
| 2 | CH_M_000 | 0 | 0 | -7 | +7 | -9 | +20 | 0 | 0 |
| 3 | CH_LFE1 | 0 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 4 | CH_M_L110 | +110 | 0 | +101 | +124 | -45 | +20 | 0 | 0 |
| 5 | CH_M_R110 | -110 | 0 | -124 | -101 | -45 | +20 | 0 | 0 |
| 6 | CH_M_L022 | +22 | 0 | +8 | +22 | -9 | +20 | 0 | 0 |
| 7 | CH_M_R022 | -22 | 0 | -22 | -8 | -9 | +20 | 0 | 0 |
| 8 | CH_M_L135 | +135 | 0 | 125 | 142 | -45 | +20 | 0 | 0 |
| 9 | CH_M_R135 | -135 | 0 | -142 | -125 | -45 | +20 | 0 | 0 |
| 10 | CH_M_180 | 180 | 0 | 158 | -158 | -45 | +20 | 0 | 0 |
| 13 | CH_M_L090 | +90 | 0 | +76 | +100 | -45 | +20 | 0 | 0 |
| 14 | CH_M_R090 | -90 | 0 | -100 | -76 | -45 | +20 | 0 | 0 |
| 15 | CH_M_L060 | +60 | 0 | +53 | +75 | -9 | +20 | 0 | 0 |
| 16 | CH_M_R060 | -60 | 0 | -75 | -53 | -9 | +20 | 0 | 0 |
| 17 | CH_U_L030 | +30 | +35 | +11 | +37 | +21 | +60 | 0 | 0 |
| 18 | CH_U_R030 | -30 | +35 | -37 | -11 | +21 | +60 | 0 | 0 |
| 19 | CH_U_000 | 0 | +35 | -10 | +10 | +21 | +60 | 0 | 0 |
| 20 | CH_U_L135 | +135 | +35 | +125 | +157 | +21 | +60 | 0 | 0 |
| 21 | CH_U_R135 | -135 | +35 | -157 | -125 | +21 | +60 | 0 | 0 |
| 22 | CH_U_180 | 180 | +35 | +158 | -158 | +21 | +60 | 0 | 0 |
| 23 | CH_U_L090 | +90 | +35 | +67 | +100 | +21 | +60 | 0 | 0 |
| 24 | CH_U_R090 | -90 | +35 | -100 | -67 | +21 | +60 | 0 | 0 |
| 25 | CH_T_000 | 0 | +90 | -180 | +180 | +61 | +90 | 0 | 0 |
| 26 | CH_LFE2 | +45 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 27 | CH_L_L045 | +45 | -15 | +11 | +75 | -45 | -10 | 0 | 0 |
| 28 | CH_L_R045 | -45 | -15 | -75 | -11 | -45 | -10 | 0 | 0 |
| 29 | CH_L_000 | 0 | -15 | -10 | +10 | -45 | -10 | 0 | 0 |
| 30 | CH_U_L110 | +110 | +35 | +101 | +124 | +21 | +60 | 0 | 0 |
| 31 | CH_U_R110 | -110 | +35 | -124 | -101 | +21 | +60 | 0 | 0 |
| 32 | CH_U_L045 | +45 | +35 | +38 | +66 | +21 | +60 | 0 | 0 |
| 33 | CH_U_R045 | -45 | +35 | -66 | -38 | +21 | +60 | 0 | 0 |
| 34 | CH_M_L045 | +45 | 0 | +38 | +52 | -9 | +20 | 0 | 0 |
| 35 | CH_M_R045 | -45 | 0 | -52 | -38 | -9 | +20 | 0 | 0 |
| 36 | CH_LFE3 | -45 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 37 | CH_M_LSCR | +60 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 38 | CH_M_RSCR | -60 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 39 | CH_M_LSCH | +30 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 40 | CH_M_RSCH | -30 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 41 | CH_M_L150 | +150 | 0 | 143 | 157 | -45 | +20 | 0 | 0 |
| 42 | CH_M_R150 | -150 | 0 | -157 | -143 | -45 | +20 | 0 | 0 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 95 — Formats with corresponding number of channels and channel ordering**

| Loudspeaker Layout Index or ChannelConfiguration as defined in ISO/IEC 23001-8 | Number of Channels | Channels (with ordering) |
|---|---|---|
| 1 | 1 | CH_M_000 |
| 2 | 2 | CH_M_L030, CH_M_R030 |
| 3 | 3 | CH_M_L030, CH_M_R030, CH_M_000 |
| 4 | 4 | CH_M_L030, CH_M_R030, CH_M_000, CH_M180 |
| 5 | 5 | CH_M_L030, CH_M_R030, CH_M_000, CH_M_L110, CH_M_R110 |
| 6 | 6 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110 |
| 7 | 8 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_L060, CH_M_R060 |
| 8 | | n.a. |
| 9 | 3 | CH_M_L030, CH_M_R030, CH_M_180 |
| 10 | 4 | CH_M_L030, CH_M_R030, CH_M_L110, CH_M_R110 |
| 11 | 7 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_180 |
| 12 | 8 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_L135, CH_M_R135 |
| 13 | 24 | CH_M_L060, CH_M_R060, CH_M_000, CH_LFE2, CH_M_L135, CH_M_R135, CH_M_L030, CH_M_R030, CH_M_180, CH_LFE3, CH_M_L090, CH_M_R090, CH_U_L045, CH_U_R045, CH_U_000, CH_T_000, CH_U_L135, CH_U_R135, CH_U_L090, CH_U_R090, CH_U_180, CH_L_000, CH_L_L045, CH_L_R045 |
| 14 | 8 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_U_L030, CH_U_R030 |
| 15 | 12 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE2, CH_M_L135, CH_M_R135, CH_LFE3, CH_M_L090, CH_M_R090, CH_U_L045, CH_U_R045, CH_U_180 |
| 16 | 10 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_U_L030, CH_U_R030, CH_U_L110, CH_U_R110 |
| 17 | 12 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_U_L030, CH_U_R030, CH_U_000, CH_U_L110, CH_U_R110, CH_T_000 |
| 18 | 14 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_L150, CH_M_R150, CH_U_L030, CH_U_R030, CH_U_000, CH_U_L110, CH_U_R110, CH_T_000 |
| 19 | 12 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L135, CH_M_R135, CH_M_L090, CH_M_R090, CH_U_L030, CH_U_R030, CH_U_L135, CH_U_R135 |
| 20 | 14 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L135, CH_M_R135, CH_M_L090, CH_M_R090, CH_U_L045, CH_U_R045, CH_U_L135, CH_U_R135, CH_M_LSCR, CH_M_RSCR |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 96 — Converter rules matrix**

| Source | Destination | Gain | EQ index |
|---|---|---|---|
| CH_M_000 | CH_M_L022, CH_M_R022 | 1.0 | 0 (off) |
| CH_M_000 | CH_M_L030, CH_M_R030 | 1.0 | 0 (off) |
| CH_M_L022 | CH_M_000, CH_M_L030 | 1.0 | 0 (off) |
| CH_M_L022 | CH_M_L030 | 1.0 | 0 (off) |
| CH_M_R022 | CH_M_000, CH_M_R030 | 1.0 | 0 (off) |
| CH_M_R022 | CH_M_R030 | 1.0 | 0 (off) |
| CH_M_L045 | CH_M_L030, CH_M_L060 | 1.0 | 0 (off) |
| CH_M_L045 | CH_M_L030 | 1.0 | 0 (off) |
| CH_M_R045 | CH_M_R030, CH_M_R060 | 1.0 | 0 (off) |
| CH_M_R045 | CH_M_R030 | 1.0 | 0 (off) |
| CH_M_L060 | CH_M_L045, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L060 | CH_M_L030, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L060 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R060 | CH_M_R045, CH_M_R110, | 1.0 | 0 (off) |
| CH_M_R060 | CH_M_R030, CH_M_R110, | 1.0 | 0 (off) |
| CH_M_R060 | CH_M_R030, | 0.8 | 0 (off) |
| CH_M_L090 | CH_M_L045, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L090 | CH_M_L030, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L090 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R090 | CH_M_R045, CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R090 | CH_M_R030, CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R090 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_L110 | CH_M_L135 | 1.0 | 0 (off) |
| CH_M_L110 | CH_M_L090 | 0.8 | 0 (off) |
| CH_M_L110 | CH_M_L045 | 0.8 | 0 (off) |
| CH_M_L110 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R110 | CH_M_R135 | 1.0 | 0 (off) |
| CH_M_R110 | CH_M_R090 | 0.8 | 0 (off) |
| CH_M_R110 | CH_M_R045 | 0.8 | 0 (off) |
| CH_M_R110 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_L135 | CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L135 | CH_M_L150 | 1.0 | 0 (off) |
| CH_M_L135 | CH_M_L090 | 0.8 | 0 (off) |
| CH_M_L135 | CH_M_L045 | 0.8 | 0 (off) |
| CH_M_L135 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R135 | CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R135 | CH_M_R150 | 1.0 | 0 (off) |
| CH_M_R135 | CH_M_R090 | 0.8 | 0 (off) |
| CH_M_R135 | CH_M_R045 | 0.8 | 0 (off) |
| CH_M_R135 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_L150 | CH_M_L135 | 1.0 | 0 (off) |
| CH_M_L150 | CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L150 | CH_M_L045 | 0.8 | 0 (off) |
| CH_M_L150 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R150 | CH_M_R135 | 1.0 | 0 (off) |
| CH_M_R150 | CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R150 | CH_M_R045 | 0.8 | 0 (off) |
| CH_M_R150 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_180 | CH_M_R150, CH_M_L150 | 1.0 | 0 (off) |
| CH_M_180 | CH_M_R135, CH_M_L135 | 1.0 | 0 (off) |
| CH_M_180 | CH_M_R110, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_180 | CH_M_R090, CH_M_L090 | 0.8 | 0 (off) |
| CH_M_180 | CH_M_R045, CH_M_L045 | 0.6 | 0 (off) |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | | |
|---|---|---|---|
| CH_M_180 | CH_M_R030, CH_M_L030 | 0.6 | 0 (off) |
| CH_U_000 | CH_U_L030, CH_U_R030 | 1.0 | 0 (off) |
| CH_U_000 | CH_M_L030, CH_M_R030 | 0.85 | 0 (off) |
| CH_U_L045 | CH_U_L030 | 1.0 | 0 (off) |
| CH_U_L045 | CH_M_L045 | 0.85 | 1 |
| CH_U_L045 | CH_M_L030 | 0.85 | 1 |
| CH_U_R045 | CH_U_R030 | 1.0 | 0 (off) |
| CH_U_R045 | CH_M_R045 | 0.85 | 1 |
| CH_U_R045 | CH_M_R030 | 0.85 | 1 |
| CH_U_L030 | CH_U_L045 | 1.0 | 0 (off) |
| CH_U_L030 | CH_M_L030 | 0.85 | 1 |
| CH_U_R030 | CH_U_R045 | 1.0 | 0 (off) |
| CH_U_R030 | CH_M_R030 | 0.85 | 1 |
| CH_U_L090 | CH_U_L030, CH_U_L110 | 1.0 | 0 (off) |
| CH_U_L090 | CH_U_L030, CH_U_L135 | 1.0 | 0 (off) |
| CH_U_L090 | CH_U_L045 | 0.8 | 0 (off) |
| CH_U_L090 | CH_U_L030 | 0.8 | 0 (off) |
| CH_U_L090 | CH_M_L045, CH_M_L110 | 0.85 | 2 |
| CH_U_L090 | CH_M_L030, CH_M_L110 | 0.85 | 2 |
| CH_U_L090 | CH_M_L030 | 0.85 | 2 |
| CH_U_R090 | CH_U_R030, CH_U_R110 | 1.0 | 0 (off) |
| CH_U_R090 | CH_U_R030, CH_U_R135 | 1.0 | 0 (off) |
| CH_U_R090 | CH_U_R045 | 0.8 | 0 (off) |
| CH_U_R090 | CH_U_R030 | 0.8 | 0 (off) |
| CH_U_R090 | CH_M_R045, CH_M_R110 | 0.85 | 2 |
| CH_U_R090 | CH_M_R030, CH_M_R110 | 0.85 | 2 |
| CH_U_R090 | CH_M_R030 | 0.85 | 2 |
| CH_U_L110 | CH_U_L135 | 1.0 | 0 (off) |
| CH_U_L110 | CH_U_L045 | 0.8 | 0 (off) |
| CH_U_L110 | CH_U_L030 | 0.8 | 0 (off) |
| CH_U_L110 | CH_M_L110 | 0.85 | 2 |
| CH_U_L110 | CH_M_L045 | 0.85 | 2 |
| CH_U_L110 | CH_M_L030 | 0.85 | 2 |
| CH_U_R110 | CH_U_R135 | 1.0 | 0 (off) |
| CH_U_R110 | CH_U_R045 | 0.8 | 0 (off) |
| CH_U_R110 | CH_U_R030 | 0.8 | 0 (off) |
| CH_U_R110 | CH_M_R110 | 0.85 | 2 |
| CH_U_R110 | CH_M_R045 | 0.85 | 2 |
| CH_U_R110 | CH_M_R030 | 0.85 | 2 |
| CH_U_L135 | CH_U_L110 | 1.0 | 0 (off) |
| CH_U_L135 | CH_U_L045 | 0.8 | 0 (off) |
| CH_U_L135 | CH_U_L030 | 0.8 | 0 (off) |
| CH_U_L135 | CH_M_L110 | 0.85 | 2 |
| CH_U_L135 | CH_M_L045 | 0.85 | 2 |
| CH_U_L135 | CH_M_L030 | 0.85 | 2 |
| CH_U_R135 | CH_U_R110 | 1.0 | 0 (off) |
| CH_U_R135 | CH_U_R045 | 0.8 | 0 (off) |
| CH_U_R135 | CH_U_R030 | 0.8 | 0 (off) |
| CH_U_R135 | CH_M_R110 | 0.85 | 2 |
| CH_U_R135 | CH_M_R045 | 0.85 | 2 |
| CH_U_R135 | CH_M_R030 | 0.85 | 2 |
| CH_U_180 | CH_U_R135, CH_U_L135 | 1.0 | 0 (off) |
| CH_U_180 | CH_U_R110, CH_U_L110 | 1.0 | 0 (off) |
| CH_U_180 | CH_M_180 | 0.85 | 2 |
| CH_U_180 | CH_M_R110, CH_M_L110 | 0.85 | 2 |
| CH_U_180 | CH_U_R030, CH_U_L030 | 0.8 | 0 (off) |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | | |
|---|---|---|---|
| CH_U_180 | CH_M_R030, CH_M_L030 | 0.85 | 2 |
| CH_T_000 | ALL_U | 1.0 | 3 |
| CH_T_000 | ALL_M | 1.0 | 4 |
| CH_L_000 | CH_M_000 | 1.0 | 0 (off) |
| CH_L_000 | CH_M_L030, CH_M_R030 | 1.0 | 0 (off) |
| CH_L_L045 | CH_M_L045 | 1.0 | 0 (off) |
| CH_L_L045 | CH_M_L030 | 1.0 | 0 (off) |
| CH_L_R045 | CH_M_R045 | 1.0 | 0 (off) |
| CH_L_R045 | CH_M_R030 | 1.0 | 0 (off) |
| CH_LFE2 | CH_LFE3 | 1.0 | 0 (off) |
| CH_LFE2 | CH_M_L030, CH_M_R030 | 1.0 | 0 (off) |
| CH_LFE3 | CH_LFE2 | 1.0 | 0 (off) |
| CH_LFE3 | CH_M_L030, CH_M_R030 | 1.0 | 0 (off) |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 97 — Normalized center frequencies of the 71 filter-bank bands**

| Normalized Frequency [0, 1] |
|---|
| 0.00458330 |
| 0.00083333 |
| 0.00208330 |
| 0.00587500 |
| 0.00979170 |
| 0.01429200 |
| 0.01979200 |
| 0.02700000 |
| 0.03541700 |
| 0.04262500 |
| 0.05675000 |
| 0.07237500 |
| 0.08800000 |
| 0.10362000 |
| 0.11925000 |
| 0.13487000 |
| 0.15050000 |
| 0.16612000 |
| 0.18175000 |
| 0.19737000 |
| 0.21300000 |
| 0.22862000 |
| 0.24425000 |
| 0.25988000 |
| 0.27550000 |
| 0.29113000 |
| 0.30675000 |
| 0.32238000 |
| 0.33800000 |
| 0.35363000 |
| 0.36925000 |
| 0.38488000 |
| 0.40050000 |
| 0.41613000 |
| 0.43175000 |
| 0.44738000 |
| 0.46300000 |
| 0.47863000 |
| 0.49425000 |
| 0.50987000 |
| 0.52550000 |
| 0.54112000 |
| 0.55675000 |
| 0.57237000 |
| 0.58800000 |
| 0.60362000 |
| 0.61925000 |
| 0.63487000 |
| 0.65050000 |
| 0.66612000 |
| 0.68175000 |
| 0.69737000 |
| 0.71300000 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| |
|---|
| 0.72862000 |
| 0.74425000 |
| 0.75987000 |
| 0.77550000 |
| 0.79112000 |
| 0.80675000 |
| 0.82237000 |
| 0.83800000 |
| 0.85362000 |
| 0.86925000 |
| 0.88487000 |
| 0.90050000 |
| 0.91612000 |
| 0.93175000 |
| 0.94737000 |
| 0.96300000 |
| 0.97454000 |
| 0.99904000 |

**Table 98 — Equalizer parameters**

| Equalizer | $P_f$ [Hz] | $P_Q$ | $P_g$[dB] | g [dB] |
|---|---|---|---|---|
| $G_{EQ,1}$ | 12000 | 0.3 | -2 | 1.0 |
| $G_{EQ,2}$ | 12000 | 0.3 | -3.5 | 1.0 |
| $G_{EQ,3}$ | 200,1300, 600 | 0.3, 0.5, 1.0 | -6.5, 1.8, 2.0 | 0.7 |
| $G_{EQ,4}$ | 5000, 1100 | 1.0, 0.8 | 4.5, 1.8 | -3.1 |
| $G_{EQ,5}$ | 35 | 0.25 | -1.3 | 1.0 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 99 — Vertically corresponding channels: Each row lists channels which are considered to be above/below each other.**

| | | |
|---|---|---|
| CH_L_000 | CH_M_000 | CH_U_000 |
| CH_L_L045 | CH_M_L030 | CH_U_L030 |
| CH_L_L045 | CH_M_L030 | CH_U_L045 |
| CH_L_L045 | CH_M_L045 | CH_U_L030 |
| CH_L_L045 | CH_M_L045 | CH_U_L045 |
| CH_L_L045 | CH_M_L060 | CH_U_L030 |
| CH_L_L045 | CH_M_L060 | CH_U_L045 |
| CH_L_R045 | CH_M_R030 | CH_U_R030 |
| CH_L_R045 | CH_M_R030 | CH_U_R045 |
| CH_L_R045 | CH_M_R045 | CH_U_R030 |
| CH_L_R045 | CH_M_R045 | CH_U_R045 |
| CH_L_R045 | CH_M_R060 | CH_U_R030 |
| CH_L_R045 | CH_M_R060 | CH_U_R045 |
| CH_M_180 | CH_U_180 | |
| CH_M_L090 | CH_U_L090 | |
| CH_M_L110 | CH_U_L110 | |
| CH_M_L135 | CH_U_L135 | |
| CH_M_L090 | CH_U_L110 | |
| CH_M_L090 | CH_U_L135 | |
| CH_M_L110 | CH_U_L090 | |
| CH_M_L110 | CH_U_L135 | |
| CH_M_L135 | CH_U_L090 | |
| CH_M_L135 | CH_U_L135 | |
| CH_M_R090 | CH_U_R090 | |
| CH_M_R110 | CH_U_R110 | |
| CH_M_R135 | CH_U_R135 | |
| CH_M_R090 | CH_U_R110 | |
| CH_M_R090 | CH_U_R135 | |
| CH_M_R110 | CH_U_R090 | |
| CH_M_R110 | CH_U_R135 | |
| CH_M_R135 | CH_U_R090 | |
| CH_M_R135 | CH_U_R135 | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

### 10.3.5 Audio signal processing

#### 10.3.5.1 General

The audio processing block of the format converter obtains time domain audio samples for $N_{\text{in}}$ channels from the core decoder and generates a downmixed time domain audio output signal consisting of $N_{\text{out}}$ channels.

The processing takes as input

— the audio data decoded by the core decoder,

— the static downmix matrix $\mathrm{M}_{\text{DMX}}$ returned by the initialization of the format converter,

It returns an $N_{\text{out}}$-channel time domain output signal for the OutConf channel configuration signaled during the initialization of the format converter.

The format converter operates on contiguous, non-overlapping frames of length $L = 2048$ time domain samples of the input audio signals and outputs one frame of $L$ samples per processed input frame of length $L$.

© ISO/IEC 2015 – All rights reserved

#### 10.3.5.2 T/F-transform (hybrid QMF analysis)

As the first processing step the converter transforms $L = 2048$ samples of the $N_{\text{in}}$ channel time domain input signal $\left[\tilde{y}^{v}_{\text{ch},1} \cdots \tilde{y}^{v}_{\text{ch},N_{in}}\right] = \tilde{\mathbf{y}}^{v}_{\text{ch}}$ to a hybrid QMF $N_{\text{in}}$ channel signal representation consisting of $L_n = 32$ QMF time slots (slot index $n$) and $K = 71$ frequency bands (band index $k$). A QMF analysis according to ISO/IEC 14496-3:2009, subclause 4.6.18.4, is performed first

$$\left[\hat{y}^{n,k}_{\text{ch},1} \cdots \hat{y}^{n,k}_{\text{ch},N_{in}}\right] = \hat{\mathbf{y}}^{n,k}_{\text{ch}} = \text{QmfAnalysis}\left(\tilde{\mathbf{y}}^{v}_{\text{ch}}\right) \quad \text{with} \quad 0 \le v < L \quad \text{and} \quad 0 \le n < L_n,$$

followed by a hybrid analysis

$$\left[y^{n,k}_{\text{ch},1} \cdots y^{n,k}_{\text{ch},N_{in}}\right] = \mathbf{y}^{n,k}_{\text{ch}} = \text{HybridAnalysis}\left(\hat{\mathbf{y}}^{n,k}_{\text{ch}}\right).$$

The hybrid filtering shall be carried out as specified in 8.6.4.3 of ISO/IEC 14496-3:2009 for the 10,20 bands configuration of parametric stereo, resulting in a 71-band hybrid QMF domain representation.

#### 10.3.5.3 Covariance analysis

Note that for clarity the frequency band parameter (superscript $k$) is omitted in the following equations if it is not required for the presentation.

Let $F$ be a monotonically increasing frame index denoting the current frame of input data, e.g. $\mathbf{y}^{F,n}_{\text{ch}} = \mathbf{y}^{n}_{\text{ch}}$ for frame $F$, starting at $F = 0$ for the first frame of input data after initialization of the format converter. An analysis frame of length $2L_n$ is formulated from the input hybrid QMF spectra as

$$\mathbf{y}^{F,n}_{\text{in, ch}} = \begin{cases} 0 & \text{for} \quad 0 \le n < L_n \quad , \quad F = 0 \\ \mathbf{y}^{F-1,n+L_n}_{\text{in, ch}} & \text{for} \quad 0 \le n < L_n \quad , \quad F > 0 \\ \mathbf{y}^{F,n-L_n}_{\text{ch}} & \text{for} \quad L_n \le n < 2L_n \quad , \quad F \ge 0 \end{cases}.$$

Note that $\mathbf{y}^{F,n}_{\text{in, ch}}$ is a row vector with $N_{\text{in}}$ elements in case of $N_{\text{in}}$ input channels. The covariance matrix is analysed from four quarter segments of $\mathbf{y}^{F,n}_{\text{in, ch}}$, so that

$$\mathbf{C}^{F}_{y,q} = \sum_{n=16q}^{16q+15} \left(\mathbf{y}^{F,n}_{\text{in, ch}}\right)^{\text{T}} \left(\mathbf{y}^{F,n}_{\text{in, ch}}\right)^{*} \quad \text{for} \quad q = 0,1,2,3,$$

where $(\cdot)^{\text{T}}$ denotes the transpose and $(\cdot)^{*}$ denotes the complex conjugate of a variable and $\mathbf{C}^{F}_{y,q}$ is an $N_{\text{in}}$ x $N_{\text{in}}$ matrix for each $q = 0,1,2,3$.

Note that $\mathbf{C}^{F}_{y,0}$ and $\mathbf{C}^{F}_{y,1}$ are the same as $\mathbf{C}^{F-1}_{y,2}$ and $\mathbf{C}^{F-1}_{y,3}$, correspondingly, and are not necessary to be re-calculated. The covariance matrices of the four quarter segments are added with center weighting assuming a staircase shape:

$$\mathbf{C}^{\text{F}}_{y,\text{sum}} = \mathbf{C}^{F}_{y,0} + 4\mathbf{C}^{F}_{y,1} + 4\mathbf{C}^{F}_{y,2} + \mathbf{C}^{F}_{y,3}.$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

The final estimation for the covariance matrix $\mathbf{C}_y^F$ is obtained by modifying the entries of $\mathbf{C}_{y,\mathrm{sum}}^F$ with a small channel dependent offset

$$C_{y,a,b} = (1 + 0.0002\sqrt{ab})C_{y,\mathrm{sum},a,b}$$

where the two indices in a notation $C_{y,a,b}$ denote the matrix element in the $a$ th row and $b$ th column of $\mathbf{C}_y$. From the covariance matrix $\mathbf{C}_y$ inter-channel correlation coefficients between the channels $A$ and $B$ are derived as

$$ICC_{A,B} = \sqrt{\frac{\left|C_{y,A,B}\right|^2}{\mathrm{eps} + C_{y,A,A} \cdot C_{y,B,B}}}\,.$$

#### 10.3.5.4 Phase-alignment matrix formulation

10.3.5.4.1 **General**

The $ICC_{A,B}$ values are mapped to an attraction measure matrix $\mathbf{T}$ with elements

$$T_{A,B} = \begin{cases} \min\left(PasMax, \max\left(0, PasCurveSlope \cdot ICC_{A,B} + PasCurveShift\right)\right) & \text{for} \quad A \neq B \\ \min\left(1, \max\left(0, 2.5 \cdot ICC_{A,B} - 1.2\right)\right) & \text{for} \quad A = B \end{cases},$$

where *PasMax, PasCurveSlope, PasCurveShift* are derived from Table 100

**Table 100 — Phase attraction mapping curve parameters.**
**phaseAlignStrength shall be set to 3 if no other value has been signaled in the bitstream.**
**If passiveDownmixFlag==1, then phaseAlignStrength shall be set to 0.**

| phaseAlignStrength | PasMax | PasCurveSlope | PasCurveShift |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0.0714 | 0.1701 | -0.0891 |
| 2 | 0.1548 | 0.3771 | -0.1896 |
| 3 | 0.25 | 0.625 | -0.3 |
| 4 | 0.3571 | 0.9184 | -0.4184 |
| 5 | 0.4762 | 1.2623 | -0.5427 |
| 6 | 0.6071 | 1.6624 | -0.6707 |
| 7 | 0.75 | 2.125 | -0.8 |

An intermediate phase-aligning mixing matrix $\mathbf{M}_{\mathrm{int}}$ is formulated. With

$P_{A,B} = T_{A,B} \cdot C_{y,A,B}$ and

$\mathbf{V} = \mathbf{M}_{\mathrm{DMX}}\mathbf{P}$

the matrix elements are derived as

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$M_{\mathrm{int},B,A} = \frac{M_{\mathrm{DMX},B,A}}{eps + \left|V_{B,A}\right|} V_{B,A} .$$

The intermediate phase-aligning mixing matrix $\mathbf{M}_{\mathrm{int}}$ is modified to avoid abrupt phase shifts, resulting in $\mathbf{M}_{\mathrm{mod}}$. This is a recursive process, running for each frame *F*, processing the frequency bands *k* in ascending order.

The regularization against phase shifts takes place in two stages: In the first stage, the regularization performs amplitude-weighted phase comparison against the previous frame, previous band, while also linking the phase-attracted channels. In the second stage, the regularization limits the update rate of the phase coefficients in comparison to the previous frame only.

Both regularization stages make use of a phase update limiting parameter, $\theta_{\mathrm{diff},A}^{F,k}$, which is formulated as function of an onset measure $o_A^{F,k}$ so that a low energy portion of a signal does not affect the phase processing after an onset:

$$o_A^{F,k} = \frac{C_{y,A,A}^{F,k}}{eps + C_{y,A,A}^{F,k} + C_{y,A,A}^{F-1,k}},$$

$$\theta_{\mathrm{diff},A}^{F,k} = \max(0.15, 20.3 o_A^{F,k} - 19)\pi .$$

10.3.5.4.2 **Regularization Stage 1:**

Stage 1 recursively takes into account comparison values $\mathbf{M}_{\mathrm{cmp}}$ from the last frame index (*F*-1) as well as for the last processing band (*k*-1). $\mathbf{M}_{\mathrm{cmp}}$ is derived from $\mathbf{M}_{\mathrm{mod}}$ at the end of the regularization process. The first step of regularization stage 1 combines the comparison data across frequency and time as follows:

if (F=0)

$$\mathbf{M}_{\mathrm{cmpFk}}^{F,k} = \begin{cases} 0 & \text{for} \quad k = 1 \\ \mathbf{M}_{\mathrm{cmp}}^{\mathrm{F,k\text{-}1}} & \text{for} \quad k > 1 \ \wedge \ k \neq 3 \\ \left(\mathbf{M}_{\mathrm{cmp}}^{\mathrm{F,k\text{-}1}}\right)^{*} & \text{for} \quad k = 3 \end{cases}$$

else (i.e. for F>0)

$$\mathbf{M}_{\mathrm{cmpFk}}^{F,k} = \begin{cases} \mathbf{M}_{\mathrm{cmp}}^{F-1,k} & \text{for} \quad k = 1 \\ \mathbf{M}_{\mathrm{cmp}}^{F-1,k} + \mathbf{M}_{\mathrm{cmp}}^{\mathrm{F,k\text{-}1}} & \text{for} \quad k > 1 \ \wedge \ k \neq 3 \\ \mathbf{M}_{\mathrm{cmp}}^{F-1,k} + \left(\mathbf{M}_{\mathrm{cmp}}^{\mathrm{F,k\text{-}1}}\right)^{*} & \text{for} \quad k = 3 \end{cases}$$

where the complex conjugate processing for the third band (*k*=3) accounts for the complex conjugate properties of the filterbank.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

The frequency index *k* is omitted in the following since the inter-band dependency is now contained in the matrix $\mathbf{M}_{\text{cmpFk}}^{F,k}$. The phase change of the current unregularized phase-aligning matrix $\mathbf{M}_{\text{int}}^{F}$ relative to $\mathbf{M}_{\text{cmpFk}}^{F,k}$ is measured by amplitude weighting with $\sqrt{C_{y,A,A}^{F}}$ and comparison against $\mathbf{M}_{\text{cmpFk}}^{F}$, forming $\mathbf{M}_{\text{timeFreq}}^{F}$ with elements

$$M_{\text{timeFreq},B,A}^{F} = M_{\text{cmpFk},B,A}^{F}\left(M_{\text{int},B,A}^{F}\right)^{*}\sqrt{C_{y,A,A}^{F}}\ .$$

To also take into account the interdependent channels in the regularization, the relevant entries are intermixed with the attraction matrix $\mathbf{T}^{F}$,

$$\mathbf{M}_{\text{timeFreqChan}}^{F} = \mathbf{M}_{\text{timeFreq}}^{F}\mathbf{T}^{F}\ .$$

The phase values of the elements of matrix $\mathbf{M}_{\text{timeFreqChans}}^{F}$ are

$$\theta_{\text{timeFreqChan},B,A}^{F} = \arg\left(M_{\text{timeFreqChan},B,A}^{F}\right).$$

To avoid constant phase offsets, $\theta_{\text{timeFreqChan},B,A}^{F}$ is adjusted towards zero by $\theta_{\text{diff},A}^{F}$:

$$\theta_{\text{timeFreqChanStage1},B,A}^{F} = \text{sign}\left(\theta_{\text{timeFreqChan},B,A}^{F}\right)\max\left(0, abs(\theta_{\text{timeFreqChan},B,A}^{F}) - \theta_{\text{diff},A}^{F}\right)$$

10.3.5.4.3 **Regularization Stage 2:**

In stage 2 of the regularization another phase comparison parameter, only across time, is formulated:

$$\theta_{\text{time},B,A}^{F} = \arg\left(M_{\text{PA},B,A}^{F-1}\left(M_{\text{int},B,A}^{F}\right)^{*}\right)$$

The final regularization parameter is such that is as close as possible to $\theta_{\text{timeFreqChanStage1},B,A}^{F}$, but not further than $\theta_{\text{diff},A}^{F}$ in respect to $\theta_{\text{time},B,A}^{F}$. Let *unwrap()* be a function that maps any angular parameter to the corresponding angle in the interval $-\pi \ldots \pi$. The final phase parameter is formulated as

$$\theta_{\text{update},B,A}^{F} = unwrap\left(\theta_{\text{timeFreqChanStage1},B,A}^{F} - \theta_{\text{time},B,A}^{F}\right),$$
$$\theta_{\text{mod},B,A}^{F} = \theta_{\text{time},B,A}^{F} + sign\left(\theta_{\text{update},B,A}^{F,k}\right)\max(0, abs(\theta_{\text{update},B,A}^{F}) - \theta_{\text{diff},A}^{F}),$$

and the modified, i.e. phase-regularized, mixing matrix elements are obtained as

$$M_{\text{mod},B,A}^{F} = M_{\text{int},B,A}^{F}\cdot\exp\left(j\cdot\theta_{\text{mod},B,A}^{F}\right).$$

Finally, $\mathbf{M}_{\text{cmp}}$ is derived by amplitude weighting the regularized downmixing coefficients,

$$M_{\text{cmp,B,A}}^{F} = M_{\text{mod},B,A}^{F}\sqrt{C_{y,A,A}^{F}}\ .$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

Note that $\mathbf{M}_{\text{cmp}}$ is used in the time and frequency recursive formulation of regularization stage 1.

10.3.5.4.4 **Energy Scaling:**

An energy scaling is applied to the mixing matrix to obtain the final phase-aligning mixing matrix $\mathbf{M}_{\text{PA}}$. With

$$\mathbf{M}_{Cy} = \mathbf{M}_{\text{mod}} \mathbf{C}_y \mathbf{M}_{\text{mod}}^{\text{H}},$$

where $(\cdot)^{\text{H}}$ denotes the conjugate transpose operator, and

$$S_B = \sqrt{\frac{\sum_{A=1}^{N_{in}} M_{\text{DMX},B,A} \cdot M_{\text{DMX},B,A} \cdot C_{y,A,A}}{\text{eps} + M_{Cy,B,B}}},$$

$$S_{\text{lim},B} = \min\left(S_{\max}, \max\left(S_{\min}, S_B\right)\right),$$

where the limits are defined as $S_{\max} = 10^{0.4}$ and $S_{\min} = 10^{-0.5}$, the final phase-aligning mixing matrix elements follow as

$$M_{\text{PA},B,A} = \left(S_{\text{lim},B} \cdot AES + (1 - AES)\right) \cdot M_{\text{mod},B,A},$$

where *AES* = (1 – **passiveDownmixFlag**).

### 10.3.5.5 Calculation of output data

The output signals for the current frame $F$ are formulated by linearly interpolating the mixing matrices from the previous frame to the current frame,

$$\mathbf{z}_{\text{ch}}^{F,n} = \left(\left(\frac{n+1}{L_n}\mathbf{M}_{\text{PA}}^{F} + \frac{L_n - n - 1}{L_n}\mathbf{M}_{\text{PA}}^{F-1}\right)\left(\mathbf{y}_{\text{in, ch}}^{F,n}\right)^{\text{T}}\right)^{\text{T}} \quad \text{for} \quad 0 \le n < L_n.$$

Note that the input audio for the above mixing procedure is the first half of the analysis window.

### 10.3.5.6 F/T-transform (hybrid QMF synthesis)

Note that the processing steps described above have to be carried out for each hybrid QMF band $k$ (recursively, for ascending $k$). In the following formulations the band index $k$ is reintroduced, i.e. $\mathbf{z}_{\text{ch}}^{F,n,k} = \mathbf{z}_{\text{ch}}^{F,n}$. The hybrid QMF frequency domain output signal $\mathbf{z}_{\text{ch}}^{F,n,k}$ is transformed to an $N_{\text{out}}$-channel time domain signal frame of length $L$ time domain samples per output channel $B$, yielding the time domain output signal $\tilde{\mathbf{z}}_{\text{ch}}^{F,\nu}$:

The hybrid synthesis

$$\hat{\mathbf{z}}_{\text{ch}}^{F,n,k} = \text{HybridSynthesis}\left(\mathbf{z}_{\text{ch}}^{F,n,k}\right)$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

is carried out as defined in Figure 8.21 of ISO/IEC 14496-3:2009, i.e. by summing the sub-subbands of the three lowest QMF subbands to obtain the three lowest QMF subbands of the 64band QMF representation. The subsequent QMF synthesis

$$\tilde{\mathbf{z}}_{\mathrm{ch}}^{F,\nu} = \mathrm{QMFSynthesis}\left(\hat{\mathbf{z}}_{\mathrm{ch}}^{F,n,k}\right)$$

shall be carried out as defined in ISO/IEC 14496-3:2009, subclause 4.6.18.4.

# 11 Immersive Loudspeaker Rendering / Format Conversion

## 11.1 Description

For the 5.0 and 5.1 channel output layouts, immersive loudspeaker rendering is chosen to provide overhead sound images using surround channel loudspeakers. The immersive loudspeaker renderer is a downmixer that converts multichannel signals from transmitted channel configurations with $N_{\mathrm{in}}$ channels to desired reproduction format of either 5.1 or 5.0 system. It has a switching scheme between 3D rendering and 2D rendering using different elevation rendering for height input channels depending on the transmitted bitstream **rednering3DType** to provide overhead sound image properly. It is thus also called 'immersive format converter'. The system consists of two major building blocks:

— An initialization algorithm that takes into account static parameters like the input and output format,

— a signal adaptive downmixing process that operates in a subband domain with a switching scheme according to the transmitted flag **rendering3DType**.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

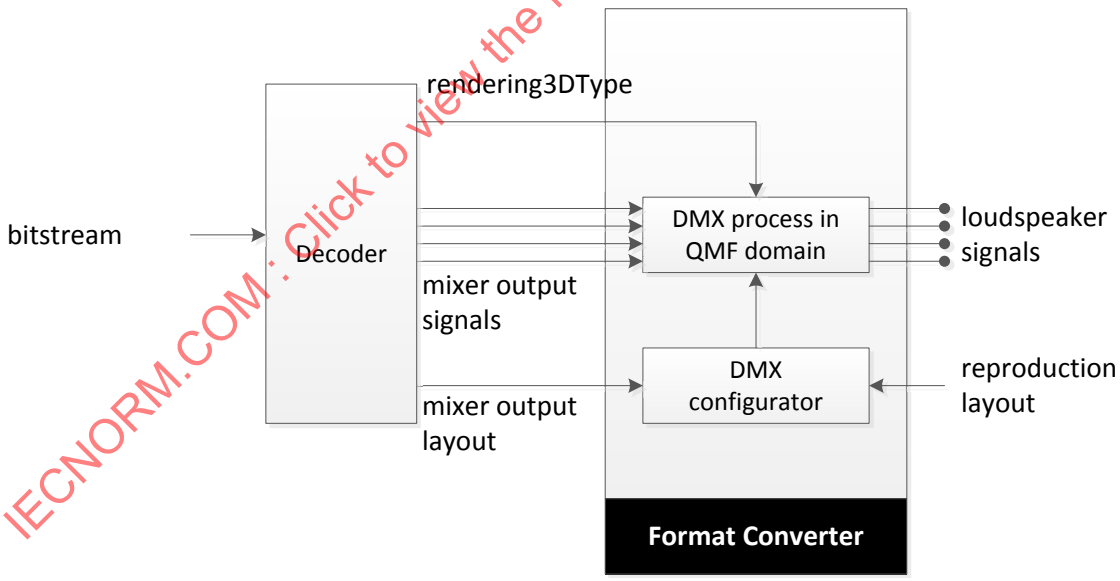



**Figure 30 — Main building blocks of the immersive format converter**

In the initialization phase the format converter automatically generates optimized downmixing parameters (like the downmixing matrix) for the given combination of input and output formats: It applies an algorithm that selects for each input loudspeaker the most appropriate mapping rule from a list of rules that has been designed to incorporate psychoacoustic considerations. Each rule describes the mapping from one input channel to one or several output loudspeaker channels.

Input channels are

© ISO/IEC 2015 – All rights reserved

— either mapped to a single output channel,

— or panned to two output channels,

— or (in case of the 'Voice of God' channel) distributed over a larger number of output cannels.

— or (in case of the 'elevation rendering') panned to multiple output channels with different panning coefficients over frequency.

The optimal mapping for each input channel is selected depending on the list of output loudspeakers that are available in the desired output format. Each mapping defines downmix gains for the input channel under consideration as well as potentially also an equalizer that is applied to the input channel under consideration.

Output setups with non-standard loudspeaker positions can be signaled to the system by providing the azimuth and elevation deviations from a regular loudspeaker setup. Further, distance variations of the desired target loudspeaker positions are taken into account.

For each frame, a bit called rendering3DType is decoded by the decoder and passed to the immersive format converter. The rendering3DType indicates whether the sound scene is appropriate for the 3D rendering or 2D rendering over either 5.0 or 5.1 channel layout. For the height input channels, the immersive format converter uses the "spatial elevation rendering" for 3D rendering when the rendering3DType is TRUE and "timbral elevation rendering" for the 2D rendering when the rendering3DType is FALSE. For the non-height input channels, non-elevation rendering uses the same downmix coefficients regardless of the rendering3DType.

The actual downmixing of the audio signals is performed on a hybrid QMF subband representation of the signals. The algorithm makes use of two mechanisms to avoid signal deteriorations like comb-filtering, coloration, or modulation artifacts:

— Phase-alignment of the multichannel input signals: Correlated input signals that differ in phase are aligned prior to downmixing them. The alignment process makes use of an attraction measure to only align the relevant channels for the relevant time-frequency tiles and to avoid modifications of other parts of the input signal. The alignment is further regularized to avoid artifacts due to rapid changes of the alignment phase modification terms. The phase-alignment improves the output signal quality by avoiding narrow spectral notches due to out-of-phase signal cancellations that could not be compensated for by energy normalization because of a limited frequency resolution. It further reduces the need of boosting signals in the energy preserving normalization, thus minimizes modulation artifacts. For the downmix cases with the 'elevation rendering', the frequency components in 2.8 ~ 10kHz are not aligned in order to provide the accurate synchronization of the rendered output signal. This stage is explained in 11.4.2.5.

— Normalization of the downmix process to preserve the input energies (except for the desired energy scaling that may be inherent in the downmix matrix).

## 11.2 Syntax

The FormatConverterFrame defines the proper rendering type for the immersive format converter.

**Table 101 — Syntax of FormatConverterFrame()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| FormatConverterFrame() | | |
| { | | |
| **rendering3DType;** | **1** | **uimsbf** |
| } | | |

The flag **rendering3DType** is created at the encoder based on the audio scene. When the audio scene is wideband and highly decorrelated at a frame, the flag **rendering3DType** becomes false and rendering is done

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

by the secondary downmix matrix $\mathbf{M}_{\mathrm{DMX2}}$. In all other cases, the flag becomes true and rendering is done by the primary downmix matrix $\mathbf{M}_{\mathrm{DMX}}$, which provides elevated sound images.

## 11.3 Definitions

### 11.3.1 General remarks

Audio signals that are fed into the format converter are referred to as *input signals* in the following. Audio signals that are the result of the format conversion process are referred to as *output signals*. Note that the audio input signals of the format converter are audio output signals of the core decoder.

Vectors and matrices are denoted by bold-faced symbols. Vector elements or matrix elements are denoted as italic variables supplemented by indices indicating the row/column of the vector/matrix element in the vector/matrix, e.g. $\left[y_1 \cdots y_a \cdots y_N\right] = \mathbf{y}$ denotes a vector and its elements. Similarly, $M_{a,b}$ denotes the element in the $a$ th row and $b$ th column of a matrix $\mathbf{M}$.

### 11.3.2 Variable definitions

| | |
|---|---|
| $N_{\mathrm{in}}$ | Number of channels in the input channel configuration. |
| $N_{\mathrm{out}}$ | Number of channels in the output channel configuration. |
| $\mathbf{M}_{\mathrm{DMX}}$ | Primary downmix matrix containing real-valued non-negative downmix coefficients (downmix gains) for the non-elevation rendering and spatial elevation rendering for 3D rendering over 2D layout. $\mathbf{M}_{\mathrm{DMX}}$ is basically of dimension ( $N_{\mathrm{out}}$ x $N_{\mathrm{in}}$ ) but possibly increased to ($N_{\mathrm{out}}$ x ($N_{\mathrm{in}}$+5)) depending on the input and output layouts. See how the dimension is changed in 11.4.2.3. |
| $\mathbf{M}_{\mathrm{DMX2}}$ | Secondary downmix matrix containing real-valued non-negative downmix coefficients (downmix gains) for the non-elevation rendering and timbral elevation rendering for 2D rendering over 2D layout. The downmix coefficients for the horizontal input channels are identical to those in $\mathbf{M}_{\mathrm{DMX}}$. The $\mathbf{M}_{\mathrm{DMX}}$ is dimension of ($N_{\mathrm{out}}$ x $N_{\mathrm{in}}$) but possibly increased to ($N_{\mathrm{out}}$ x ($N_{\mathrm{in}}$+5)) depending on the input and output layouts. See how the dimension is changed in 11.4.2.3. |
| $\mathbf{G}_{\mathrm{EQ}}$ | Matrix consisting of gain values per processing band determining frequency responses of equalizing filters for all rendering mapping. $\mathbf{G}_{\mathrm{EQ},1\sim5}$ are used for the non-elevation rendering and timbral elevation rendering, $\mathbf{G}_{\mathrm{EQ},7\sim14}$ are used for spatial elevation rendering, $\mathbf{G}_{\mathrm{EQ},15\sim20}$ are used for spatial coloration filter, and $\mathbf{G}_{\mathrm{EQ},21\sim}$ are used for modified EQ for randomized setup in 11.4.1.6.5 and coloration filter in 11.4.1.6.7.6. |
| $\mathbf{I}_{\mathrm{EQ}}$ | Vector signalling which equalizer filters to apply to the input channels (if any). |
| $L$ | Frame length measured in time domain audio samples. |
| $\nu$ | Time domain sample index. |
| $n$ | QMF time slot index (= subband sample index). |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $L_n$ | Frame length measured in QMF slots. |
| $F$ | Frame index (frame number). |
| $K$ | Number of hybrid QMF frequency bands, $K$ = 71. |
| $k$ | QMF band index (1..64) or hybrid QMF band index (1.. $K$ ). |
| $A, B$ | Channel indices. |
| eps | Numerical constant, $\text{eps} = 10^{-35}$ . |
| **rendering3DType** | Flag from the bitstream identifying the rendering type for the elevation rendering. True for the general audio scene and false for the highly decorrelated wideband scene (e.g. applause). Accordingly, the primary downmix matrix $\mathbf{M}_{\text{DMX}}$ is chosen when the **rendering3DType** is TRUE and the secondary downmnix matrix $\mathbf{M}_{\text{DMX2}}$ is chosen when the **rendering3DType** is FALSE in the immersive format converter. |
| $i_{in}$ | Label of the input channel to be rendered by immersive format converter (e.g. CH_U_000) |
| $G_{vH0,1\sim6}\ (i_{in})$ | Spatial elevation panning coefficients for the input channel ($i_{in}$) for the 2.8~10kHz in order to provide overhead image. Note that the coefficient is always normalized to preserve the input power after mixing. The number index represents output channels as shown in Table 110. |
| $G_{vL0,1\sim6}\ (i_{in})$ | Spatial elevation panning coefficients for the input channel ($i_{in}$) for below 2.8 kHz and above 10kHz. Note that the coefficient is always normalized to reserve the input power after mixing. The number index represents output channels as shown in Table 110. |
| COLOR_A_B | Tone coloration filter to the output at the azimuth of ±B degree from the input at the azimuth of ±A degree based on the ratio between HRTF at A and HRTF at B. It is determined by a frequency dependent dynamic cue which represents speaker-to-listener orientation. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

## 11.4 Processing

### 11.4.1 Initialization of the format converter

#### 11.4.1.1 General description of the initialization

The initialization of the format converter is carried out before processing of the audio samples delivered by the core decoder takes place.

The initialization takes into account as input parameters

- The sampling rate of the audio data to process.
- The channel configuration of the audio data to process with the format converter (number and geometric positions of input channels).


© ISO/IEC 2015 – All rights reserved

— The channel configuration of the desired output format (number and geometric positions of output channels).

— Optional: Parameters signaling the deviation of the output loudspeaker positions from a standard loudspeaker setup (random setup functionality).

It returns

— The primary frequency dependent downmix matrix $\mathrm{M}_{\mathrm{DMX}}$ that is applied in the audio signal processing of the format converter when the rendering3DType is TRUE. Note that the $\mathrm{M}_{\mathrm{DMX}}$ is independent variable in the Format Converter and shall not be taken into account in the core decoding process in 5.5.3.1.2

— The secondary frequency dependent downmix matrix $\mathbf{M}_{\mathrm{DMX2}}$ that is applied in the audio signal processing of the format converter when the rendering3DType is FALSE. The $\mathbf{M}_{\mathrm{DMX2}}$ is identical to $\mathbf{M}_{\mathrm{DMX}}$ if there is no 'height' input channel or 'spatial elevation rendering' is not possible.

The input parameters to the initialization algorithm are listed in Table 102.

**Table 102 — Format converter initialization input parameters**

| | |
|---|---|
| | Input format: number of channels and nominal channel setup geometry |
| | Output format: number of channels and nominal channel setup geometry |
| $f_s$ | Sampling frequency in Hertz. |
| $r_{\mathrm{azi},A}$ | For each output channel *A*, an azimuth angle is specified, determining the deviation from the standard format loudspeaker azimuth. |
| $r_{\mathrm{ele},A}$ | For each output channel *A*, an elevation angle is specified, determining the deviation from the standard format loudspeaker elevation. |

Table 103 lists the output parameters that are derived during the initialization of the format converter.

**Table 103 — Format converter initialization output parameters**

| | |
|---|---|
| $\mathbf{M}_{\mathrm{DMX}}$ | Primary Downmix matrix [linear gains] for spatial elevation rendering (for rendering3DType == 1) |
| $\mathbf{M}_{\mathrm{DMX2}}$ | Secondary Downmix matrix [linear gains] for timbral elevation rendering (for rendering3DType == 0) |

Note that the $\mathbf{M}_{\mathrm{DMX1}}$ and $\mathbf{M}_{\mathrm{DMX2}}$ include the same input-output downmix matrix for non-elevation rendering input channels.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

#### 11.4.1.2 Assignment of format converter channel labels to input/output format channels

The format converter initialization is based on a system of rules that are defined in terms of *format converter channel labels*, see Table 112. To allow the application of the initialization rules, the channel labels have to be assigned to the channels of the input and output formats. Each format converter channel label is associated with a segment of the surface of the unit sphere, as defined in Table 112. The segments are designed non-overlapping.

The assignment of channel labels to channels is done by geometrically matching the segments to the position data associated with the channels of the input and output formats. The azimuth and elevation angles in degrees of the position data associated with the channels shall be rounded towards the nearest integer number before performing the channel label assignment. Note that the *nominal* channel positions shall be

© ISO/IEC 2015 – All rights reserved

applied in the following matching to channel label sectors, i.e. the azimuth and elevation angles *without* taking into account potential angle deviations signalled in $r_{\mathrm{azi},A}$ and/or $r_{\mathrm{ele},A}$ .

For each channel that is not an LFE (low-frequency effects) channel:

If the nominal position of the current channel, defined by its azimuth angle and elevation angle, is within or on the border of one of the segments defined in Table 112 then:

> Assign the corresponding channel label (e.g. CH_M_L030) associated with the matching segment.
>
> Add the angle differences between the nominal position of the current channel and the nominal position associated with the matching segment (i.e. the angles in the second and third column of Table 112) to the angle deviations stored in $r_{\mathrm{azi},A}$ and $r_{\mathrm{ele},A}$.

Else (i.e. no matching sector found), then:

> Assign the CH_EMPTY label.

If an input or output format contains exactly one LFE channel, then the label CH_LFE1 shall be assigned to this channel.

If an input or output format contains exactly two LFE channels, then the labels CH_LFE1 and CH_LFE2 shall be assigned to the two LFE channels in the order that minimizes the maximum azimuth distance from the channels to the assigned CH_LFE1 and CH_LFE2 nominal azimuth positions.

If an input or output format contains more than 2 LFE channels, then those 2 LFE channels out of the considered setup shall be selected that minimize the maximum azimuth distance to the CH_LFE1 and CH_LFE2 nominal azimuth positions. The labels CH_LFE1 and CH_LFE2 shall be assigned as in the case of two LFE channels. The remaining LFE channels shall not be considered further in the calculation of downmix coefficients, i.e. the corresponding lines/columns of the downmix matrix shall remain filled with zeros.

#### 11.4.1.3 Handling for unknown input channels

If the label CH_EMPTY is assigned to an input channel, this channel shall be considered unknown to the rules-based initialization and the downmix coefficients for mapping this input channel to the output channels shall be derived as specified in 10.3.4.6.7.

#### 11.4.1.4 Handling for unknown output formats

If the output format contains at least one channel with the label CH_EMPTY assigned to it, or if at least one channel label is assigned to more than one channel of the output format, the output format shall be considered unknown and the derivation of the downmixing coefficients shall be carried out as specified in 11.4.1.6.9. The rules-based derivation of downmix coefficients shall not be applied for unknown output formats.

#### 11.4.1.5 Handling of deviations from standard loudspeaker positions

If the below conditions are not met, the rules-based initialization is considered to have failed, the ouput format shall be considered to be unknown, and the downmixing gains shall be obtained as defined in 11.4.1.6.9

The absolute values of $r_{\mathrm{azi},A}$ and $r_{\mathrm{ele},A}$ shall not exceed 35 and 55 degrees, respectively. The minimum angle between any loudspeaker pair (without LFE channels) shall not be smaller than 15 degrees.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

The values of $r_{\mathrm{azi},A}$ shall be such that the ordering by azimuth angles of the horizontal loudspeakers does not change. Likewise, the ordering of the height and low loudspeakers shall not change.

The values of $r_{\mathrm{ele},A}$ shall be such that the ordering by elevation angles of loudspeakers which are (approximately) above/below each other does not change. To verify this, the following procedure is applied:

For each row of **Table 117** which contains two or three channels of the output format, do:

— Order the channels by elevation without randomization

— Order the channels by elevation with considering randomization

— If the two orderings differ, return an initialization error

If the below conditions are not met, converter initialization is considered to have failed, and an error shall be returned.

#### 11.4.1.6 Rules-Based Initialization algorithm

11.4.1.6.1 **General**

The rules-based initialization algorithm is defined in the following. It shall not be applied if the output format is considered unknown as defined above. Note that the following description makes use of the intermediate parameters defined in Table 104 only for clarity reasons. An implementation may omit the introduction of these intermediate parameters.

**Table 104 — Format converter initialization intermediate parameters**

| | |
|---|---|
| $\mathbf{S}$, $\mathbf{S}^{\mathrm{P}}$, $\mathbf{S}^{\mathrm{S}}$ | Vector of converter source channels [input channel indices] |
| $\mathbf{D}$, $\mathbf{D}^{\mathrm{P}}$, $\mathbf{D}^{\mathrm{S}}$ | Vector of converter destination channels [output channel indices] |
| $\mathbf{G}$, $\mathbf{G}^{\mathrm{P}}$, $\mathbf{G}^{\mathrm{S}}$ | Vector of converter gains [linear] |
| $\mathbf{E}$, $\mathbf{E}^{\mathrm{P}}$, $\mathbf{E}^{\mathrm{S}}$ | Vector of converter EQ indices |
| $\mathbf{G}_{\mathrm{EQ}}$ | Matrix containing equalizer gain values for all EQ indices and frequency bands |

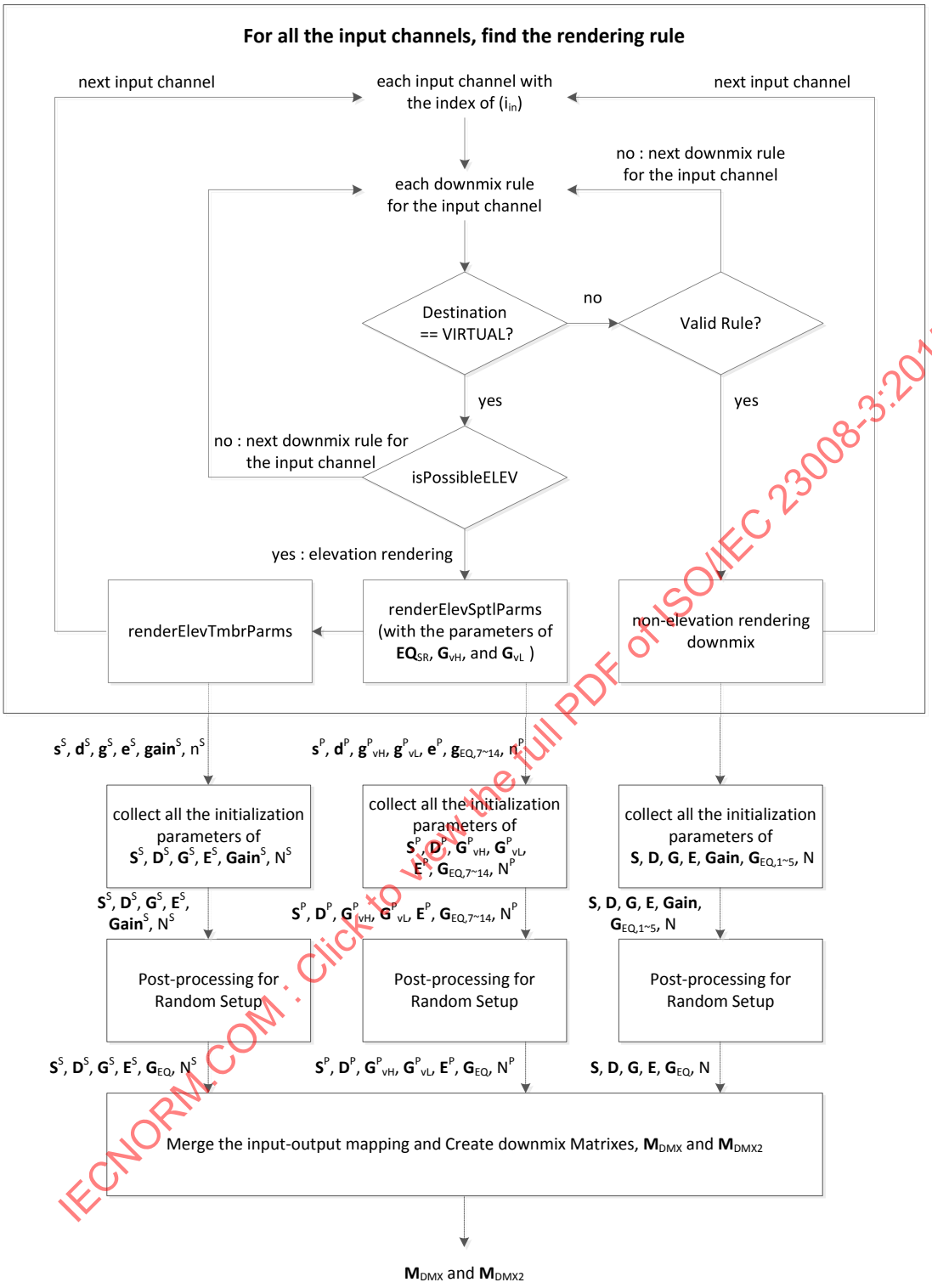
* The superscript S/P is the discriminator for the elevation rendering type. Those with P are initialized to be used for the 'spatial elevation rendering' and used to create the primary downmix matrix $\mathbf{M}_{\mathrm{DMX}}$, those with S are for the 'timbral elevation rendering' and used to create the secondary downmix matrix, and those without superscript are for the non-elevation rendering and used to create both the primary and secondary downmix matrixes.

The intermediate parameters describe the dowmixing parameters in a mapping-oriented way, i.e. as sets of parameters $S_i$, $D_i$, $G_i$, $E_i$ per mapping *i*.

The format converter initialization output parameters are derived as described in the following figure,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved




IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Figure 31 — Rule based downmix initialization flow**

, with following steps for each channel:

1. If the destination of the downmix rule for the input channel ($i_{in}$) is VIRTUAL, isPossibleElev decides whether the input channel should be rendered by the elevation rendering defined in 11.4.1.6.7.


© ISO/IEC 2015 – All rights reserved

A. If isPossibleElev returns TRUE,
   i. A set of parameters of $\mathbf{s}^{P}$, $\mathbf{d}^{P}$, $\mathbf{g}^{P}_{H}$, $\mathbf{g}^{P}_{L}$, $\mathbf{e}^{P}$, $\mathbf{G}_{EQ,eq(i_{in})}$, and $n^{P}$ and another set of parameters of $\mathbf{s}^{S}$, $\mathbf{d}^{S}$, $\mathbf{g}^{S}$, $\mathbf{e}^{S}$, $\mathbf{gain}^{S}$, and $n^{S}$ are initialized by the renderElevParms and renderTmbrParms, respectively. Note that the parameters of $n^{P}$ and $n^{S}$ indicates the number of output loudspeaker required for the input channel $(i_{in})$, $\mathbf{s}^{P}$, $\mathbf{d}^{P}$, $\mathbf{g}^{P}_{H}$, $\mathbf{g}^{P}_{L}$, and $\mathbf{e}^{P}$ are $n^{P}$ column vectors, $\mathbf{s}^{S}$, $\mathbf{d}^{S}$, $\mathbf{g}^{S}$, $\mathbf{e}^{S}$, and $\mathbf{gain}^{S}$ are $n^{S}$ column vectors, $\mathbf{G}_{EQ,eq(i_{in})}$ is a 71 row vector representing the EQ coefficients for 71 bands for the input channel $(i_{in})$.
   ii. The initialization parameters are collected among the group of parameters, each primary or secondary, and the downmix rules for next input channel are investigated until all the input channel mapping is found.
B. If isPossibleElev returns FALSE, the current downmix rule is ignored and the next downmix rule is investigated

2. If the destination is not VIRTUAL, the downmix rule is investigated whether the rule is valid checking the output layout includes all the channels in the destination column.
   A. If the downmix rule is valid,
      i. A set of parameters is initialized by the non-elevation rendering initialization and added directly to the **S**, **D**, **G**, **E**, and **Gain** as specified in 11.4.1.6.3 and to the $\mathbf{G}_{EQ}$ as specified in 11.4.1.6.4.
      ii. The downmix rules for next input channel are investigated until all the input channel mappings are found.
   B. If the downmix rule is invalid, the current downmix rule is ignored and the next downmix rule is investigated.
3. After collecting all the initializations parameters for the 'spatial elevation rendering,' 'timbral elevation rendering,' and 'non-elevation rendering,' post-processing for random setup shall be applied.
4. Create the primary downmix matrix combining the 'spatial elevation rendering' and 'non-elevation rendering' parameters and the secondary downmix matrix combining the 'timbral elevation rendering' and 'non-elevation rendering' parameters.

11.4.1.6.2 **Random setups Pre-Processing**

Random output loudspeaker setups, i.e. output setups that contain loudspeakers at positions deviating from the positions defined for the desired output format are signalled by specifying the loudspeaker position deviation angles as input parameters $r_{azi,A}$ and $r_{ele,A}$. The angle deviations are taken into account as a pre-processing step:

Modify in **Table 112** the channels' azimuth and elevation angles by adding $r_{azi,A}$ and $r_{ele,A}$ to the corresponding channels' azimuth and elevation angles.

11.4.1.6.3 **Derivation of input channel/output channel mapping parameters:**

The parameters vectors $\mathbf{S}$, $\mathbf{D}$, $\mathbf{G}$, $\mathbf{E}$ define the mapping of input channels to output channels. For each mapping *i* from an input channel to an output channel with non-zero downmix gain they define the downmix gain as well as an equalizer index that indicates which equalizer curve has to be applied to the input channel under consideration in mapping *i*.

The elements of the parameter vectors $\mathbf{S}$, $\mathbf{D}$, $\mathbf{G}$, $\mathbf{E}$ are derived by the following algorithm:

Initialize the mapping counter *i*: $i=1$, $i^{P}=1$, $i^{S}=1$;

Initialize the EQ counter *e*: $e=21$ (EQ slots for *e* from 1 to 5 are occupied by 11.4.1.6.4, that for *e* from 7 to 14 are occupied by **Table 109**, and that for *e* from 15 to 20 are occupied by **Table 106** . The EQ counter *e* will be shared in 11.4.1.6.5 and 11.4.1.6.7.6 in an incremental way)

For each input channel, ignoring channels with label CH_EMPTY assigned to them:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

If the input channel also exists in the output format (e.g. input channel under consideration is CH_M_R030 and channel CH_M_R030 exists in the output format), then:

$S_i$ = index of source channel in input *(**Example:** Example: channel CH_M_R030 in ChannelConfiguration 6 is at second place according to Table 113, i.e. has index 2 in this forma)*

$D_i$ = index of same channel in output

$G_i$ = 1.0

$E_i$ = 0

$i = i + 1$

Else (i.e. if the input channel does not exist in the output format)

search the first entry of this channel in the **Source** column of Table 114, for which the channels in the corresponding row of the **Destination** column exist. The **VIRTUAL** destination shall be considered valid if the **isPossibleElev** returns **TRUE**, which indicates the output format contains required channels for the elevation rendering of the input channel. The **isPossibleElev**, **renderElevSptlParms,** and **renderElevTmbrParms** are defined in 11.4.1.6.7. The ALL_U destination shall be considered valid (i.e. the relevant output channels exist) if the output format contains at least one “CH_U_” channel. The ALL_M destination shall be considered valid (i.e. the relevant output channels exist) if the output format contains at least one “CH_M_” channel. If for no entry in Table 114 corresponding to the input channel the channels in the **Destination** column exist, the rules-based initialization shall terminate and the downmix gains shall be derived according to 11.4.1.6.9.

If **Destination** column contains **VIRTUAL**, then:

$[\mathbf{s}^{\mathrm{P}}, \mathbf{d}^{\mathrm{P}}, \mathbf{g}^{\mathrm{P}}{}_{\mathrm{H}}, \mathbf{g}^{\mathrm{P}}{}_{\mathrm{L}}, \mathbf{e}^{\mathrm{P}}, n^{\mathrm{P}}]$= **renderElevSptlParms** (specified in 11.4.1.6.7.4)

for $n$ = 1 to $n^{\mathrm{P}}$

$m = i^{\mathrm{P}}$

$S^{\mathrm{P}}{}_m = s^{\mathrm{P}}{}_n(\mathrm{i_{in}})$

$D^{\mathrm{P}}{}_m = d^{\mathrm{P}}{}_n(\mathrm{i_{in}})$

$G^{\mathrm{P}}{}_{mH} = g^{\mathrm{P}}{}_{nH}(\mathrm{i_{in}})$

$G^{\mathrm{P}}{}_{mL} = g^{\mathrm{P}}{}_{nH}(\mathrm{i_{in}})$

$E^{\mathrm{P}}{}_m = e^{\mathrm{P}}{}_n(\mathrm{i_{in}})$

$C^{\mathrm{P}}{}_{mi} = (\mathrm{i_{in}})$

$C^{\mathrm{P}}{}_{mo}$ = label of the output channel $d^{\mathrm{P}}{}_n(\mathrm{i_{in}})$

$i^{\mathrm{P}} = i^{\mathrm{P}} + 1$

$[\mathbf{s}^{\mathrm{S}}, \mathbf{d}^{\mathrm{S}}, \mathbf{g}^{\mathrm{S}}, \mathbf{e}^{\mathrm{S}}, \mathbf{gain}, n^{\mathrm{S}}]$= **renderElevSptlParms** (described in 11.4.1.6.7.5)

for $n$ = 1 to $n^{\mathrm{S}}$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$m$ $= i^S$

$S^S{}_m$ $= s^S{}_n(\mathrm{i_{in}})$

$D^S{}_m$ $= d^S{}_n(\mathrm{i_{in}})$

$G^S{}_m$ $= g^S{}_n(\mathrm{i_{in}})$

$E^S{}_m$ $= e^S{}_n(\mathrm{i_{in}})$

$C^S{}_{mi}$ $= (\mathrm{i_{in}})$

$C^S{}_{mo}$ = label of the output channel $d^P{}_n(\mathrm{i_{in}})$

$Gain^S{}_m$ $= gain_n$

$i^S$ $= i^S + 1$

where $\mathrm{i_{in}}$ js the input channel label.

Else, if **Destination** column contains ALL_U, then:

For each output channel x with “CH_U_” in its name, do:

$S_i$ = index of source channel in input

$D_i$ = index of channel x in output

$G_i$ = (value of Gain column) / sqrt(number of “CH_U_” output channels)

$E_i$ = value of EQ column

$Gain_i$ = (value of Gain column)

$i = i + 1$

Else if **Destination** column contains ALL_M, then:

For each output channel x with “CH_M_” in its name, do:

$S_i$ = index of source channel in input

$D_i$ = index of channel x in output

$G_i$ = (value of Gain column) / sqrt(number of “CH_M_” output channels)

$E_i$ = value of EQ column

$Gain_i$ = (value of Gain column)

$i = i + 1$

Else If there is one channel in the **Destination** column, then:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$S_i$ = index of source channel in input

$D_i$ = index of destination channel in output

$G_i$ = value of Gain column

$E_i$ = value of EQ column

$Gain_i$ = (value of Gain column)

$i = i + 1$

Else (two channels in **Destination** column)

$S_i$ = index of source channel in input

$D_i$ = index of first destination channel in output

$G_i$ = (value of Gain column) * $g_1$

$E_i$ = value of EQ column

$Gain_i$ = (value of Gain column)

$i = i + 1$

$S_i = S_{i-1}$

$D_i$ = index of second destination channel in output

$G_i$ = (value of Gain column) * $g_2$

$E_i = E_{i-1}$

$Gain_i$ = (value of Gain column)

$i = i + 1$

The gains $g_1$ and $g_2$ are computed by applying tangent law amplitude panning in the following way:

— Unwrap source destination channel azimuth angles to be positive.

— The azimuth angles of the destination channels are $\alpha_1$ and $\alpha_2$ (see Table 112).

— The azimuth angle of the source channel ( = panning target) is $\alpha_{\text{src}}$.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

— $\alpha_0 = \frac{|\alpha_1 - \alpha_2|}{2}$

— $\alpha_{\text{center}} = \frac{\alpha_1 + \alpha_2}{2}$

— $\alpha = (\alpha_{\text{center}} - \alpha_{\text{src}}) \cdot \text{sgn}(\alpha_2 - \alpha_1)$

— $g_1 = \frac{g}{\sqrt{1+g^2}}, \quad g_2 = \frac{1}{\sqrt{1+g^2}}$ with $g = \frac{\tan\alpha_0 - \tan\alpha + 10^{-10}}{\tan\alpha_0 + \tan\alpha + 10^{-10}}$

—

11.4.1.6.4 **Derivation of equalizer gains $\mathbf{G}_{\text{EQ}}$:**

$\mathbf{G}_{\text{EQ}}$ consists of gain values per frequency band *k* and equalizer index *e*. The 5 predefined equalizers are combinations of different peak filters for $1 \le e \le 5$. Each equalizer is a serial cascade of one or more peak filters and a gain:

$$G_{\text{EQ},e}^{k} = 10^{\frac{g}{20}} \prod_{n=1}^{N} \text{peak}\left( band(k)\frac{f_s}{2}, P_{f,n}, P_{Q,n}, P_{g,n} \right)$$

where *band*(*k*) is the normalized center frequency of frequency band *k*, specified in Table 115, $f_s$ is the sampling frequency, and function *peak()* is for negative *G*

$$\text{peak}(b, f, Q, G) = \sqrt{\frac{b^4 + \left(\frac{1}{Q^2} - 2\right) f^2 b^2 + f^4}{b^4 + \left(\frac{10^{\frac{-G}{10}}}{Q^2} - 2\right) f^2 b^2 + f^4}}$$

and otherwise

$$\text{peak}(b, f, Q, G) = \sqrt{\frac{b^4 + \left(\frac{10^{\frac{G}{10}}}{Q^2} - 2\right) f^2 b^2 + f^4}{b^4 + \left(\frac{1}{Q^2} - 2\right) f^2 b^2 + f^4}}$$

The parameters for the equalizers are specified in Table 116.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

11.4.1.6.5 **Post-Processing for Random Setups**

Once the output parameters are computed, they are modified related to the specific random azimuth and elevations angles. This step has only to be carried out, if not all $r_{\text{ele},A}$ are zero. Definition of the post-processing algorithm for non-elevation rendering:

For each element *i* in $D_i$, do:

**if** output channel with index $D_i$ is a horizontal channel by definition (i.e. output channel label contains the label '_M_'), **and**

**if** this output channel is now a height channel (elevation in range 0..60 degrees), **and**

**if** input channel with index $S_i$ is a height channel (i.e. label contains '_U_'), **then**

—*h* = min(elevation of randomized output channel, 35) / 35
—$G_{comp} = h \cdot \frac{1}{Gain_i} + (1 - h)$
—Apply compensation gain to DMX gain: $G_i = G_i \cdot G_{\text{comp}}$
— Define new equalizer $\mathbf{G}_{\mathbf{EQ},e}$ with the index *e*, where $G^k_{EQ,e} = h + (1-h) \cdot G^k_{\text{EQ},E_i}$
— $E_i = e$
—$e = e + 1$

**else if** input channel with index $S_i$ is a horizontal channel (label contains '_M_')

— *h* = min(elevation of randomized output channel, 35) / 35
— Define new equalizer $\mathbf{G}_{\mathbf{EQ},e}$ with the index *e*,
where $G^k_{EQ,e} = \max\left(0.6310,\ \min\left(1.5849, \left(h \cdot IEQ(D_i)^k + (1-h)\right) \cdot G^k_{EQ,E_i}\right)\right)$
and $IEQ(D_i)^k$ is defined in Table 105
— $E_i = e$
— $e = e + 1$

**Table 105 — Inverse spatial elevation filter ($EQ_{0,lin}$ is defined in Table 109)**

| **Azimuth of $D_i$** | **Front Center (-15 15)** | **Front (-90 -15) or (15 90)** | **Side/Rear [-180 -90] or [90 180]** |
|---|---|---|---|
| $IEQ(D_i)^k$ | $\frac{1}{G^k_{EQ,9}}$ | $\frac{1}{G^k_{EQ,7}}$ | $\frac{1}{G^k_{EQ,8}}$ |

*Explanation of the post-processing steps defined above:*

*h* is a normalized elevation parameter indicating the elevation of a nominally horizontal output channel ('_M_') due to a random setup elevation offset $r_{\text{ele},A}$. For zero elevation offset *h*=0 follows and effectively no post-processing is applied.

The rule table (Table 114) in general applies a gain of the value in the gain column when mapping an upper input channel ('_U_' in channel label) to one or several horizontal output channels ('_M_' in channel label(s)). In case the output channel gets elevated due to a random setup elevation offset $r_{\text{ele},A}$, the gain is partially

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

(0<*h*<1) or fully (*h*=1) compensated for. Similarly the equalizer definitions fade towards a flat EQ-curve ( $G_{\mathrm{EQ},e}^{k} = const. = 1.0$ ) for *h* approaching *h* =1.

In case a horizontal input channel gets mapped to an output channel that gets elevated due to a random setup elevation offset $r_{\mathrm{ele},A}$, the equalizer $G_{EQ,E_i}^{k}$ fully applied and $IEQ(D_i)^k$, an inverse form of spatial elevation filter defined in Table 105, is partially (0<*h*<1) or fully (*h*=1) applied. As the spatial elevation filter is defined to provide the tone color of overhead loudspeakers on horizontal loudspeakers, the inverse of the spatial elevation filter is used to provide the tone color of horizontal loudspeakers on overhead loudspeakers. Note that the modified EQ is thresholded within the level of 4 dB, [0.6310, 1.5849].

11.4.1.6.6 **Spatial coloration filter for the horizontal input channels**

When a horizontal input channel at side or rear is panned by two output loudspeakers, e.g. CH_M_L090 is panned by CH_M_L030 and CH_M_L110, the tone color changes. In order to avoid such a change in tone color, a set of horizontal coloration filters of $\mathbf{G}_{\mathrm{EQ},15\sim20}$ are defined as in Table 106. Here, the filter name COLOR_A_B means the coloration filter for the output at the azimuth of ±B from the input at the azimuth of ±A. The filtering algorithm is:

For each element *i* in $\mathbf{S}$ do:

A : the magnitude of the azimuth of the input channel with index $S_i$

B : the magnitude of the azimuth of the output channel with index $D_i$

**If** both $S_i$ and $D_i$ are horizontal channels and there exists a filter COLOR_A_B_ in Table 106,

**If** the output channel has no deviation in azimuth and elevation, ( $r_{azi,D_i}$=0 and $r_{ele,D_i}$=0 )

If A==60 && B == 30 $E_i$ = 15 (COLOR_60_30)

Elseif A==90 && B == 30 $E_i$ = 16 (COLOR_90_30)

Elseif A==60 && B == 110 $E_i$ = 17 (COLOR_60_110)

Elseif A==90 && B == 110 $E_i$ = 18 (COLOR_90_110)

Elseif A==135 && B == 110 $E_i$ = 19 (COLOR_135_110)

Elseif A==180 && B == 110 $E_i$ = 20 (COLOR_180_110)

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Table 106 — Spatial coloration filters for horizontal channels**

| Hybrid QMF band | COLOR_180_110 $G_{EQ,20}$ | COLOR_090_030 $G_{EQ,16}$ | COLOR_060_110 $G_{EQ,17}$ | COLOR_135_110 $G_{EQ,19}$ | COLOR_090_110 $G_{EQ,18}$ | COLOR_060_030 $G_{EQ,15}$ |
|---|---|---|---|---|---|---|
| 0 | 1.2207277 | 0.82575887 | 1.0478644 | 1.0402648 | 0.9383601 | 0.92212301 |
| 1 | 1.2207277 | 0.82575887 | 1.0478644 | 1.0402648 | 0.9383601 | 0.92212301 |
| 2 | 1.2207277 | 0.82575887 | 1.0478644 | 1.0402648 | 0.9383601 | 0.92212301 |
| 3 | 1.2207277 | 0.82575887 | 1.0478644 | 1.0402648 | 0.9383601 | 0.92212301 |

© ISO/IEC 2015 – All rights reserved

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 0.79365081 | 1.1754434 | 1.01643 | 0.9288547 | 1.0755285 | 1.1108547 |
| 5 | 0.79365081 | 1.1754434 | 1.01643 | 0.9288547 | 1.0755285 | 1.1108547 |
| 6 | 0.79365081 | 1.1754434 | 1.01643 | 0.9288547 | 1.0755285 | 1.1108547 |
| 7 | 0.79365081 | 1.1585163 | 1.0149473 | 0.90169001 | 0.98411781 | 1.1948091 |
| 8 | 0.79365081 | 1.1585163 | 1.0149473 | 0.90169001 | 0.98411781 | 1.1948091 |
| 9 | 0.79365081 | 1.26 | 0.92721885 | 1.0003462 | 1.0241666 | 1.26 |
| 10 | 0.79365081 | 1.208599 | 0.81262708 | 0.90693218 | 0.87535268 | 1.1219938 |
| 11 | 0.79365081 | 0.97740591 | 1.0512857 | 1.0171721 | 0.97889137 | 1.0496904 |
| 12 | 0.85933852 | 0.84453988 | 1.26 | 0.97543412 | 1.2534994 | 0.90018034 |
| 13 | 0.90390807 | 0.94619727 | 1.26 | 0.91973215 | 1.26 | 1.0997185 |
| 14 | 0.88763344 | 1.0500977 | 1.26 | 0.86256438 | 1.26 | 1.1820639 |
| 15 | 0.80779493 | 1.0856131 | 1.26 | 0.79612881 | 1.26 | 1.26 |
| 16 | 0.80421317 | 0.99716073 | 1.26 | 0.79365081 | 1.26 | 1.1002132 |
| 17 | 0.85141277 | 0.89153987 | 1.26 | 0.79365081 | 1.26 | 0.98295653 |
| 18 | 0.83064753 | 0.94386107 | 1.26 | 0.79365081 | 1.26 | 1.0519516 |
| 19 | 0.81076753 | 0.99713677 | 1.26 | 0.79365081 | 1.26 | 1.1632811 |
| 20 | 0.79365081 | 1.1471872 | 1.26 | 0.79365081 | 1.26 | 1.26 |
| 21 | 0.79365081 | 1.26 | 1.26 | 0.79365081 | 1.26 | 1.26 |
| 22 | 0.79365081 | 1.26 | 1.26 | 0.79365081 | 1.26 | 1.26 |
| 23 | 0.79365081 | 1.26 | 1.26 | 0.79365081 | 1.26 | 1.26 |
| 24 | 0.79365081 | 1.26 | 1.1467814 | 0.79365081 | 1.26 | 1.26 |
| 25 | 0.79365081 | 1.26 | 1.0131826 | 0.79365081 | 1.233322 | 1.26 |
| 26 | 0.79365081 | 1.26 | 0.90711004 | 0.79365081 | 1.1852934 | 1.26 |
| 27 | 0.79365081 | 1.26 | 0.81770629 | 0.79365081 | 1.0953486 | 1.26 |
| 28 | 0.79365081 | 1.26 | 0.79365081 | 0.79365081 | 1.0236902 | 1.26 |
| 29 | 0.79365081 | 1.26 | 0.79365081 | 0.79365081 | 0.98883402 | 1.26 |
| 30 | 0.79365081 | 1.26 | 0.79365081 | 0.79365081 | 0.92739761 | 1.26 |
| 31 | 0.79365081 | 1.26 | 0.79365081 | 0.82225895 | 0.9255684 | 1.26 |
| 32 | 0.79365081 | 1.26 | 0.82704854 | 0.84583306 | 0.97202152 | 1.26 |
| 33 | 0.79365081 | 1.26 | 1.0166014 | 0.79365081 | 1.030863 | 1.26 |
| 34 | 0.79365081 | 1.26 | 1.2003248 | 0.79365081 | 1.1947373 | 1.26 |
| 35 | 0.79365081 | 1.26 | 1.2016704 | 0.79365081 | 1.26 | 1.26 |
| 36 | 0.79365081 | 1.26 | 1.0472132 | 0.79365081 | 1.26 | 1.26 |
| 37 | 0.79365081 | 1.26 | 0.92362517 | 0.79365081 | 1.218715 | 1.26 |
| 38 | 0.79365081 | 1.26 | 0.8078649 | 0.80314553 | 1.2225702 | 1.26 |
| 39 | 0.79365081 | 1.26 | 0.79365081 | 0.825122 | 1.1719567 | 1.1482208 |
| 40 | 0.79365081 | 1.26 | 0.79365081 | 0.82577235 | 1.0976481 | 0.95546353 |
| 41 | 0.79365081 | 1.26 | 0.79365081 | 0.82158899 | 1.0299088 | 0.81616014 |
| 42 | 0.79365081 | 1.26 | 0.79365081 | 0.81514698 | 0.98753756 | 0.79365081 |
| 43 | 0.79365081 | 1.1031394 | 0.79365081 | 0.84100527 | 0.99518955 | 0.79365081 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | | | | | |
|---|---|---|---|---|---|---|
| 44 | 0.79365081 | 0.95639628 | 0.79365081 | 0.89222193 | 1.0115879 | 0.79365081 |
| 45 | 0.79365081 | 0.79763001 | 0.79365081 | 0.88660526 | 0.97977269 | 0.79365081 |
| 46 | 0.79365081 | 0.79365081 | 0.89757115 | 0.85831517 | 0.99019438 | 0.79365081 |
| 47 | 0.79365081 | 0.79365081 | 1.0443652 | 0.86586368 | 1.0341145 | 0.79365081 |
| 48 | 0.79365081 | 0.79365081 | 1.1831371 | 0.84102261 | 1.0767646 | 0.79365081 |
| 49 | 0.79365081 | 0.79365081 | 1.26 | 0.86726952 | 1.1670161 | 0.79365081 |
| 50 | 0.79365081 | 0.79365081 | 1.26 | 0.89460015 | 1.26 | 0.79365081 |
| 51 | 0.79365081 | 0.79365081 | 1.26 | 0.80968457 | 1.26 | 0.79365081 |
| 52 | 0.79365081 | 0.79365081 | 1.26 | 0.79365081 | 1.26 | 0.80474436 |
| 53 | 0.79365081 | 0.79365081 | 1.26 | 0.79365081 | 1.26 | 0.87647426 |
| 54 | 0.79365081 | 0.87519455 | 1.26 | 0.79365081 | 1.26 | 0.93920386 |
| 55 | 0.79365081 | 0.98095393 | 1.26 | 0.79365081 | 1.26 | 0.99611974 |
| 56 | 0.79365081 | 1.0163895 | 1.26 | 0.79365081 | 1.26 | 0.97187245 |
| 57 | 0.79365081 | 1.0274936 | 1.26 | 0.79365081 | 1.26 | 0.93501246 |
| 58 | 0.79365081 | 1.0639353 | 1.26 | 0.79365081 | 1.26 | 0.95474315 |
| 59 | 0.79365081 | 1.1517589 | 1.26 | 0.79365081 | 1.26 | 1.0320153 |
| 60 | 0.79365081 | 1.22469 | 1.26 | 0.79365081 | 1.26 | 1.1006937 |
| 61 | 0.79365081 | 1.26 | 1.2404966 | 0.79365081 | 1.26 | 1.1356862 |
| 62 | 0.79365081 | 1.26 | 1.226367 | 0.79365081 | 1.26 | 1.1986256 |
| 63 | 0.79365081 | 1.26 | 1.2287724 | 0.79365081 | 1.26 | 1.26 |
| 64 | 0.79365081 | 1.26 | 1.223058 | 0.79365081 | 1.26 | 1.26 |
| 65 | 0.79365081 | 1.26 | 1.1695373 | 0.79365081 | 1.26 | 1.26 |
| 66 | 0.79365081 | 1.26 | 1.1292651 | 0.79365081 | 1.26 | 1.26 |
| 67 | 0.79365081 | 1.26 | 1.0702422 | 0.79365081 | 1.1744785 | 1.26 |
| 68 | 0.79365081 | 1.26 | 0.96692592 | 0.79365081 | 1.0516917 | 1.26 |
| 69 | 0.79365081 | 1.26 | 0.96692592 | 0.79365081 | 1.0516917 | 1.26 |
| 70 | 0.79365081 | 1.26 | 0.99395496 | 0.79365081 | 0.97151875 | 1.26 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

11.4.1.6.7 **Elevation rendering**

**11.4.1.6.7.1 General**

Elevation rendering provides overhead sound image using 5.1 channel layout. When the **Destination** column contains **VIRTUAL**, the **isPossibleElev** returns whether the input channel shall be rendered by the elevation rendering by comparing the output channel configuration and the required output channels for the spatial elevation rendering.

© ISO/IEC 2015 – All rights reserved

**Table 107 — Detail information of the Elevation rendering initialization parameters**

| | | |
|---|---|---|
| $\mathbf{S}^{P}$ | Vector of converter source channels [input channel indices] | Initialization parameters for spatial elevation rendering (renderElevSptlParms) |
| $\mathbf{D}^{P}$ | Vector converter destination channels [output channel indices] | |
| $\mathbf{G}^{P}_{H}$ | Vector of converter gains for 2.8~10kHz components | |
| $\mathbf{G}^{P}_{L}$ | Vector of converter gains below 2.8kHz and above 10kHz components | |
| $\mathbf{E}^{P}$ | Vector of converter EQ indices | |
| $\mathbf{S}^{S}$ | Vector of converter source channels [input channel indices] | Initialization parameters for timbral elevation rendering (renderElevTmbrParms) |
| $\mathbf{D}^{S}$ | Vector converter destination channels [output channel indices] | |
| $\mathbf{G}^{S}$ | Vector of converter gains [linear] | |
| $\mathbf{E}^{S}$ | Vector of converter EQ indices | |
| $\mathbf{G}_{EQ}$ | Matrix containing equalizer gain values for all EQ indices and frequency bands | EQ matrix for all rendering types |

When **isPossibleElev** is **TRUE**, two sets of initialization parameters for the spatial elevation rendering and timbral elevation rendering is initialized using **renderElevSptlParms** and **renderElevTmbrParms**. The parameters of $\mathbf{S}^{P}$, $\mathbf{D}^{P}$, $\mathbf{G}^{P}_{H}$, $\mathbf{G}^{P}_{L}$, and $\mathbf{E}^{P}$ initialized by **renderElevSptlParms** are for the spatial elevation rendering, and $\mathbf{S}^{S}$, $\mathbf{D}^{S}$, $\mathbf{G}^{S}$, and $\mathbf{E}^{S}$ by **renderElevTmbrParms** are for the timbral elevation rendering. The intermediate parameters describe the dowmixing parameters in a mapping-oriented way, i.e. as sets of parameters $S^{P}_{i}$, $D^{P}_{i}$, $G^{P}_{iH}$, $G^{P}_{iL}$, and $E^{P}_{i}$ per mapping *i*.

#### 11.4.1.6.7.2 isPossibleElev : Decision whether elevation rendering is valid for the output layout

The **isPossibleElev** returns **TRUE** when all the required output channels exist. The required output channels are defined in Table 108, indicating 1 for the required channel and 0 for the unnecessary channel. For example, CH_M_L030, CH_M_L110, and CH_M_R110 are required in order to use elevation rendering for the input channel CH_U_L030. If any of the required output channels for the input channel is not in the output channel configuration, **isPossibleElev** returns **FALSE** and elevation rendering is not applied for the input channel.

**Table 108 — Required Output Channels for Elevation Rendering for isPossibleElev**

| Input Channel | Required Output Channels | | | | | |
|---|---|---|---|---|---|---|
| | CH_M_L030 | CH_M_R030 | CH_M_000 | CH_LFE1 | CH_M_L110 | CH_M_R110 |
| CH_U_000 | 1 | 1 | 1 | 0 | 1 | 1 |
| CH_U_L045 | 1 | 1 | 0 | 0 | 1 | 1 |
| CH_U_R045 | 1 | 1 | 0 | 0 | 1 | 1 |
| CH_U_L030 | 1 | 1 | 0 | 0 | 1 | 1 |
| CH_U_R030 | 1 | 1 | 0 | 0 | 1 | 1 |
| CH_U_L090 | 1 | 0 | 0 | 0 | 1 | 1 |
| CH_U_R090 | 0 | 1 | 0 | 0 | 1 | 1 |
| CH_U_L110 | 1 | 0 | 0 | 0 | 1 | 1 |
| CH_U_R110 | 0 | 1 | 0 | 0 | 1 | 1 |
| CH_U_L135 | 1 | 0 | 0 | 0 | 1 | 1 |
| CH_U_R135 | 0 | 1 | 0 | 0 | 1 | 1 |
| CH_U_180 | 1 | 1 | 0 | 0 | 1 | 1 |
| CH_T_000 | 1 | 1 | 1 | 0 | 1 | 1 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 11.4.1.6.7.3 initElevSptlParms : Initialization of elevation rendering parameters based on the input channel elevation

The initial values of the spatial elevation filters and the elevation panning coefficients are defined in Table 109, Table 110, and Table 111 for the 'height' input channels, except CH_T_000. When a 'height' input channel except CH_T_000 has the elevation higher than 35 degree, the spatial elevation filter coefficients and elevation panning coefficients shall be updated according to the degree of the elevation. Note that the elevation sector is defined from +21 to +60 degree for the height channels in Table 114.

For the elevation rendering of an input channel, an EQ is selected from the EQ column of the Table 114. For convenience, $eq(i_{in})$ is defined as the EQ column, e.g. if the $i_{in}$ is CH_U_000, $eq(i_{in})$ is 9 (EQVFC).

**Table 109 — Spatial elevation filter initial values (for the 35 degree in elevation)**

| Hybrid QMF (71 bands) | $EQ_{0,lin}(.)$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 (EQVF) | 8 (EQVB) | 9 (EQVFC) | 10 (EQVBC) | 11 (EQVOG) | 12 (EQVS) | 13 (EQBTM) | 14 (EQVBA) |
| 0 | 0.841395 | 0.819093 | 0.68508532 | 0.770312 | 0.62528018 | 0.877674 | 0.922571 | 0.877674 |
| 1 | 0.841395 | 0.819093 | 0.68508532 | 0.770312 | 0.62528018 | 0.877674 | 0.922571 | 0.877674 |
| 2 | 0.874312 | 0.922571 | 0.71188768 | 0.940445 | 0.62528018 | 0.901571 | 1.143756 | 0.988553 |
| 3 | 0.874312 | 0.922571 | 0.71188768 | 0.940445 | 0.62528018 | 0.901571 | 1.143756 | 0.988553 |
| 4 | 0.926119 | 0.980995 | 0.83 | 1.015469 | 0.67516235 | 1 | 1.117721 | 1.051155 |
| 5 | 0.926119 | 0.980995 | 0.83 | 1.015469 | 0.67516235 | 1 | 1.117721 | 1.051155 |
| 6 | 0.944061 | 0.887837 | 0.76867877 | 1 | 0.72902365 | 0.905038 | 1.023293 | 0.951335 |
| 7 | 0.944061 | 0.887837 | 0.76867877 | 1 | 0.72902365 | 0.905038 | 1.023293 | 0.951335 |
| 8 | 0.944061 | 0.887837 | 0.76867877 | 1 | 0.72902365 | 0.905038 | 1.023293 | 0.951335 |
| 9 | 0.908518 | 0.958665 | 0.76867877 | 0.887837 | 0.70157643 | 1.039122 | 0.922571 | 1.027228 |
| 10 | 0.908518 | 0.958665 | 0.76867877 | 0.887837 | 0.70157643 | 1.039122 | 0.922571 | 1.027228 |
| 11 | 0.944061 | 0.936843 | 0.79875133 | 0.838172 | 0.72902365 | 1.023293 | 0.940445 | 0.860994 |
| 12 | 0.944061 | 0.936843 | 0.79875133 | 0.838172 | 0.72902365 | 1.023293 | 0.940445 | 0.860994 |
| 13 | 0.944061 | 0.936843 | 0.79875133 | 0.838172 | 0.72902365 | 1.023293 | 0.940445 | 0.860994 |
| 14 | 0.980995 | 0.962351 | 0.83 | 0.922571 | 0.64974255 | 0.940445 | 0.940445 | 0.992354 |
| 15 | 0.980995 | 0.962351 | 0.83 | 0.922571 | 0.64974255 | 0.940445 | 0.940445 | 0.992354 |
| 16 | 0.980995 | 0.962351 | 0.83 | 0.922571 | 0.64974255 | 0.940445 | 0.940445 | 0.992354 |
| 17 | 0.980995 | 0.962351 | 0.83 | 0.922571 | 0.64974255 | 0.940445 | 0.940445 | 0.992354 |
| 18 | 0.887153 | 0.916638 | 0.72234319 | 0.916638 | 0.63446387 | 0.9525 | 0.829592 | 0.90963 |
| 19 | 0.875087 | 0.899232 | 0.71251848 | 0.899232 | 0.62583427 | 0.934412 | 0.813839 | 0.892357 |
| 20 | 0.933012 | 0.863327 | 0.7596824 | 0.830823 | 0.66726037 | 0.917998 | 0.883436 | 0.856726 |
| 21 | 0.922109 | 0.849236 | 0.75080472 | 0.817263 | 0.65946291 | 0.903015 | 0.869017 | 0.842743 |
| 22 | 0.912036 | 0.836276 | 0.74260266 | 0.804791 | 0.65225907 | 0.889235 | 0.855756 | 0.829882 |
| 23 | 0.902688 | 0.824301 | 0.73499156 | 0.793266 | 0.64557314 | 0.876501 | 0.843501 | 0.817998 |
| 24 | 0.893979 | 0.816315 | 0.72790004 | 0.813189 | 0.61527373 | 0.864685 | 0.922978 | 0.874698 |
| 25 | 0.885822 | 0.805907 | 0.72125921 | 0.802821 | 0.60966047 | 0.853661 | 0.91121 | 0.863545 |
| 26 | 0.878166 | 0.796173 | 0.71502508 | 0.793123 | 0.60439092 | 0.843349 | 0.900203 | 0.853114 |
| 27 | 0.90502 | 0.802272 | 0.7368906 | 0.78401 | 0.64724144 | 0.866273 | 0.93538 | 0.917603 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 28 | 0.897936 | 0.793493 | 0.7311221 | 0.775431 | 0.6421749 | 0.856795 | 0.925145 | 0.907563 |
| 29 | 0.7943 | 0.785196 | 0.64673932 | 0.815915 | 0.56805749 | 0.815915 | 1.003794 | 0.918991 |
| 30 | 0.788618 | 0.777344 | 0.6421129 | 0.807755 | 0.56399394 | 0.807755 | 0.993756 | 0.909801 |
| 31 | 0.783206 | 0.738062 | 0.63770643 | 0.710275 | 0.53903577 | 0.769886 | 0.84093 | 0.790848 |
| 32 | 0.778048 | 0.731266 | 0.63350663 | 0.703734 | 0.53548544 | 0.762797 | 0.833186 | 0.783566 |
| 33 | 0.773116 | 0.724785 | 0.62949109 | 0.697497 | 0.53209122 | 0.756036 | 0.825802 | 0.776621 |
| 34 | 0.659052 | 0.545115 | 0.53661741 | 0.483973 | 0.48977268 | 0.566441 | 0.691546 | 0.584101 |
| 35 | 0.655171 | 0.540626 | 0.53345677 | 0.479988 | 0.48688833 | 0.561776 | 0.685851 | 0.579291 |
| 36 | 0.651447 | 0.536328 | 0.53042478 | 0.476172 | 0.48412123 | 0.557311 | 0.6804 | 0.574686 |
| 37 | 0.647865 | 0.532204 | 0.52750816 | 0.47251 | 0.48145865 | 0.553025 | 0.675167 | 0.570266 |
| 38 | 0.644418 | 0.528244 | 0.52470193 | 0.468995 | 0.47889724 | 0.54891 | 0.670144 | 0.566024 |
| 39 | 0.641096 | 0.524436 | 0.52199696 | 0.465614 | 0.47642896 | 0.544953 | 0.665313 | 0.561943 |
| 40 | 0.637889 | 0.520767 | 0.51938578 | 0.462356 | 0.39127933 | 0.54114 | 0.660658 | 0.558012 |
| 41 | 0.634793 | 0.517232 | 0.51686507 | 0.459218 | 0.38937988 | 0.537467 | 0.656173 | 0.554224 |
| 42 | 0.631798 | 0.513818 | 0.5144257 | 0.456187 | 0.38754274 | 0.53392 | 0.651843 | 0.550566 |
| 43 | 0.628901 | 0.510523 | 0.51206767 | 0.453261 | 0.3857659 | 0.530496 | 0.647662 | 0.547035 |
| 44 | 0.626093 | 0.507335 | 0.50978102 | 0.450431 | 0.38404333 | 0.527183 | 0.643618 | 0.543619 |
| 45 | 0.623373 | 0.504251 | 0.50756575 | 0.447693 | 0.38237503 | 0.523978 | 0.639706 | 0.540315 |
| 46 | 0.620731 | 0.501262 | 0.50541522 | 0.445039 | 0.3807543 | 0.520873 | 0.635914 | 0.537112 |
| 47 | 0.618168 | 0.498367 | 0.5033286 | 0.442469 | 0.37918248 | 0.517864 | 0.632241 | 0.53401 |
| 48 | 0.615676 | 0.495557 | 0.50129925 | 0.439974 | 0.37765354 | 0.514944 | 0.628675 | 0.530998 |
| 49 | 0.613254 | 0.49283 | 0.49932717 | 0.437553 | 0.37616815 | 0.51211 | 0.625216 | 0.528077 |
| 50 | 0.610896 | 0.490179 | 0.49740738 | 0.435199 | 0.37472162 | 0.509356 | 0.621853 | 0.525236 |
| 51 | 0.608602 | 0.487604 | 0.49553988 | 0.432913 | 0.37331462 | 0.50668 | 0.618586 | 0.522477 |
| 52 | 0.606366 | 0.485097 | 0.49371886 | 0.430687 | 0.37194246 | 0.504075 | 0.615406 | 0.519791 |
| 53 | 0.604187 | 0.482659 | 0.49194432 | 0.428522 | 0.37060648 | 0.501541 | 0.612313 | 0.517178 |
| 54 | 0.602061 | 0.480282 | 0.49021294 | 0.426412 | 0.36930199 | 0.499072 | 0.609298 | 0.514632 |
| 55 | 0.599987 | 0.477968 | 0.48852472 | 0.424358 | 0.36803033 | 0.496667 | 0.606362 | 0.512152 |
| 56 | 0.597962 | 0.475711 | 0.48687551 | 0.422354 | 0.36678748 | 0.494321 | 0.603498 | 0.509733 |
| 57 | 0.595984 | 0.47351 | 0.48526614 | 0.4204 | 0.36557478 | 0.492034 | 0.600706 | 0.507375 |
| 58 | 0.594051 | 0.47136 | 0.48369163 | 0.418491 | 0.36438888 | 0.489801 | 0.59798 | 0.505072 |
| 59 | 0.592162 | 0.469263 | 0.48215364 | 0.416629 | 0.36322978 | 0.487622 | 0.595319 | 0.502825 |
| 60 | 0.590313 | 0.467213 | 0.48064802 | 0.414809 | 0.36209614 | 0.485492 | 0.592718 | 0.500628 |
| 61 | 0.588505 | 0.465211 | 0.4791756 | 0.413032 | 0.36098729 | 0.483411 | 0.590178 | 0.498483 |
| 62 | 0.586734 | 0.463252 | 0.47773389 | 0.411293 | 0.35990055 | 0.481376 | 0.587694 | 0.496384 |
| 63 | 0.585001 | 0.461338 | 0.47632289 | 0.409593 | 0.35883726 | 0.479386 | 0.585265 | 0.494332 |
| 64 | 0.583302 | 0.459463 | 0.47493928 | 0.407929 | 0.35779541 | 0.477439 | 0.582887 | 0.492324 |
| 65 | 0.581638 | 0.45763 | 0.47358472 | 0.406301 | 0.356775 | 0.475533 | 0.58056 | 0.490359 |
| 66 | 0.580006 | 0.455833 | 0.47225589 | 0.404705 | 0.35577402 | 0.473666 | 0.578281 | 0.488434 |
| 67 | 0.578407 | 0.454074 | 0.47095362 | 0.403144 | 0.35479247 | 0.471838 | 0.57605 | 0.486549 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 68 | 0.576837 | 0.45235 | 0.46967542 | 0.401613 | 0.35382968 | 0.470046 | 0.573862 | 0.484701 |
| 69 | 0.575697 | 0.451098 | 0.46874665 | 0.400502 | 0.3531302 | 0.468746 | 0.572275 | 0.48336 |
| 70 | 0.573327 | 0.448501 | 0.46681773 | 0.398196 | 0.35167697 | 0.466048 | 0.56898 | 0.480578 |

When the 'height' input channel $i_{in}$ has higher elevation than 35 degree, the spatial elevation filter is calculated from $EQ^{k}_{0,lin}(eq(i_{in}))$ in Table 109 by

If the input channel, $i_{in}$, is frontal side : azimuth is in the range of (-90, 90)

$$EQ^{k}_{1,db}(eq(i_{in})) = \begin{cases} 20 \times \log_{10}\left(EQ^{k}_{0,lin}(eq(i_{in}))\right) + 0.05 \times \log_2(f_k \times f_s/6000) + 1 & \text{for } f_k \times f_s > 8000 \\ 20 \times \log_{10}\left(EQ^{k}_{0,lin}(eq(i_{in}))\right) + 1 & \text{otherwise} \end{cases}$$

$$EQ^{k}_{2,db}(eq(i_{in})) = EQ^{k}_{1,db}(eq(i_{in})) \times (1 + \min(\max(elv - 35, 0), 25) \times 0.05)$$

$$G^{k}_{EQ,eq(i_{in})} = \begin{cases} 10^{\left(EQ^{k}_{2,db}(eq(i_{in}))-1\right)/20 - 0.05 \times \log_2(f_k \times f_s/6000)} & \text{for } f_k \times f_s > 8000 \\ 10^{\left(EQ^{k}_{2,db}(eq(i_{in}))-1\right)/20} & \text{otherwise} \end{cases}$$

else (the input channel is rear side : azimuth is either in [-180, -90] or in [90, 180])

$$EQ^{k}_{1,db}(eq(i_{in})) = \begin{cases} 20 \times \log_{10}\left(EQ^{k}_{0,lin}(eq(i_{in}))\right) + 0.07 \times \log_2(f_k \times f_s/6000) + 1 & \text{for } f_k \times f_s > 8000 \\ 20 \times \log_{10}\left(EQ^{k}_{0,lin}(eq(i_{in}))\right) + 1 & \text{otherwise} \end{cases}$$

$$EQ^{k}_{2,db}(eq(i_{in})) = EQ^{k}_{1,db}(eq(i_{in})) \times (1 + \min(\max(elv - 35, 0), 25) \times 0.05)$$

$$G^{k}_{EQ,eq(i_{in})} = \begin{cases} 10^{\left(EQ^{k}_{2,db}(eq(i_{in}))-1\right)/20 - 0.07 \times \log_2(f_k \times f_s/6000)} & \text{for } f_k \times f_s > 8000 \\ 10^{\left(EQ^{k}_{2,db}(eq(i_{in}))-1\right)/20} & \text{otherwise} \end{cases}$$

where $f_k$ is the normalized center frequency of frequency band $k$, specified in Table 115, $f_s$ is the sampling frequency, and elv is the elevation of the input channel.

When the 'height' input channel $i_{in}$ does not have higher elevation than 35 degree, use the initial filter coefficients by:

$$G^{k}_{EQ,eq(i_{in})} = EQ^{k}_{0,lin}(eq(i_{in}))$$

The spatial elevation panning coefficients shall also be updated for the 'height' input channels, except CH_T_000 and CH_U_180, for the different elevation degree. Table 110 and Table 111show the initial panning coefficients for the input channels with the elevation of 35 degree. When the elevation of the input channel is higher than 35 degree, the ipsilateral gain to the input channel shall be reduced and the contralateral channel shall be boosted with the gain difference $g_I(elv)$ and $g_C(elv)$.

**Table 110 — Initial spatial localization panning coefficients for 2.8 kHz ~ 10 kHz (35 degree in elevation)**

| Channel Label, $i_{in}$ | $G_{vH0,1\sim6}(i_{in})$ | | | | | |
|---|---|---|---|---|---|---|
| | CH_M_L030 ($G_{vH0,1}(i_{in})$) | CH_M_R030 ($G_{vH0,2}(i_{in})$) | CH_M_000 ($G_{vH0,3}(i_{in})$) | CH_LFE1 ($G_{vH0,4}(i_{in})$) | CH_M_L110 ($G_{vH0,5}(i_{in})$) | CH_M_R110 ($G_{vH0,6}(i_{in})$) |
| CH_U_000 | 0.49146774 | 0.49146774 | 0.34746769 | 0 | 0.44507593 | 0.44507593 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | | | | | |
|---|---|---|---|---|---|---|
| CH_U_L045 | 0.70918131 | 0.27444959 | 0 | 0 | 0.56982642 | 0.31150770 |
| CH_U_R045 | 0.27444959 | 0.70918131 | 0 | 0 | 0.31150770 | 0.56982642 |
| CH_U_L030 | 0.70918131 | 0.27444959 | 0 | 0 | 0.56982642 | 0.31150770 |
| CH_U_R030 | 0.27444959 | 0.70918131 | 0 | 0 | 0.31150770 | 0.56982642 |
| CH_U_L090 | 0.56040317 | 0 | 0 | 0 | 0.81550622 | 0.14456093 |
| CH_U_R090 | 0 | 0.56040317 | 0 | 0 | 0.14456093 | 0.81550622 |
| CH_U_L110 | 0.34278116 | 0 | 0 | 0 | 0.91200900 | 0.22525696 |
| CH_U_R110 | 0 | 0.34278116 | 0 | 0 | 0.22525696 | 0.91200900 |
| CH_U_L135 | 0.34278116 | 0 | 0 | 0 | 0.91200900 | 0.22525696 |
| CH_U_R135 | 0 | 0.34278116 | 0 | 0 | 0.22525696 | 0.91200900 |
| CH_U_180 | 0.22851810 | 0.22851810 | 0 | 0 | 0.66916323 | 0.66916323 |
| CH_T_000 | 0.45328009 | 0.45328009 | 0.33519593 | 0 | 0.48822021 | 0.48822021 |

**Table 111 — Initial spatial localization panning coefficients below 2.8 kHz and above 10 kHz (35 degree in elevation)**

| Channel Label, $i_{in}$ | $G_{vL0,1\sim6}(i_{in})$ | | | | | |
|---|---|---|---|---|---|---|
| | CH_M_L030 ($G_{vL0,1}(i_{in})$) | CH_M_R030 ($G_{vL0,2}(i_{in})$) | CH_M_000 ($G_{vL0,3}(i_{in})$) | CH_LFE1 ($G_{vL0,4}(i_{in})$) | CH_M_L110 ($G_{vL0,5}(i_{in})$) | CH_M_R110 ($G_{vL0,6}(i_{in})$) |
| CH_U_000 | 0.61940062 | 0.61940062 | 0.43791625 | 0 | 0 | 0 |
| CH_U_L045 | 1 | 0 | 0 | 0 | 0 | 0 |
| CH_U_R045 | 0 | 1 | 0 | 0 | 0 | 0 |
| CH_U_L030 | 1 | 0 | 0 | 0 | 0 | 0 |
| CH_U_R030 | 0 | 1 | 0 | 0 | 0 | 0 |
| CH_U_L090 | 0.36730000 | 0 | 0 | 0 | 0.93010002 | 0 |
| CH_U_R090 | 0 | 0.36730000 | 0 | 0 | 0 | 0.93010002 |
| CH_U_L110 | 0 | 0 | 0 | 0 | 1 | 0 |
| CH_U_R110 | 0 | 0 | 0 | 0 | 0 | 1 |
| CH_U_L135 | 0.34278116 | 0 | 0 | 0 | 0.91200900 | 0.22525696 |
| CH_U_R135 | 0 | 0.34278116 | 0 | 0 | 0.22525696 | 0.91200900 |
| CH_U_180 | 0.22851810 | 0.22851810 | 0 | 0 | 0.66916323 | 0.66916323 |
| CH_T_000 | 0.45328009 | 0.45328009 | 0.33519593 | 0 | 0.48822021 | 0.48822021 |

**For** all height input channel $i_{in}$, the $G_{vH,1\sim6}$ and $G_{vL,1\sim6}$ shall be initialized with $G_{vH0,1\sim6}$:

$$G_{vH,1\sim6}(i_{in}) = G_{vH0,1\sim6}(i_{in})$$

$$G_{vL,1\sim6}(i_{in}) = G_{vL0,1\sim6}(i_{in})$$

**For** each height input channel $i_{in}$, the $G_{vH,1\sim6}$ shall be calculated from $G_{vH0,1\sim6}$:

**If** the input channel is CH_U_000,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$G_{vH,5}(i_{in}) = 10^{(0.25 \times \min(\max(elv-35,0),25))/20} \times G_{vH0,5}(i_{in})$$

$$G_{vH,6}(i_{in}) = 10^{(0.25 \times \min(\max(elv-35,0),25))/20} \times G_{vH0,6}(i_{in})$$

**Elseif** the input channel is not CH_U_180

**if** the input channel is side : azimuth is either in (-110, -70) or in (70, 110),

$$g_I(elv) = 10^{(-0.05522 \times \min(\max(elv-35,0),25))/20}$$

$$g_C(elv) = 10^{(0.41879 \times \min(\max(elv-35,0),25))/20}$$

**else** (the input channel is frontal or rear in [-70, 70], [-180, -110], or [110 180] )

$$g_I(elv) = 10^{(-0.047401 \times \min(\max(elv-35,0),25))/20}$$

$$g_C(elv) = 10^{(0.14985 \times \min(\max(elv-35,0),25))/20}$$

**For** each output channels ipsilateral to the input channel,
(e.g. CH_M_L030 and CH_M_L110 for CH_U_L045)

$$G_{vH,I}(i_{in}) = g_I(elv) \times G_{vH0,I}(i_{in})$$

where the $I$ of $G_{vH,I}(i_{in})$ represents the indices ipsilateral to the $i_{in}$

**For** each output channels contralateral to the input channel,
(e.g. CH_M_R030 and CH_M_R110 for CH_U_L045)

$$G_{vH,C}(i_{in}) = g_C(elv) \times G_{vH0,C}(i_{in})$$

where the $C$ of $G_{vH,C}(i_{in})$ represents the indices contralateral to the $i_{in}$

A set of spatial localization panning coefficients for the input channel is normalized to make the sum of powers be 1.

$$P_{G_{vH}}(i_{in}) = \sqrt{\sum_{o=1}^{6} G_{vH,o}^2(i_{in})}$$

$$G_{vH,1\sim6}(i_{in}) = \frac{1}{P_{G_{vH}}} G_{vH,1\sim6}(i_{in})$$

**If** the input channel is in [-160, -110) or (110 160], $G_{vL,C}(i_{in})$ shall be updated :

$$G_{vL,I}(i_{in}) = g_I(elv) \times G_{vL0,I}(i_{in})$$

$$G_{vL,C}(i_{in}) = g_C(elv) \times G_{vL0,C}(i_{in})$$

$$P_{G_{vL}}(i_{in}) = \sqrt{\sum_{o=1}^{6} G_{vL,o}^2(i_{in})}$$

$$G_{vL,1\sim6}(i_{in}) = \frac{1}{P_{G_{vL}}} G_{vL,1\sim6}(i_{in})$$

Note that only the panning coefficients of the $G_{vL,1\sim6}$ for the rear input channels change and those for the frontal/side input channel do not.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

As a result of **initElevSptlParms** following parameters are derived :

$G^{k}_{EQ,eq(i_{in})}$: Updated spatial elevation filter coefficient vector (71band) for the input channel $i_{in}$.

$G_{vH,1\sim6}(i_{in})$ : Updated spatial elevation panning coefficients for the input signal in the range of 2.8~10 kHz

$G_{vL,1\sim6}(i_{in})$ : Updated spatial elevation panning coefficients for the input signal below 2.8 kHz and above 10kHz

#### 11.4.1.6.7.4 renderElevSptlParms : Derivation of input-output channel mapping and equalizer for spatial elevation rendering

**renderElevSptlParms** initializes the input-output channel mapping for spatial elevation rendering for an input channel ($i_{in}$).

Initialize the mapping counter $i$=1;

For $m$=1 to 6

If $G_{vH,m}(i_{in}) > 1$

$s^{P}{}_{i}$ = index of source channel $i_{in}$

$d^{P}{}_{i}$ = index of channel $m$ in output

$g^{P}{}_{iH}$ = (value of Gain column) * $G_{vH,m}(i_{in})$

$g^{P}{}_{iL}$ = (value of Gain column) * $G_{vL,m}(i_{in})$

$e^{P}{}_{i}$ = $eq(i_{in})$

$i = i + 1$

$n^{P} = i - 1$;

return { $\mathbf{s}^{P}$, $\mathbf{d}^{P}$, $\mathbf{g}^{P}{}_{H}$, $\mathbf{g}^{P}{}_{L}$, $\mathbf{e}^{P}$, and $n^{P}$ }

Note that the initialization does not add the input-output channel mapping that the gain $g^{P}{}_{iH}$ is zero and $eq(i_{in})$ is from 7 to 15.

By applying the downmix rules defined in Table 114 with the spatial elevation filter and spatial localization panning coefficients, the spatial elevation rendering parameters for each input channel is summarized as :

- CH_U_L135, CH_U_R135, CH_U_180, or CH_T_000
  - Spatial Elevation Filter : the HRTF-based EQs (EQVB , EQVBC, or EQVOG)
  - Spatial Elevation Panning : Filtered signal is multiplied with the same panning coefficients over all frequency (the $\mathbf{g}^{P}{}_{H}$ is identical to the $\mathbf{g}^{P}{}_{L}$ )
- CH_U_L110, and CH_U_R110
  - Spatial Elevation Filter : the HRTF-based EQ (EQVBA)
  - Spatial Elevation Panning : Filtered EQ signal is multiplied with the different panning coefficients over frequency range
    - $\mathbf{g}^{P}{}_{H}$ for the elevation-effective range (2.8k~10kHz)
    - $\mathbf{g}^{P}{}_{L}$ for the rest frequency range : Use the “add-to-the-closest channel” method in order to provide enough envelopment and keep the audio channel wide enough.
- CH_U_L090 and CH_U_R090
  - Spatial Elevation Filter : the HRTF-based EQs (EQVS)

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

  - Spatial Elevation Panning : Filtered EQ signal is multiplied with the different panning coefficients over frequency range
    - $\mathbf{g}^{P}_{H}$ for the elevation-effective range (2.8k~10kHz)
    - $\mathbf{g}^{P}_{L}$ for the rest frequency range : Panned at 90 degree using front and surround loudspeakers in order to provide enough envelopment and keep the audio channel wide enough.
- CH_U_L030, CH_U_R030, CH_U_L045, CH_U_R045
  - Spatial Elevation Filter : the HRTF-based EQs (EQVF)
  - Spatial Elevation Panning : Filtered EQ signal is multiplied with the different panning coefficients over frequency range
    - $\mathbf{g}^{P}_{H}$ for the elevation-effective range (2.8k~10kHz)
    - $\mathbf{g}^{P}_{L}$ for the rest frequency range : Use the “add-to-the-closest channel” method in order to provide enough envelopment and keep the audio channel wide enough.
  - Signals for the surround speakers, CH_M_L110 and CH_M_R110, are delayed with approximately 3 msec in order to avoid front-back confusion using precedence effect. See details in 11.4.2.3
- CH_U_000
  - Spatial Elevation Filter : : the HRTF-based EQ (EQVFC)
  - Spatial Elevation Panning : Filtered EQ signal is multiplied with the different panning coefficients over frequency range
    - $\mathbf{g}^{P}_{H}$ for the elevation-effective range (2.8k~10kHz)
    - $\mathbf{g}^{P}_{L}$ for the rest frequency range : panned among three frontal output channels, CH_M_L030, CH_M_000, and CH_M_R030.
  - Signals for the surround speakers, CH_M_L110 and CH_M_R110, are delayed with approximately 3 msec in order to avoid front-back confusion using precedence effect. See details in 11.4.2.3

#### 11.4.1.6.7.5 renderElevTmbrParms : Derivation of input-output channel mapping and equalizer for timbral elevation rendering

After the **renderElevSptlParms**, **renderElevTmbrParms** initializes another input-output channel mapping for timbral elevation rendering for the same input channel $i_{in}$. The parameters are initialized following 11.4.1.6.3 but ignoring the downmix rules with a destination field of **VIRTUAL**. As a result, a set of $\mathbf{s}^{S}$, $\mathbf{d}^{S}$, $\mathbf{g}^{S}$, $\mathbf{e}^{S}$, and $\mathbf{gain}^{S}$ are defined for the input channel $i_{in}$ by the process:

Initialize the mapping counter $i=1$;

Search the first entry of the input channel in the Source column of the Table 114 and the channels in the Destination column exist below the entry with VIRTUAL destination.

If **Destination** column contains ALL_U, then:
For each output channel x with “CH_U_” in its name, do:
$s^{S}_{i}$ = index of source channel in input
$d^{S}_{i}$ = index of channel x in output
$g^{S}_{i}$ = (value of Gain column) / sqrt(number of “CH_U_” output channels)
$e^{S}_{i}$ = value of EQ column
$gain_{i}$ = (value of Gain column)
$i = i + 1$
Else if **Destination** column contains ALL_M, then:
For each output channel x with “CH_M_” in its name, do:
$s^{S}_{i}$ = index of source channel in input
$d^{S}_{i}$ = index of channel x in output
$g^{S}_{i}$ = (value of Gain column) / sqrt(number of “CH_M_” output channels)
$e^{S}_{i}$ = value of EQ column
$gain_{i}$ = (value of Gain column)
$i = i + 1$
Else If there is one channel in the **Destination** column, then:
$s^{S}_{i}$ = index of source channel in input
$d^{S}_{i}$ = index of destination channel in output
$g^{S}_{i}$ = value of Gain column

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$e^{S}_i$ = value of EQ column
if $e^{S}_i$ == 13 (EQBTM)
$e^{S}_i$ = 0 (No Process)
$gain_i$ = (value of Gain column)
$i = i + 1$
Else (two channels in **Destination** column)
$s^{S}_i$ = index of source channel in input
$d^{S}_i$ = index of first destination channel in output
$g^{S}_i$ = (value of Gain column) * $g_1$
$e^{S}_i$ = value of EQ column
if $e^{S}_i$ == 13 (EQBTM)
$e^{S}_i$ = 0 (No Process)
$gain_i$ = (value of Gain column)
$i = i + 1$

$s^{S}_i$ = $s^{S}_{i-1}$
$d^{S}_i$ = index of second destination channel in output
$g^{S}_i$ = (value of Gain column) * $g_2$
$e^{S}_i$ = $e^{S}_{i-1}$
$gain_i$ = (value of Gain column)
$i = i + 1$
$n^{S}$ = $i$ - 1;
return { $\mathbf{s}^{S}$, $\mathbf{d}^{S}$, $\mathbf{g}^{S}$, $\mathbf{e}^{S}$, **gain** and $n^{S}$ }

The gains $g_1$ and $g_2$ are computed by applying tangent law amplitude panning in the following way:

— Unwrap source destination channel azimuth angles to be positive.

— The azimuth angles of the destination channels are $\alpha_1$ and $\alpha_2$ (see Table 112).

— The azimuth angle of the source channel (= panning target) is $\alpha_{\text{src}}$.

— $\alpha_0 = \dfrac{|\alpha_1 - \alpha_2|}{2}$

— $\alpha_{\text{center}} = \dfrac{\alpha_1 + \alpha_2}{2}$

— $\alpha = (\alpha_{\text{center}} - \alpha_{\text{src}}) \cdot \text{sgn}(\alpha_2 - \alpha_1)$

— $g_1 = \dfrac{g}{\sqrt{1+g^2}}, \quad g_2 = \dfrac{1}{\sqrt{1+g^2}} \qquad \text{with} \qquad g = \dfrac{\tan\alpha_0 - \tan\alpha + 10^{-10}}{\tan\alpha_0 + \tan\alpha + 10^{-10}}$

#### 11.4.1.6.7.6 Post-Processing for Random Setups with Elevation Rendering

After the parameters of $\mathbf{S}^{P}$, $\mathbf{D}^{P}$, $\mathbf{G}^{P}_{H}$, $\mathbf{G}^{P}_{L}$, and $\mathbf{E}^{P}$ are initialized based on channel information, they are modified according to the azimuth and elevation deviations. For the convenience, $C^{P}_{ii}$ refers the label of $S^{P}_i$, $C^{P}_{io}$ refers

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

the label of $D^{P}_{i}$, $r_{ele,D_i}$ refers the elevation deviation of the $C^{P}_{io}$, and $r_{azi,D_i}$ refers the azimuth deviation of the $C^{P}_{io}$,

(1) Elevation Post-Processing 1 : Find the “practically identical” channel

For each element $i$ in $\mathbf{S}^{P}$, do

$flag(i)=0$

For each element $i$ in $\mathbf{S}^{P}$, do

**If** $r_{ele,D_i}$ > 20 and $r_{azi,D_i}$ ≤ 15
**If** ( $C^{P}_{io}$ == CH_M_L030 and ( $C^{P}_{ii}$ == CH_U_L030 || $C^{P}_{ii}$ == CH_M_L045 ) ) ||
( $C^{P}_{io}$ == CH_M_R030 and ( $C^{P}_{ii}$ == CH_U_R030 || $C^{P}_{ii}$ == CH_M_R045 ) ) ||
( $C^{P}_{io}$ == CH_M_000 and $C^{P}_{ii}$ == CH_U_000 ) ||
{
$G^{P}_{iH}$ = 1
$G^{P}_{iL}$ = 1
$flag(i)$= 1
For each element $j$ in $\mathbf{S}^{P}$, do
If $C^{P}_{ii}$ == $C^{P}_{ji}$ && $i \neq j$
$G^{P}_{jH}$ = 0
$G^{P}_{jL}$ = 0
$flag(j)$= 1
}
**If** $r_{ele,D_i}$ > 20 and $r_{azi,D_i}$ ≤ 25
**If** ( $C^{P}_{io}$ == CH_M_L110 and ( $C^{P}_{ii}$ == CH_U_L110 || $C^{P}_{ii}$ == CH_M_L135 ) ) ||
( $C^{P}_{io}$ == CH_M_R110 and ( $C^{P}_{ii}$ == CH_U_R110 || $C^{P}_{ii}$ == CH_M_R135 ) )
{
$G^{P}_{iH}$ = 1
$G^{P}_{iL}$ = 1
$flag(i)$= 1
For each element $j$ in $\mathbf{S}^{P}$, do
If $C^{P}_{ii}$ == $C^{P}_{ji}$ && $i \neq j$
$G^{P}_{jH}$ = 0
$G^{P}_{jL}$ = 0
$flag(j)$= 1
}
For each element $i$ in $\mathbf{S}^{P}$, do
**If** $r_{ele,D_i}$ > 20 and $C^{P}_{io}$ == CH_M_L110 and $C^{P}_{ii}$ == CH_U_L090
For each element $j$ in $\mathbf{S}^{P}$, do
**If** $r_{ele,D_j}$ > 20 and $C^{P}_{jo}$ == CH_M_L030 and $C^{P}_{ii}$ == CH_U_L090
For each element $k$ in $\mathbf{S}^{P}$, do
**If** $C^{P}_{ki}$ == CH_U_L090
$G^{P}_{kH}$ = 0
$G^{P}_{kL}$ = 0
$flag(k)$= 1
$G^{P}_{iH}$ = $g_1$
$G^{P}_{iL}$ = $g_1$
$G^{P}_{jH}$ = $g_2$
$G^{P}_{jL}$ = $g_2$
$flag(i)$= 1
$flag(j)$= 1
**If** $r_{ele,D_i}$ > 20 and $C^{P}_{io}$ == CH_M_R110 and $C^{P}_{ii}$ == CH_U_R090
For each element $j$ in $\mathbf{S}^{P}$, do

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**If** $r_{ele,D_j}$ > 20 and $C^{P}_{jo}$ == CH_M_R030 and $C^{P}_{ii}$ == CH_U_R090

For each element $k$ in $\mathbf{S}^{P}$, do

**If** $C^{P}_{ki}$ == CH_U_R090

$G^{P}_{kH}$ = 0

$G^{P}_{kL}$ = 0

$flag(k)$= 1

$G^{P}_{iH}$ = $g_1$

$G^{P}_{iL}$ = $g_1$

$G^{P}_{jH}$ = $g_2$

$G^{P}_{jL}$ = $g_2$

$flag(i)$= 1

$flag(j)$= 1

The gains $g_1$ and $g_2$ are computed by applying tangent law amplitude panning in the following way:

— Unwrap source destination channel azimuth angles to be positive.

— The azimuth with the deviation for $C^{P}_{io}$ and $C^{P}_{jo}$ are $\alpha_1$ and $\alpha_2$

— The azimuth angle of the source channel ( = panning target) is $\alpha_{\text{src}}$.

— $\alpha_0 = \dfrac{|\alpha_1 - \alpha_2|}{2}$

— $\alpha_{\text{center}} = \dfrac{\alpha_1 + \alpha_2}{2}$

— $\alpha = (\alpha_{\text{center}} - \alpha_{\text{src}}) \cdot \text{sgn}(\alpha_2 - \alpha_1)$

— $g_1 = \dfrac{g}{\sqrt{1+g^2}}, \quad g_2 = \dfrac{1}{\sqrt{1+g^2}} \qquad \text{with} \qquad g = \dfrac{\tan\alpha_0 - \tan\alpha + 10^{-10}}{\tan\alpha_0 + \tan\alpha + 10^{-10}}$

(2) Elevation Post-Processing 2 on panning coefficients : Find the “practically dual mono” channel

For each element $i$ in $\mathbf{S}^{P}$, if $flag(i)$==0

**If** both CH_M_L030 and CH_M_R030 have elevation deviations more than 20 degree

**if** $C^{P}_{ii}$ == CH_U_000

**if** $C^{P}_{io}$ == CH_M_L030 || CH_M_R030

$G^{P}_{iH}$ = 1

$G^{P}_{iL}$ = 1

$flag(i)$=1

**else**

$G^{P}_{iH}$ = 0

$G^{P}_{iL}$ = 0

$flag(i)$=1

**If** both CH_M_L110 and CH_M_R110 are elevated

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015



© ISO/IEC 2015 – All rights reserved

**if** $C^{P}_{ii}$ == CH_U_180
**if** $C^{P}_{io}$ == CH_M_L110 || CH_M_R110
$G^{P}_{iH}$ = 1
$G^{P}_{iL}$ = 1
$flag(i)$=1
**else**
$G^{P}_{iH}$ = 0
$G^{P}_{iL}$ = 0
$flag(i)$=1

(3) Elevation Post-Processing 3 on panning coefficients : Keep the central image

For each element $i$ in $\mathbf{S}^{P}$, if $flag(i)$==0

**if** $C^{P}_{ii}$ == CH_U_000 || CH_T_000 || CH_U_180
**if** only one of the output channels of CH_M_L030 or CH_M_R030 has an elevation deviation more than 20 degree, **then**
**if** $C^{P}_{ii}$ == CH_U_000 || CH_T_000
**if** $C^{P}_{io}$ == CH_M_L030 || CH_M_R030
**if** $C^{P}_{io}$ is elevated
$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{3}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
**if** $C^{P}_{ii}$ == CH_U_000
$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
**if** $C^{P}_{ii}$ == CH_T_000
$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{3}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$flag(i)$=1
**else**
$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$flag(i)$=1
**if** $C^{P}_{io}$ == CH_M_000
$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$flag(i)$=1
**if** $C^{P}_{ii}$ == CH_U_180
**if** $C^{P}_{io}$ == CH_M_L030 || CH_M_R030
**if** $C^{P}_{io}$ is not elevated
$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$flag(i)$=1
**if** only one of the output channels of CH_M_L110 or CH_M_R110 has an elevation deviation more than 20 degree, **then**
**if** $C^{P}_{ii}$ == CH_U_180 || CH_T_000
**if** $C^{P}_{io}$ == CH_M_L110 || CH_M_R110
**if** $C^{P}_{io}$ is elevated
$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{2}{20}\times\frac{r_{el}\varepsilon_{D_i}}{35}}$
$flag(i)$=1
**if** $C^{P}_{ii}$ == CH_U_000
**if** $C^{P}_{io}$ == CH_M_L110 || CH_M_R110
**if** $C^{P}_{io}$ is not elevated

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{2}{20}\times\frac{r_{gle,D_i}}{35}}$$
$$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{2}{20}\times\frac{r_{gle,D_i}}{35}}$$
$$flag(i)=1$$

(4) Elevation Post-Processing 4 on panning coefficients : Keep the L/R balance when the Contralateral Frontal Channel Elevated

For each element $i$ in $\mathbf{S}^{P}$, if $flag(i)==0$
**if** CH_M_L030 is elevated more than 20 degree and CH_M_R030 is not
**if** $C^{P}_{ii}$ == CH_U_R030 || $C^{P}_{ii}$ == CH_U_R045, do
**if** $C^{P}_{io}$ == CH_M_L030
$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{1}{20}\times\frac{r_{gle,D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{1}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** $C^{P}_{io}$ == CH_M_L110
$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{4}{20}\times\frac{r_{gle,D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{4}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** $C^{P}_{ii}$ == CH_U_L090, do
**if** $C^{P}_{io}$ == CH_M_L030
$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{8.7}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** $C^{P}_{io}$ == CH_M_L110
$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{4}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** $C^{P}_{io}$ == CH_M_R110
$G^{P}_{iH} = 0$
$flag(i)=1$
**elseif** $C^{P}_{ii}$ == CH_U_L110 || $C^{P}_{ii}$ == CH_U_L135, do
**if** $C^{P}_{io}$ == CH_M_L030
$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{5.6}{20}\times\frac{r_{gle,D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{5.6}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** $C^{P}_{io}$ == CH_M_L110
$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{4.6}{20}\times\frac{r_{gle,D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{4.6}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** $C^{P}_{io}$ == CH_M_R110
$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{1}{20}\times\frac{r_{gle,D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{1}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** CH_M_R030 is elevated more than 20 degree and CH_M_L030 is not
**if** $C^{P}_{ii}$ == CH_U_L030 || $C^{P}_{ii}$ == CH_U_L045, do
**if** $C^{P}_{io}$ == CH_M_R030
$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{1}{20}\times\frac{r_{gle,D_i}}{35}}$
$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{1}{20}\times\frac{r_{gle,D_i}}{35}}$
$flag(i)=1$
**elseif** $C^{P}_{io}$ == CH_M_R110

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{4}{20}\times\frac{r_{azi,D_i}}{35}}$

$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{4}{20}\times\frac{r_{azi,D_i}}{35}}$

$flag(i)=1$

**elseif** $C^{P}_{ii}$ == CH_U_R090, do

**if** $C^{P}_{io}$ == CH_M_R030

$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{8.7}{20}\times\frac{r_{azi,D_i}}{35}}$

$flag(i)=1$

**elseif** $C^{P}_{io}$ == CH_M_R110

$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{4}{20}\times\frac{r_{azi,D_i}}{35}}$

$flag(i)=1$

**elseif** $C^{P}_{io}$ == CH_M_R110

$G^{P}_{iH} = 0$

$flag(i)=1$

**elseif** $C^{P}_{ii}$ == CH_U_R110 || $C^{P}_{ii}$ == CH_U_R135, do

**if** $C^{P}_{io}$ == CH_M_R030

$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{5.6}{20}\times\frac{r_{azi,D_i}}{35}}$

$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{5.6}{20}\times\frac{r_{azi,D_i}}{35}}$

$flag(i)=1$

**elseif** $C^{P}_{io}$ == CH_M_R110

$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{4.6}{20}\times\frac{r_{azi,D_i}}{35}}$

$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{4.6}{20}\times\frac{r_{azi,D_i}}{35}}$

$flag(i)=1$

**elseif** $C^{P}_{io}$ == CH_M_L110

$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{1}{20}\times\frac{r_{azi,D_i}}{35}}$

$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{1}{20}\times\frac{r_{azi,D_i}}{35}}$

$flag(i)=1$

(5) Azimuth Post-Processing on panning coefficients

$fbias = r_{azi,A}$(CH_M_L030)+ $r_{azi,A}$(CH_M_R030)

$bbias = r_{azi,A}$(CH_M_L110)+ $r_{azi,A}$(CH_M_R110)

**For** each element $i$ in $S^{P}$,

**If** $fbias > 10$

**If** ( $C^{P}_{ii}$ == CH_U_000 || $C^{P}_{ii}$ == CH_T_000 ) && ( $C^{P}_{io}$ == CH_M_L030 )

$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{1}{20}\times\frac{(fbias)}{15}}$

**If** $C^{P}_{ii}$ == CH_U_000

$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{2}{20}\times\frac{(fbias)}{15}}$

**elseif** $C^{P}_{ii}$ == CH_T_000

$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{1}{20}\times\frac{(fbias)}{15}}$

$flag(i)=1$

**If** $fbias < -10$

**If** ( $C^{P}_{ii}$ == CH_U_000 || $C^{P}_{ii}$ == CH_T_000 ) && ( $C^{P}_{io}$ == CH_M_R030 )

$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{1}{20}\times\frac{(fbias)}{15}}$

**If** $C^{P}_{ii}$ == CH_U_000

$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{2}{20}\times\frac{(fbias)}{15}}$

**elseif** $C^{P}_{ii}$ == CH_T_000

$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{1}{20}\times\frac{(fbias)}{15}}$

$flag(i)=1$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**If** $bbias > 10$

**If** ( $C^{P}_{ii}$ == CH_U_180 || $C^{P}_{ii}$ == CH_T_000 ) && ( $C^{P}_{io}$ == CH_M_L110 )

$G^{P}_{iH} = G^{P}_{iH} \times 10^{\frac{1}{20}\times\frac{(bbias)}{15}}$

$G^{P}_{iL} = G^{P}_{iL} \times 10^{\frac{1}{20}\times\frac{(bbias)}{15}}$

$flag(i)=1$

**If** $bbias < -10$

**If** ( $C^{P}_{ii}$ == CH_U_180 || $C^{P}_{ii}$ == CH_T_000 ) && ( $C^{P}_{io}$ == CH_M_R110 )

$G^{P}_{iH} = G^{P}_{iH} \times 10^{-\frac{1}{20}\times\frac{(bbias)}{15}}$

$G^{P}_{iL} = G^{P}_{iL} \times 10^{-\frac{1}{20}\times\frac{(bbias)}{15}}$

$flag(i)=1$

(6) Spatial elevation coefficient normalization

**For** each element $i$ in $\mathbf{S}^{P}$, if $flag(i)==1$

$P_{iL} = 0$

$P_{iH} = 0$

**For** each element $i$ in $\mathbf{S}^{P}$, if $flag(i)==1$

**For** each element $k$ in $\mathbf{S}^{P}$

**If** $C^{P}_{ii}$ == $C^{P}_{ki}$

$P_{iL} = P_{iL} + (G^{P}_{iL})^2$

$P_{iH} = P_{iH} + (G^{P}_{iH})^2$

$flag(i)==0$

**For** each element $k$ in $\mathbf{S}^{P}$

**If** $C^{P}_{ii}$ == $C^{P}_{ki}$

$G^{P}_{iL} = G^{P}_{iL} \times \sqrt{P_{iL}}$

$G^{P}_{iH} = G^{P}_{iH} \times \sqrt{P_{iH}}$

$flag(i)==0$

(7) Elevation Post-Processing on spatial elevation filters

For each element $i$ in $\mathbf{S}^{P}$

**If** $C^{P}_{io}$ is elevated ($r_{ele,A}(C^{P}_{io})$ > 20)

$E^{P}_{i}$ = 0 (no EQ)

(8) Update the $\mathbf{E}^{S}$, $\mathbf{G}_{EQ}$ by same process defined in 11.4.1.6.5 as below :

For each element $i$ in $D^{S}$, do:

**if** output channel with index $D^{S}_{i}$ is a horizontal channel by definition (i.e. output channel label contains the label '_M_'), **and**

**if** this output channel is now a height channel (elevation in range 0..60 degrees), **and**

**if** input channel with index $S^{S}_{i}$ is a height channel (i.e. label contains '_U_'), **then**

—$h$ = min(elevation of randomized output channel, 35) / 35

—$G_{comp} = h \cdot \frac{1}{Gain^{S}_{i}} + (1-h)$

—Apply compensation gain to DMX gain: $G^{S}_{i} = G^{S}_{i} \times G_{comp}$

— Define new equalizer $\mathbf{G}_{EQ,e}$ with the index $e$, where $G^{k}_{EQ,e} = h + (1-h) \cdot G^{k}_{EQ,E_i}$

—$E^{S}_{i} = e$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015



© ISO/IEC 2015 – All rights reserved

— $e = e + 1$

**else if** input channel with index $S^S_i$ is a horizontal channel (label contains '_M_')

— $h$ = min(elevation of randomized output channel, 35) / 35
— Define new equalizer $\mathbf{G}_{\mathrm{EQ},e}$ with the index $e$,
where $G^k_{\mathrm{EQ},e} = h \cdot G^k_{\mathrm{EQ},5} + (1-h) \cdot G^k_{\mathrm{EQ},\mathrm{E}_i}$
— $E^S_i = e$
— $e = e + 1$

#### 11.4.1.6.7.7 Merge general downmix rules and elevation rules

After the elevation rendering parameters are initialized, the vectors of $\mathbf{S}^{\mathrm{P}}$, $\mathbf{D}^{\mathrm{P}}$, $\mathbf{G}^{\mathrm{P}}{}_{\mathrm{H}}$, $\mathbf{G}^{\mathrm{P}}{}_{\mathrm{L}}$, and $\mathbf{E}^{\mathrm{P}}$ that covers elevation rendered height input channel shall be merged with $\mathbf{S}$, $\mathbf{D}$, $\mathbf{G}$, $\mathbf{E}$, and $\mathbf{G}_{\mathrm{EQ}}$ that covers the rest input channels by

$$\mathbf{S}^{\mathrm{P}} = \begin{bmatrix} \mathbf{S}^{\mathrm{P}} \\ \mathbf{S} \end{bmatrix}, \quad \mathbf{D}^{\mathrm{P}} = \begin{bmatrix} \mathbf{D}^{\mathrm{P}} \\ \mathbf{D} \end{bmatrix}, \quad \mathbf{G}^{\mathrm{P}}{}_{\mathrm{H}} = \begin{bmatrix} \mathbf{G}^{\mathrm{P}}{}_{\mathrm{H}} \\ \mathbf{G} \end{bmatrix}, \quad \mathbf{G}^{\mathrm{P}}{}_{\mathrm{L}} = \begin{bmatrix} \mathbf{G}^{\mathrm{P}}{}_{\mathrm{L}} \\ \mathbf{G} \end{bmatrix}, \quad \mathbf{E}^{\mathrm{P}} = \begin{bmatrix} \mathbf{E}^{\mathrm{P}} \\ \mathbf{E} \end{bmatrix}$$

$\mathbf{S}^{\mathrm{S}}$, $\mathbf{D}^{\mathrm{S}}$, $\mathbf{G}^{\mathrm{S}}$, and $\mathbf{E}^{\mathrm{S}}$ are also merged with $\mathbf{S}$, $\mathbf{D}$, $\mathbf{G}$, and $\mathbf{E}$ by

$$\mathbf{S}^{\mathrm{S}} = \begin{bmatrix} \mathbf{S}^{\mathrm{S}} \\ \mathbf{S} \end{bmatrix}, \quad \mathbf{D}^{\mathrm{S}} = \begin{bmatrix} \mathbf{D}^{\mathrm{S}} \\ \mathbf{D} \end{bmatrix}, \quad \mathbf{G}^{\mathrm{S}} = \begin{bmatrix} \mathbf{G}^{\mathrm{S}} \\ \mathbf{G} \end{bmatrix}, \quad \mathbf{E}^{\mathrm{S}} = \begin{bmatrix} \mathbf{E}^{\mathrm{S}} \\ \mathbf{E} \end{bmatrix},$$

When no input channel is rendered by elevation rendering, the vectors of $\mathbf{S}^{\mathrm{P}}$, $\mathbf{D}^{\mathrm{P}}$, $\mathbf{G}^{\mathrm{P}}{}_{\mathrm{H}}$ (or $\mathbf{G}^{\mathrm{P}}{}_{\mathrm{L}}$), and $\mathbf{E}^{\mathrm{P}}$ are identical with $\mathbf{S}^{\mathrm{S}}$, $\mathbf{D}^{\mathrm{S}}$, $\mathbf{G}^{\mathrm{S}}$, and $\mathbf{E}^{\mathrm{S}}$ and also with $\mathbf{S}$, $\mathbf{D}$, $\mathbf{G}$, and $\mathbf{E}$.

11.4.1.6.8 **Derivation of rules-based initialization downmix matrix:**

$\mathbf{M}_{\mathrm{DMX}}$ and $\mathbf{M}_{\mathrm{DMX2}}$ are derived by rearranging the temporary parameters from the mapping-oriented representation (enumerated by mapping counter $i$) to a channel-oriented representation as defined in the following:

Initialize $\mathbf{M}_{\mathrm{DMX}}$ and $\mathbf{M}_{\mathrm{DMX2}}$ as $N_{\mathrm{out}}$ x $N_{\mathrm{in}}$ zero matrixes for all processing bands $k$.

For each $i$ in $\mathbf{S}^{\mathrm{P}}$do:

If ($E^{\mathrm{P}}{}_i$ = 0)

$M^k_{DMX,A,B} = G^{\mathrm{P}}{}_{iL}$ for $A = D^{\mathrm{P}}{}_i$, $B = S^{\mathrm{P}}{}_i$, for $f_k \times f_s / 2 < 2800$

$M^k_{DMX,A,B} = G^{\mathrm{P}}{}_{iH}$ for $A = D^{\mathrm{P}}{}_i$, $B = S^{\mathrm{P}}{}_i$, for $2800 \le f_k \times f_s / 2 \le 10000$

$M^k_{DMX,A,B} = G^{\mathrm{P}}{}_{iL}$ for $A = D^{\mathrm{P}}{}_i$, $B = S^{\mathrm{P}}{}_i$, for $f_k \times f_s / 2 > 10000$

Else

$M^k_{DMX,A,B} = G^{\mathrm{P}}{}_{iL} \times G^k{}_{EQ,E^{\mathrm{P}}_i}$ for $A = D^{\mathrm{P}}{}_i$, $B = S^{\mathrm{P}}{}_i$, $f_k \times f_s / 2 < 2800$

$M^k_{DMX,A,B} = G^{\mathrm{P}}{}_{iH} \times G^k{}_{EQ,E^{\mathrm{P}}_i}$ for $A = D^{\mathrm{P}}{}_i$, $B = S^{\mathrm{P}}{}_i$, $2800 \le f_k \times f_s / 2 \le 10000$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$M^{k}_{DMX,A,B} = G^{P}_{iL} \times G^{k}_{EQ,E^{F}_{i}} \text{ for } A = D^{P}{}_{i},\ B = S^{P}{}_{i},\ f_k \times f_s / 2 > 10000$$

For each *i* in $\mathbf{S}^{S}$do*:*

If ($E^{S}{}_{i}$= 0)

$$M^{k}_{DMX2,A,B} = G^{S}{}_{i} \text{ for } A = D^{S}{}_{i},\ B = S^{S}{}_{i},\qquad \text{for } 1 \le k \le K$$

Else

$$M^{k}_{DMX2,A,B} = G^{S}{}_{i} \times G^{k}_{EQ,E^{S}_{i}} \text{ for } A = D^{S}{}_{i},\ B = S^{S}{}_{i},\ 1 \le k \le K$$

where $f_k$ is the normalized center frequency of frequency band *k*, specified in Table 115, $f_s$ is the sampling frequency, $M^{k}_{DMX,A,B}$ and $M^{k}_{DMX2,A,B}$ denotes the matrix element in the $A$ th row and $B$ th column of $\mathbf{M}_{DMX}$ and $\mathbf{M}_{DMX2}$. Note that after the rules-based initialization this matrix of downmix coefficients will contain columns of zeros, if unknown channels are present in the input format. Those columns are filled with downmix gains as specified in 11.4.1.6.9.

11.4.1.6.9 **VBAP-based downmix coefficients derivation**

Handling of unknown output formats:

In case the output format is considered unknown, the downmix coefficients for all input channels shall be derived as follows:

Each channel of the input setup is regarded as a static audio object at the position defined by the azimuth and elevation angles associated with the input channel. For each input channel the mixing gains to all output loudspeakers are calculated as VBAP panning gains $\mathbf{g}_{\mathbf{scaled}}$ according to 8.4.3, where the same output format shall be signaled to the VBAP algorithm as to the format converter. The panning gain vectors $\mathbf{g}_{\mathbf{scaled}}$ shall be post-processed according to 11.4.1.6.10.

The downmix matrix $\mathbf{M}^{k}_{\mathrm{DMX}}$ is finally derived by filling each matrix column with the post-processed panning gain vector elements of the corresponding input channel, independently of the processing band index k.

Handling of unknown input channels:

In case the input format contains unknown input channels, the downmix coefficients for these channels shall be derived as follows:

Each unknown channel of the input setup is regarded as a static audio object at the position defined by the azimuth and elevation angles associated with the input channel. For each unknown input channel the mixing gains to all output loudspeakers are calculated as VBAP panning gains $\mathbf{g}_{\mathbf{scaled}}$ according to 8.4.3, where the same output format shall be signaled to the VBAP algorithm as to the format converter. The panning gain vectors $\mathbf{g}_{\mathbf{scaled}}$ shall be post-processed according to 11.4.1.6.10.

The downmix matrix $\mathbf{M}^{k}_{\mathrm{DMX}}$ is finally derived by filling each matrix column corresponding to an unknown input channel with the post-processed panning gain vector elements of the corresponding unknown input channel, independently of the processing band index k.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

11.4.1.6.10 **VBAP gains post-processing**

The mixing gains obtained from the VBAP rendering algorithm shall be post-processed to avoid excessive use of phantom sources. Therefore, small matrix gains are set to zero, followed by a renormalization of the panning gains to ensure energy-preservation:

For each panning gain vector $\mathbf{g}_{\mathbf{scaled}}$ do:

If the vector contains at least one panning gain that exceeds the threshold value 0.3, then

Set all vector elements smaller or equal to 0.3 to the value 0.0

Normalize the gain vector such that the sum of squares of the vector elements remains the same as before the post-processing

### 11.4.1.7 Format converter initialization tables

Table 112 lists channel labels, corresponding azimuth and elevation angles, and associated sectors. The sectors are defined as points on the unit sphere, whose azimuth/elevation angles are within or on the borders of the intervals given by the azimuth/elevation start and end values in the table, connecting azimuth start and end values in counter-clockwise direction and connecting elevation start and end values in direction of increasing elevation angles.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 112 — Channels definitions: Channel labels, corresponding azimuth and elevation angles, and associated sectors**

| LoudspeakerGeometry as defined in ISO/IEC 23001-8) | Channel | Azimuth [deg] | Elevation [deg] | Azimuth start angle of sector [deg] | Azimuth end angle of sector [deg] | Elevation start angle of sector [deg] | Elevation end angle of sector [deg] | Ch. is LFE | Position is relative |
|---|---|---|---|---|---|---|---|---|---|
| | CH_EMPTY | n/a | n/a | n/a | n/a | n/a | n/a | 0 | 0 |
| 0 | CH_M_L030 | +30 | 0 | +23 | +37 | -9 | +20 | 0 | 0 |
| 1 | CH_M_R030 | -30 | 0 | -37 | -23 | -9 | +20 | 0 | 0 |
| 2 | CH_M_000 | 0 | 0 | -7 | +7 | -9 | +20 | 0 | 0 |
| 3 | CH_LFE1 | 0 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 4 | CH_M_L110 | +110 | 0 | +101 | +124 | -45 | +20 | 0 | 0 |
| 5 | CH_M_R110 | -110 | 0 | -124 | -101 | -45 | +20 | 0 | 0 |
| 6 | CH_M_L022 | +22 | 0 | +8 | +22 | -9 | +20 | 0 | 0 |
| 7 | CH_M_R022 | -22 | 0 | -22 | -8 | -9 | +20 | 0 | 0 |
| 8 | CH_M_L135 | +135 | 0 | 125 | 142 | -45 | +20 | 0 | 0 |
| 9 | CH_M_R135 | -135 | 0 | -142 | -125 | -45 | +20 | 0 | 0 |
| 10 | CH_M_180 | 180 | 0 | 158 | -158 | -45 | +20 | 0 | 0 |
| 13 | CH_M_L090 | +90 | 0 | +76 | +100 | -45 | +20 | 0 | 0 |
| 14 | CH_M_R090 | -90 | 0 | -100 | -76 | -45 | +20 | 0 | 0 |
| 15 | CH_M_L060 | +60 | 0 | +53 | +75 | -9 | +20 | 0 | 0 |
| 16 | CH_M_R060 | -60 | 0 | -75 | -53 | -9 | +20 | 0 | 0 |
| 17 | CH_U_L030 | +30 | +35 | +11 | +37 | +21 | +60 | 0 | 0 |
| 18 | CH_U_R030 | -30 | +35 | -37 | -11 | +21 | +60 | 0 | 0 |
| 19 | CH_U_000 | 0 | +35 | -10 | +10 | +21 | +60 | 0 | 0 |
| 20 | CH_U_L135 | +135 | +35 | +125 | +157 | +21 | +60 | 0 | 0 |
| 21 | CH_U_R135 | -135 | +35 | -157 | -125 | +21 | +60 | 0 | 0 |
| 22 | CH_U_180 | 180 | +35 | +158 | -158 | +21 | +60 | 0 | 0 |
| 23 | CH_U_L090 | +90 | +35 | +67 | +100 | +21 | +60 | 0 | 0 |
| 24 | CH_U_R090 | -90 | +35 | -100 | -67 | +21 | +60 | 0 | 0 |
| 25 | CH_T_000 | 0 | +90 | -180 | +180 | +61 | +90 | 0 | 0 |
| 26 | CH_LFE2 | +45 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 27 | CH_L_L045 | +45 | -15 | +11 | +75 | -45 | -10 | 0 | 0 |
| 28 | CH_L_R045 | -45 | -15 | -75 | -11 | -45 | -10 | 0 | 0 |
| 29 | CH_L_000 | 0 | -15 | -10 | +10 | -45 | -10 | 0 | 0 |
| 30 | CH_U_L110 | +110 | +35 | +101 | +124 | +21 | +60 | 0 | 0 |
| 31 | CH_U_R110 | -110 | +35 | -124 | -101 | +21 | +60 | 0 | 0 |
| 32 | CH_U_L045 | +45 | +35 | +38 | +66 | +21 | +60 | 0 | 0 |
| 33 | CH_U_R045 | -45 | +35 | -66 | -38 | +21 | +60 | 0 | 0 |
| 34 | CH_M_L045 | +45 | 0 | +38 | +52 | -9 | +20 | 0 | 0 |
| 35 | CH_M_R045 | -45 | 0 | -52 | -38 | -9 | +20 | 0 | 0 |
| 36 | CH_LFE3 | -45 | n/a | n/a | n/a | n/a | n/a | 1 | 0 |
| 37 | CH_M_LSCR | +60 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 38 | CH_M_RSCR | -60 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 39 | CH_M_LSCH | +30 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 40 | CH_M_RSCH | -30 | 0 | n/a | n/a | n/a | n/a | 0 | 1 |
| 41 | CH_M_L150 | +150 | 0 | 143 | 157 | -45 | +20 | 0 | 0 |
| 42 | CH_M_R150 | -150 | 0 | -157 | -143 | -45 | +20 | 0 | 0 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 113 — Formats with corresponding number of channels and channel ordering**

| Loudspeaker Layout Index or ChannelConfiguration as defined in ISO/IEC 23001-8 | Number of Channels | Channels (with ordering) |
|---|---|---|
| 1 | 1 | CH_M_000 |
| 2 | 2 | CH_M_L030, CH_M_R030 |
| 3 | 3 | CH_M_L030, CH_M_R030, CH_M_000 |
| 4 | 4 | CH_M_L030, CH_M_R030, CH_M_000, CH_M180 |
| 5 | 5 | CH_M_L030, CH_M_R030, CH_M_000, CH_M_L110, CH_M_R110 |
| 6 | 6 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110 |
| 7 | 8 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_L060, CH_M_R060 |
| 8 | | n.a. |
| 9 | 3 | CH_M_L030, CH_M_R030, CH_M_180 |
| 10 | 4 | CH_M_L030, CH_M_R030, CH_M_L110, CH_M_R110 |
| 11 | 7 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_180 |
| 12 | 8 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_L135, CH_M_R135 |
| 13 | 24 | CH_M_L060, CH_M_R060, CH_M_000, CH_LFE1, CH_M_L135, CH_M_R135, CH_M_L030, CH_M_R030, CH_M_180, CH_LFE2, CH_M_L090, CH_M_R090, CH_U_L045, CH_U_R045, CH_U_000, CH_T_000, CH_U_L135, CH_U_R135, CH_U_L090, CH_U_R090, CH_U_180, CH_L_000, CH_L_L045, CH_L_R045 |
| 14 | 8 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_U_L030, CH_U_R030 |
| 15 | 12 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L135, CH_M_R135, CH_LFE2, CH_M_L090, CH_M_R090, CH_U_L045, CH_U_R045, CH_U_180 |
| 16 | 10 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_U_L030, CH_U_R030, CH_U_L110, CH_U_R110 |
| 17 | 12 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_U_L030, CH_U_R030, CH_U_000, CH_U_L110, CH_U_R110, CH_T_000 |
| 18 | 14 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L110, CH_M_R110, CH_M_L150, CH_M_R150, CH_U_L030, CH_U_R030, CH_U_000, CH_U_L110, CH_U_R110, CH_T_000 |
| 19 | 12 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L135, CH_M_R135, CH_M_L090, CH_M_R090, CH_U_L030, CH_U_R030, CH_U_L135, CH_U_R135 |
| 20 | 14 | CH_M_L030, CH_M_R030, CH_M_000, CH_LFE1, CH_M_L135, CH_M_R135, CH_M_L090, CH_M_R090, CH_U_L045, CH_U_R045, CH_U_L135, CH_U_R135, CH_M_LSCR, CH_M_RSCR |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 114 — Converter rules matrix**

| Source | Destination | Gain | EQ index |
|---|---|---|---|
| CH_M_000 | CH_M_L022, CH_M_R022 | 1.0 | 0 (off) |
| CH_M_000 | CH_M_L030, CH_M_R030 | 1.0 | 0 (off) |
| CH_M_L022 | CH_M_000, CH_M_L030 | 1.0 | 0 (off) |
| CH_M_L022 | CH_M_L030 | 1.0 | 0 (off) |
| CH_M_R022 | CH_M_000, CH_M_R030 | 1.0 | 0 (off) |
| CH_M_R022 | CH_M_R030 | 1.0 | 0 (off) |
| CH_M_L045 | CH_M_L030, CH_M_L060 | 1.0 | 0 (off) |
| CH_M_L045 | CH_M_L030 | 1.0 | 0 (off) |
| CH_M_R045 | CH_M_R030, CH_M_R060 | 1.0 | 0 (off) |
| CH_M_R045 | CH_M_R030 | 1.0 | 0 (off) |
| CH_M_L060 | CH_M_L045, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L060 | CH_M_L030, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L060 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R060 | CH_M_R045, CH_M_R110, | 1.0 | 0 (off) |
| CH_M_R060 | CH_M_R030, CH_M_R110, | 1.0 | 0 (off) |
| CH_M_R060 | CH_M_R030, | 0.8 | 0 (off) |
| CH_M_L090 | CH_M_L045, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L090 | CH_M_L030, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L090 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R090 | CH_M_R045, CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R090 | CH_M_R030, CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R090 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_L110 | CH_M_L135 | 1.0 | 0 (off) |
| CH_M_L110 | CH_M_L090 | 0.8 | 0 (off) |
| CH_M_L110 | CH_M_L045 | 0.8 | 0 (off) |
| CH_M_L110 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R110 | CH_M_R135 | 1.0 | 0 (off) |
| CH_M_R110 | CH_M_R090 | 0.8 | 0 (off) |
| CH_M_R110 | CH_M_R045 | 0.8 | 0 (off) |
| CH_M_R110 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_L135 | CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L135 | CH_M_L150 | 1.0 | 0 (off) |
| CH_M_L135 | CH_M_L090 | 0.8 | 0 (off) |
| CH_M_L135 | CH_M_L045 | 0.8 | 0 (off) |
| CH_M_L135 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R135 | CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R135 | CH_M_R150 | 1.0 | 0 (off) |
| CH_M_R135 | CH_M_R090 | 0.8 | 0 (off) |
| CH_M_R135 | CH_M_R045 | 0.8 | 0 (off) |
| CH_M_R135 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_L150 | CH_M_L135 | 1.0 | 0 (off) |
| CH_M_L150 | CH_M_L110 | 1.0 | 0 (off) |
| CH_M_L150 | CH_M_L045 | 0.8 | 0 (off) |
| CH_M_L150 | CH_M_L030 | 0.8 | 0 (off) |
| CH_M_R150 | CH_M_R135 | 1.0 | 0 (off) |
| CH_M_R150 | CH_M_R110 | 1.0 | 0 (off) |
| CH_M_R150 | CH_M_R045 | 0.8 | 0 (off) |
| CH_M_R150 | CH_M_R030 | 0.8 | 0 (off) |
| CH_M_180 | CH_M_R150, CH_M_L150 | 1.0 | 0 (off) |
| CH_M_180 | CH_M_R135, CH_M_L135 | 1.0 | 0 (off) |
| CH_M_180 | CH_M_R110, CH_M_L110 | 1.0 | 0 (off) |
| CH_M_180 | CH_M_R090, CH_M_L090 | 0.8 | 0 (off) |
| CH_M_180 | CH_M_R045, CH_M_L045 | 0.6 | 0 (off) |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | | |
|---|---|---|---|
| CH_M_180 | CH_M_R030, CH_M_L030 | 0.6 | 0 (off) |
| CH_U_000 | CH_U_L030, CH_U_R030 | 1.0 | 0 (off) |
| CH_U_000 | VIRTUAL | 1.0 | 9 (EQVFC) |
| CH_U_000 | CH_M_L030, CH_M_R030 | 0.85 | 0 (off) |
| CH_U_L045 | CH_U_L030 | 1.0 | 0 (off) |
| CH_U_L045 | VIRTUAL | 1.0 | 7 (EQVF) |
| CH_U_L045 | CH_M_L045 | 0.85 | 1 |
| CH_U_L045 | CH_M_L030 | 0.85 | 1 |
| CH_U_R045 | CH_U_R030 | 1.0 | 0 (off) |
| CH_U_R045 | VIRTUAL | 1.0 | 7 (EQVF) |
| CH_U_R045 | CH_M_R045 | 0.85 | 1 |
| CH_U_R045 | CH_M_R030 | 0.85 | 1 |
| CH_U_L030 | CH_U_L045 | 1.0 | 0 (off) |
| CH_U_L030 | VIRTUAL | 1.0 | 7 (EQVF) |
| CH_U_L030 | CH_M_L030 | 0.85 | 1 |
| CH_U_R030 | CH_U_R045 | 1.0 | 0 (off) |
| CH_U_R030 | VIRTUAL | 1.0 | 7 (EQVF) |
| CH_U_R030 | CH_M_R030 | 0.85 | 1 |
| CH_U_L090 | CH_U_L030, CH_U_L110 | 1.0 | 0 (off) |
| CH_U_L090 | CH_U_L030, CH_U_L135 | 1.0 | 0 (off) |
| CH_U_L090 | CH_U_L045 | 0.8 | 0 (off) |
| CH_U_L090 | CH_U_L030 | 0.8 | 0 (off) |
| CH_U_L090 | VIRTUAL | 1.0 | 12 (EQVS) |
| CH_U_L090 | CH_M_L045, CH_M_L110 | 0.85 | 2 |
| CH_U_L090 | CH_M_L030, CH_M_L110 | 0.85 | 2 |
| CH_U_L090 | CH_M_L030 | 0.85 | 2 |
| CH_U_R090 | CH_U_R030, CH_U_R110 | 1.0 | 0 (off) |
| CH_U_R090 | CH_U_R030, CH_U_R135 | 1.0 | 0 (off) |
| CH_U_R090 | CH_U_R045 | 0.8 | 0 (off) |
| CH_U_R090 | CH_U_R030 | 0.8 | 0 (off) |
| CH_U_R090 | VIRTUAL | 1.0 | 12 (EQVS) |
| CH_U_R090 | CH_M_R045, CH_M_R110 | 0.85 | 2 |
| CH_U_R090 | CH_M_R030, CH_M_R110 | 0.85 | 2 |
| CH_U_R090 | CH_M_R030 | 0.85 | 2 |
| CH_U_L110 | CH_U_L135 | 1.0 | 0 (off) |
| CH_U_L110 | CH_U_L045 | 0.8 | 0 (off) |
| CH_U_L110 | CH_U_L030 | 0.8 | 0 (off) |
| CH_U_L110 | VIRTUAL | 1.0 | 14 (EQVBA) |
| CH_U_L110 | CH_M_L110 | 0.85 | 2 |
| CH_U_L110 | CH_M_L045 | 0.85 | 2 |
| CH_U_L110 | CH_M_L030 | 0.85 | 2 |
| CH_U_R110 | CH_U_R135 | 1.0 | 0 (off) |
| CH_U_R110 | CH_U_R045 | 0.8 | 0 (off) |
| CH_U_R110 | CH_U_R030 | 0.8 | 0 (off) |
| CH_U_R110 | VIRTUAL | 1.0 | 14 (EQVBA) |
| CH_U_R110 | CH_M_R110 | 0.85 | 2 |
| CH_U_R110 | CH_M_R045 | 0.85 | 2 |
| CH_U_R110 | CH_M_R030 | 0.85 | 2 |
| CH_U_L135 | CH_U_L110 | 1.0 | 0 (off) |
| CH_U_L135 | CH_U_L045 | 0.8 | 0 (off) |
| CH_U_L135 | CH_U_L030 | 0.8 | 0 (off) |
| CH_U_L135 | VIRTUAL | 1.0 | 8 (EQVB) |
| CH_U_L135 | CH_M_L110 | 0.85 | 2 |
| CH_U_L135 | CH_M_L045 | 0.85 | 2 |
| CH_U_L135 | CH_M_L030 | 0.85 | 2 |
| CH_U_R135 | CH_U_R110 | 1.0 | 0 (off) |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | | |
|---|---|---|---|
| CH_U_R135 | CH_U_R045 | 0.8 | 0 (off) |
| CH_U_R135 | CH_U_R030 | 0.8 | 0 (off) |
| CH_U_R135 | VIRTUAL | 1.0 | 8 (EQVB) |
| CH_U_R135 | CH_M_R110 | 0.85 | 2 |
| CH_U_R135 | CH_M_R045 | 0.85 | 2 |
| CH_U_R135 | CH_M_R030 | 0.85 | 2 |
| CH_U_180 | CH_U_R135, CH_U_L135 | 1.0 | 0 (off) |
| CH_U_180 | CH_U_R110, CH_U_L110 | 1.0 | 0 (off) |
| CH_U_180 | VIRTUAL | 1.0 | 10 (EQVBC) |
| CH_U_180 | CH_M_180 | 0.85 | 2 |
| CH_U_180 | CH_M_R110, CH_M_L110 | 0.85 | 2 |
| CH_U_180 | CH_U_R030, CH_U_L030 | 0.8 | 0 (off) |
| CH_U_180 | CH_M_R030, CH_M_L030 | 0.85 | 2 |
| CH_T_000 | ALL_U | 1.0 | 3 |
| CH_T_000 | VIRTUAL | 1.0 | 11 (EQVOG) |
| CH_T_000 | ALL_M | 1.0 | 4 |
| CH_L_000 | CH_M_000 | 1.0 | 13 (EQBTM) |
| CH_L_000 | CH_M_L030, CH_M_R030 | 1.0 | 13 (EQBTM) |
| CH_L_L045 | CH_M_L045 | 1.0 | 13 (EQBTM) |
| CH_L_L045 | CH_M_L030 | 1.0 | 13 (EQBTM) |
| CH_L_R045 | CH_M_R045 | 1.0 | 13 (EQBTM) |
| CH_L_R045 | CH_M_R030 | 1.0 | 13 (EQBTM) |
| CH_LFE1 | CH_LFE2 | 1.0 | 0 (off) |
| CH_LFE1 | CH_M_L030, CH_M_R030 | 1.0 | 0 (off) |
| CH_LFE2 | CH_LFE1 | 1.0 | 0 (off) |
| CH_LFE2 | CH_M_L030, CH_M_R030 | 1.0 | 0 (off) |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 115 — Normalized center frequencies of the 71 filter-bank bands**

| Normalized Frequency [0, 1] |
|---|
| 0.00458330 |
| 0.00083333 |
| 0.00208330 |
| 0.00587500 |
| 0.00979170 |
| 0.01429200 |
| 0.01979200 |
| 0.02700000 |
| 0.03541700 |
| 0.04262500 |
| 0.05675000 |
| 0.07237500 |
| 0.08800000 |
| 0.10362000 |
| 0.11925000 |
| 0.13487000 |
| 0.15050000 |
| 0.16612000 |
| 0.18175000 |
| 0.19737000 |
| 0.21300000 |
| 0.22862000 |
| 0.24425000 |
| 0.25988000 |
| 0.27550000 |
| 0.29113000 |
| 0.30675000 |
| 0.32238000 |
| 0.33800000 |
| 0.35363000 |
| 0.36925000 |
| 0.38488000 |
| 0.40050000 |
| 0.41613000 |
| 0.43175000 |
| 0.44738000 |
| 0.46300000 |
| 0.47863000 |
| 0.49425000 |
| 0.50987000 |
| 0.52550000 |
| 0.54112000 |
| 0.55675000 |
| 0.57237000 |
| 0.58800000 |
| 0.60362000 |
| 0.61925000 |
| 0.63487000 |
| 0.65050000 |
| 0.66612000 |
| 0.68175000 |
| 0.69737000 |
| 0.71300000 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| |
|---|
| 0.72862000 |
| 0.74425000 |
| 0.75987000 |
| 0.77550000 |
| 0.79112000 |
| 0.80675000 |
| 0.82237000 |
| 0.83800000 |
| 0.85362000 |
| 0.86925000 |
| 0.88487000 |
| 0.90050000 |
| 0.91612000 |
| 0.93175000 |
| 0.94737000 |
| 0.96300000 |
| 0.97454000 |
| 0.99904000 |

**Table 116 — Equalizer parameters**

| Equalizer | $P_f$ [Hz] | $P_Q$ | $P_g$[dB] | g [dB] |
|---|---|---|---|---|
| $G_{EQ,1}$ | 12000 | 0.3 | -2 | 1.0 |
| $G_{EQ,2}$ | 12000 | 0.3 | -3.5 | 1.0 |
| $G_{EQ,3}$ | 200,1300, 600 | 0.3, 0.5, 1.0 | -6.5, 1.8, 2.0 | 0.7 |
| $G_{EQ,4}$ | 5000, 1100 | 1.0, 0.8 | 4.5, 1.8 | -3.1 |
| $G_{EQ,5}$ | 35 | 0.25 | -1.3 | 1.0 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 117 — Vertically corresponding channels: Each row lists channels which are considered to be above/below each other.**

| | | |
|---|---|---|
| CH_L_000 | CH_M_000 | CH_U_000 |
| CH_L_L045 | CH_M_L030 | CH_U_L030 |
| CH_L_L045 | CH_M_L030 | CH_U_L045 |
| CH_L_L045 | CH_M_L045 | CH_U_L030 |
| CH_L_L045 | CH_M_L045 | CH_U_L045 |
| CH_L_L045 | CH_M_L060 | CH_U_L030 |
| CH_L_L045 | CH_M_L060 | CH_U_L045 |
| CH_L_R045 | CH_M_R030 | CH_U_R030 |
| CH_L_R045 | CH_M_R030 | CH_U_R045 |
| CH_L_R045 | CH_M_R045 | CH_U_R030 |
| CH_L_R045 | CH_M_R045 | CH_U_R045 |
| CH_L_R045 | CH_M_R060 | CH_U_R030 |
| CH_L_R045 | CH_M_R060 | CH_U_R045 |
| CH_M_180 | CH_U_180 | |
| CH_M_L090 | CH_U_L090 | |
| CH_M_L110 | CH_U_L110 | |
| CH_M_L135 | CH_U_L135 | |
| CH_M_L090 | CH_U_L110 | |
| CH_M_L090 | CH_U_L135 | |
| CH_M_L110 | CH_U_L090 | |
| CH_M_L110 | CH_U_L135 | |
| CH_M_L135 | CH_U_L090 | |
| CH_M_L135 | CH_U_L135 | |
| CH_M_R090 | CH_U_R090 | |
| CH_M_R110 | CH_U_R110 | |
| CH_M_R135 | CH_U_R135 | |
| CH_M_R090 | CH_U_R110 | |
| CH_M_R090 | CH_U_R135 | |
| CH_M_R110 | CH_U_R090 | |
| CH_M_R110 | CH_U_R135 | |
| CH_M_R135 | CH_U_R090 | |
| CH_M_R135 | CH_U_R135 | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

### 11.4.2 Audio signal processing

#### 11.4.2.1 General

The audio processing block of the format converter obtains time domain audio samples for $N_{\text{in}}$ channels from the core decoder and generates a downmixed time domain audio output signal consisting of $N_{\text{out}}$ channels.

The processing takes as input

— the audio data and the flag **rendering3DType** decoded by the core decoder,

— the static downmix matrixes $\mathbf{M}_{\text{DMX}}$ and $\mathbf{M}_{\text{DMX2}}$ returned by the initialization of the format converter,

It returns an $N_{\text{out}}$-channel time domain output signal for the OutConf channel configuration signaled during the initialization of the format converter.

The format converter operates on contiguous, non-overlapping frames of length $L = 2048$ time domain samples of the input audio signals and outputs one frame of $L$ samples per processed input frame of length $L$.

© ISO/IEC 2015 – All rights reserved

The **rendering3DType** is used in the selection of the downmix matrix as shown in Figure 32.

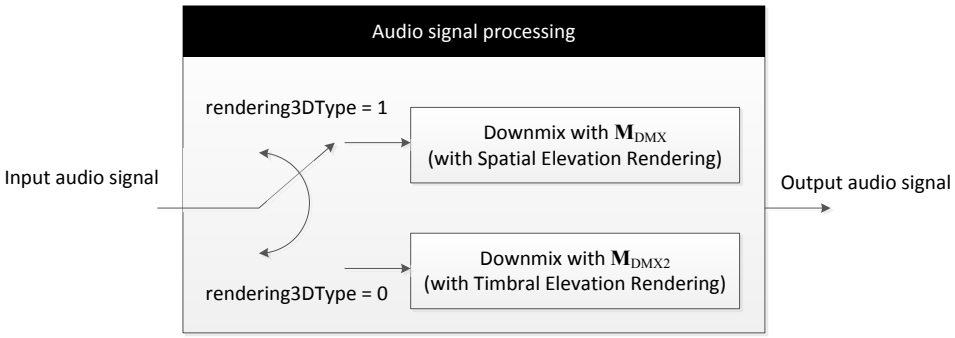



**Figure 32 — Audio signal processing with switching scheme by rendering3DType**

#### 11.4.2.2 T/F-transform (hybrid QMF analysis)

As the first processing step the converter transforms $L = 2048$ samples of the $N_{in}$ channel time domain input signal $\left[\tilde{y}^{v}_{\mathrm{ch},1} \cdots \tilde{y}^{v}_{\mathrm{ch},N_{in}}\right] = \tilde{\mathbf{y}}^{v}_{\mathrm{ch}}$ to a hybrid QMF $N_{\mathrm{in}}$ channel signal representation consisting of $L_n = 32$ QMF time slots (slot index $n$) and $K = 71$ frequency bands (band index $k$). A QMF analysis according to ISO/IEC 14496-3:2009, subclause 4.6.18.4, is performed first

$$\left[\hat{y}^{n,k}_{\mathrm{ch},1} \cdots \hat{y}^{n,k}_{\mathrm{ch},N_{in}}\right] = \hat{\mathbf{y}}^{n,k}_{\mathrm{ch}} = \mathrm{QmfAnalysis}\left(\tilde{\mathbf{y}}^{v}_{\mathrm{ch}}\right) \quad \text{with} \quad 0 \le v < L \text{ and } \quad 0 \le n < L_n,$$

followed by a hybrid analysis

$$\left[y^{n,k}_{\mathrm{ch},1} \cdots y^{n,k}_{\mathrm{ch},N_{in}}\right] = \mathbf{y}^{n,k}_{\mathrm{ch}} = \mathrm{HybridAnalysis}\left(\hat{\mathbf{y}}^{n,k}_{\mathrm{ch}}\right).$$

The hybrid filtering shall be carried out as specified in 8.6.4.3 of ISO/IEC 14496-3:2009 for the 10,20 bands configuration of parametric stereo, resulting in a 71-band hybrid QMF domain representation.

#### 11.4.2.3 Delayed channels and downmix matrix modification

If there exists a frontal height input channel to be rendered by spatial elevation rendering, delayed QMF samples of the input channel are added to the input QMF samples and downmix matrix is expanded with modified coefficients. This step enables the spatial elevation rendered signal more stable by avoiding the front-back confusion. The process is done by :

For each input channel $i$ in [1 $N_{in}$], do

If the $i^{th}$ input channel has attached one of the labels CH_U_L030, CH_U_L045, CH_U_R030, CH_U_R045, and CH_U_000

If the channel $i$ is to be rendered by spatial rendering

(1) Create delayed QMF samples of the input channel by:

$$delay = \mathrm{round}\left(f_s \times 0.003/64\right)$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$\mathbf{y}_{\mathrm{ch}}^{\mathrm{n,k}} = \begin{cases} \left[\mathbf{y}_{\mathrm{ch}}^{\mathrm{n,k}} \quad y_{\mathrm{ch},i}^{n-delay,k}\right] & \text{for n} \geq \text{delay} \\ \left[\mathbf{y}_{\mathrm{ch}}^{\mathrm{n,k}} \quad y_{\mathrm{ch,prev},i}^{L_n+(n-delay),k}\right] & \text{otherwise} \end{cases},$$

where $y_{\mathrm{ch,prev},i}^{L_n+(n-delay),k}$ is the $L_n + (n - delay)$ th QMF subband sample in k[th] band of the previous frame

(2) Modify the downmix matrix $\mathbf{M}_{\mathrm{DMX}}$ for the CH_M_L110 and CH_M_R110

$$\mathbf{M}_{\mathrm{DMX}} = [\mathbf{M}_{\mathrm{DMX}} \quad M_{DMX,1\sim N_{out},i}]$$

$$\mathbf{M}_{\mathrm{DMX2}} = [\mathbf{M}_{\mathrm{DMX2}} \quad [0\ 0 \ldots 0]^{\mathrm{T}}]$$

$$N_{in} = N_{in} + 1$$

For each $j$ in [1 $N_{out}$], do

If the channel $j^{\mathrm{th}}$ output channel has attached the label of either CH_M_L110 or CH_M_R110

$$M_{DMX,j,i} = 0$$

Else

$$M_{DMX,j,N_{\mathrm{in}}} = 0$$

Explanation : By reproducing the spatial elevation rendered output surround channel for frontal height input channel with delay approximately 3 msec, front-back confusion is avoided by precedence effect. The delay for 48 kHz sampled input signal is 2 subband samples.

#### 11.4.2.4 Covariance analysis

Note that for clarity the frequency band parameter (superscript $k$) is omitted in the following equations if it is not required for the presentation.

Let $F$ be a monotonically increasing frame index denoting the current frame of input data, e.g. $\mathbf{y}_{\mathrm{ch}}^{F,n} = \mathbf{y}_{\mathrm{ch}}^{n}$ for frame $F$, starting at $F = 0$ for the first frame of input data after initialization of the format converter. An analysis frame of length $2L_n$ is formulated from the input hybrid QMF spectra as

$$\mathbf{y}_{\mathrm{in,\,ch}}^{F,n} = \begin{cases} 0 & \text{for} \quad 0 \leq n < L_n \quad , \quad F = 0 \\ \mathbf{y}_{\mathrm{in,\,ch}}^{F-1,n+L_n} & \text{for} \quad 0 \leq n < L_n \quad , \quad F > 0 \\ \mathbf{y}_{\mathrm{ch}}^{F,n-L_n} & \text{for} \quad L_n \leq n < 2L_n \quad , \quad F \geq 0 \end{cases}.$$

Note that $\mathbf{y}_{\mathrm{in,\,ch}}^{F,n}$ is a row vector with $N_{\mathrm{in}}$ elements in case of $N_{\mathrm{in}}$ input channels. The covariance matrix is analysed from four quarter segments of $\mathbf{y}_{\mathrm{in,\,ch}}^{F,n}$, so that

$$\mathbf{C}_{y,q}^{F} = \sum_{n=16q}^{16q+15} \left(\mathbf{y}_{\mathrm{in,\,ch}}^{F,n}\right)^{\mathrm{T}} \left(\mathbf{y}_{\mathrm{in,\,ch}}^{F,n}\right)^{*} \quad \text{for} \quad q = 0,1,2,3\,,$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

where $(\cdot)^{\mathrm{T}}$ denotes the transpose and $(\cdot)^{*}$ denotes the complex conjugate of a variable and $\mathbf{C}_{y,q}^{F}$ is an $N_{\text{in}}$ x $N_{\text{in}}$ matrix for each $q = 0,1,2,3$.

Note that $\mathbf{C}_{y,0}^{F}$ and $\mathbf{C}_{y,1}^{F}$ are the same as $\mathbf{C}_{y,2}^{F-1}$ and $\mathbf{C}_{y,3}^{F-1}$, correspondingly, and are not necessary to be re-calculated. The covariance matrices of the four quarter segments are added with center weighting assuming a staircase shape:

$$\mathbf{C}_{y,\text{sum}}^{\text{F}} = \mathbf{C}_{y,0}^{F} + 4\mathbf{C}_{y,1}^{F} + 4\mathbf{C}_{y,2}^{F} + \mathbf{C}_{y,3}^{F}.$$

The final estimation for the covariance matrix $\mathbf{C}_{y}^{F}$ is obtained by modifying the entries of $\mathbf{C}_{y,\text{sum}}^{F}$ with a small channel dependent offset

$$C_{y,a,b} = (1 + 0.0002\sqrt{ab})C_{y,\text{sum},a,b}$$

where the two indices in a notation $C_{y,a,b}$ denote the matrix element in the $a$ th row and $b$ th column of $\mathbf{C}_{y}$. From the covariance matrix $\mathbf{C}_{y}$ inter-channel correlation coefficients between the channels $A$ and $B$ are derived as

$$ICC_{A,B} = \sqrt{\frac{\left|C_{y,A,B}\right|^{2}}{\text{eps} + C_{y,A,A} \cdot C_{y,B,B}}}.$$

#### 11.4.2.5 Phase-alignment matrix formulation

11.4.2.5.1 **General**

The $ICC_{A,B}$ values are mapped to an attraction measure matrix $\mathbf{T}$ with elements

$$T_{A,B} = \begin{cases} \min\left(PasMax, \max\left(0, PasCurveSlope \cdot ICC_{A,B} + PasCurveShift\right)\right) & \text{for} \quad A \neq B \\ \min\left(1.0, \max\left(0, PasCurveSlope \cdot 4 \cdot ICC_{A,B} + 4 \cdot PasCurveShift\right)\right) & \text{for} \quad A = B \end{cases},$$

where *PasMax, PasCurveSlope, PasCurveShift* are derived from Table 118

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 118 — Phase attraction mapping curve parameters.**
**phaseAlignStrength shall be set to 3 if no other value has been signaled in the bitstream.**
**If passiveDownmixFlag==1, then phaseAlignStrength shall be set to 0.**

| phaseAlignStrength | PasMax | PasCurveSlope | PasCurveShift |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0.0714 | 0.1701 | -0.0891 |
| 2 | 0.1548 | 0.3771 | -0.1896 |
| 3 | 0.25 | 0.625 | -0.3 |
| 4 | 0.3571 | 0.9184 | -0.4184 |
| 5 | 0.4762 | 1.2623 | -0.5427 |
| 6 | 0.6071 | 1.6624 | -0.6707 |
| 7 | 0.75 | 2.125 | -0.8 |

An intermediate phase-aligning mixing matrix $\mathbf{M}_{\text{int}}$ is formulated. With

$P_{A,B} = T_{A,B} \cdot C_{y,A,B}$ ,

formulate $\mathbf{M}_{\text{int}}$ depending on the **rendering3DType** by:

If **rendering3DType** == TRUE

$\mathbf{V} = \mathbf{M}_{\text{DMX}}\mathbf{P}$

If the input channel A is rendered by spatial elevation rendering,

$M^k_{int,B,A} = M^k_{DMX,B,A}$ for $f_k \times f_s / 2 \geq 2800$ and $f_k \times f_s / 2 \leq 10000$

$M^k_{int,B,A} = \frac{M^k_{DMX,B,A}}{eps + |V_{B,A}|} V_{B,A}$ otherwise

else

$M^k_{int,B,A} = \frac{M^k_{DMX,B,A}}{eps + |V_{B,A}|} V_{B,A}$

else

$\mathbf{V} = \mathbf{M}_{\text{DMX2}}\mathbf{P}$

$M^k_{int,B,A} = \frac{M^k_{DMX2,B,A}}{eps + |V_{B,A}|} V_{B,A}$ for *all k*

where $f_k$ is the normalized center frequency of frequency band $k$, specified in **Table 115**, $f_s$ is the sampling frequency, and $M^k_{DMX,A,B}$ and $M^k_{DMX2,A,B}$ denotes the matrix element in the $A$ th row and $B$ th column of $\mathbf{M}_{\text{DMX}}$ and $\mathbf{M}_{\text{DMX2}}$.

Note that the phase alignment for the frequency components of 2.8~10 kHz for the spatial elevation rendered downmix coefficients are avoided for the spatial elevation rendering when the **rendering3DType** is true. Although the immersive format converter has an active downmix with phase correction, the synchronization of the spatial elevation rendered signal from a height input channel without phase correction plays an important role to provide overhead sound image in 3D rendering.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

The intermediate phase-aligning mixing matrix $\mathbf{M}_{\text{int}}$ is modified to avoid abrupt phase shifts, resulting in $\mathbf{M}_{\text{mod}}$. This is a recursive process, running for each frame *F*, processing the frequency bands *k* in ascending order.

The regularization against phase shifts takes place in two stages: In the first stage, the regularization performs amplitude-weighted phase comparison against the previous frame, previous band, while also linking the phase-attracted channels. In the second stage, the regularization limits the update rate of the phase coefficients in comparison to the previous frame only.

Both regularization stages make use of a phase update limiting parameter, $\theta_{\text{diff},A}^{F,k}$, which is formulated as function of an onset measure $o_A^{F,k}$ so that a low energy portion of a signal does not affect the phase processing after an onset:

$$o_A^{F,k} = \frac{C_{y,A,A}^{F,k}}{eps + C_{y,A,A}^{F,k} + C_{y,A,A}^{F-1,k}},$$

$$\theta_{\text{diff},A}^{F,k} = \max(0.15, 20.3 o_A^{F,k} - 19)\pi .$$

11.4.2.5.2 **Regularization Stage 1:**

Stage 1 recursively takes into account comparison values $\mathbf{M}_{\text{cmp}}$ from the last frame index (*F*-1) as well as for the last processing band (*k*-1). $\mathbf{M}_{\text{cmp}}$ is derived from $\mathbf{M}_{\text{mod}}$ at the end of the regularization process. The first step of regularization stage 1 combines the comparison data across frequency and time as follows:

if (F=0)

$$\mathbf{M}_{\text{cmpFk}}^{F,k} = \begin{cases} 0 & \text{for} \quad k = 1 \\ \mathbf{M}_{\text{cmp}}^{\text{F,k-1}} & \text{for} \quad k > 1 \ \wedge \ k \neq 3 \\ \left(\mathbf{M}_{\text{cmp}}^{\text{F,k-1}}\right)^* & \text{for} \quad k = 3 \end{cases}$$

else (i.e. for F>0)

$$\mathbf{M}_{\text{cmpFk}}^{F,k} = \begin{cases} \mathbf{M}_{\text{cmp}}^{F-1,k} & \text{for} \quad k = 1 \\ \mathbf{M}_{\text{cmp}}^{F-1,k} + \mathbf{M}_{\text{cmp}}^{\text{F,k-1}} & \text{for} \quad k > 1 \ \wedge \ k \neq 3 \\ \mathbf{M}_{\text{cmp}}^{F-1,k} + \left(\mathbf{M}_{\text{cmp}}^{\text{F,k-1}}\right)^* & \text{for} \quad k = 3 \end{cases}$$

where the complex conjugate processing for the third band (*k*=3) accounts for the complex conjugate properties of the filterbank.

The frequency index *k* is omitted in the following since the inter-band dependency is now contained in the matrix $\mathbf{M}_{\text{cmpFk}}^{F,k}$. The phase change of the current unregularized phase-aligning matrix $\mathbf{M}_{\text{int}}^{F}$ relative to

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$\mathbf{M}^{F,k}_{\text{cmpFk}}$ is measured by amplitude weighting with $\sqrt{C^{F}_{y,A,A}}$ and comparison against $\mathbf{M}^{F}_{\text{cmpFk}}$, forming $\mathbf{M}^{F}_{\text{timeFreq}}$ with elements

$$M^{F}_{\text{timeFreq},B,A} = M^{F}_{\text{cmpFk},B,A}\left(M^{F}_{\text{int},B,A}\right)^{*}\sqrt{C^{F}_{y,A,A}}\ .$$

To also take into account the interdependent channels in the regularization, the relevant entries are intermixed with the attraction matrix $\mathbf{T}^{F}$,

$$\mathbf{M}^{F}_{\text{timeFreqChan}} = \mathbf{M}^{F}_{\text{timeFreq}}\mathbf{T}^{F}\ .$$

The phase values of the elements of matrix $\mathbf{M}^{F}_{\text{timeFreqChans}}$ are

$$\theta^{F}_{\text{timeFreqChan},B,A} = \arg\left(M^{F}_{\text{timeFreqChan},B,A}\right).$$

To avoid constant phase offsets, $\theta^{F}_{\text{timeFreqChan},B,A}$ is adjusted towards zero by $\theta^{F}_{\text{diff},A}$:

$$\theta^{F}_{\text{timeFreqChanStage1},B,A} = \text{sign}\left(\theta^{F}_{\text{timeFreqChan},B,A}\right)\max\left(0, abs(\theta^{F}_{\text{timeFreqChan},B,A}) - \theta^{F}_{\text{diff},A}\right)$$

11.4.2.5.3 **Regularization Stage 2:**

In stage 2 of the regularization another phase comparison parameter, only across time, is formulated:

$$\theta^{F}_{\text{time},B,A} = \arg\left(M^{F-1}_{\text{PA},B,A}\left(M^{F}_{\text{int},B,A}\right)^{*}\right)$$

The final regularization parameter is such that is as close as possible to $\theta^{F}_{\text{timeFreqChanStage1},B,A}$, but not further than $\theta^{F}_{\text{diff},A}$ in respect to $\theta^{F}_{\text{time},B,A}$. Let *unwrap()* be a function that maps any angular parameter to the corresponding angle in the interval $-\pi\ldots\pi$. The final phase parameter is formulated as

$$\theta^{F}_{\text{update},B,A} = unwrap\left(\theta^{F}_{\text{timeFreqChanStage1},B,A} - \theta^{F}_{\text{time},B,A}\right),$$
$$\theta^{F}_{\text{mod},B,A} = \theta^{F}_{\text{time},B,A} + sign\left(\theta^{F,k}_{\text{update},B,A}\right)\max(0, abs(\theta^{F}_{\text{update},B,A}) - \theta^{F}_{\text{diff},A}),$$

and the modified, i.e. phase-regularized, mixing matrix elements are obtained as

$$M^{F}_{\text{mod},B,A} = M^{F}_{\text{int},B,A}\cdot\exp\left(j\cdot\theta^{F}_{\text{mod},B,A}\right).$$

Finally, $\mathbf{M}_{\text{cmp}}$ is derived by amplitude weighting the regularized downmixing coefficients,

$$M^{F}_{\text{cmp,B,A}} = M^{F}_{\text{mod},B,A}\sqrt{C^{F}_{y,A,A}}\ .$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

Note that $\mathbf{M}_{\text{cmp}}$ is used in the time and frequency recursive formulation of regularization stage 1.

11.4.2.5.4 **Energy Scaling:**

An energy scaling is applied to the mixing matrix to obtain the final phase-aligning mixing matrix $\mathbf{M}_{\text{PA}}$. With

$$\mathbf{M}_{Cy} = \mathbf{M}_{\text{mod}} \mathbf{C}_y \mathbf{M}_{\text{mod}}^{\text{H}},$$

where $(\cdot)^{\text{H}}$ denotes the conjugate transpose operator, and

if **rendering3DType** == TRUE

$$S_B = \sqrt{\frac{\sum_{A=1}^{N_{in}} M_{DMX,B,A} \cdot M_{DMX,B,A} \cdot C_{y,A,A}}{eps + M_{C_y,B,B}}}$$

Else

$$S_B = \sqrt{\frac{\sum_{A=1}^{N_{in}} M_{DMX2,B,A} \cdot M_{DMX2,B,A} \cdot C_{y,A,A}}{eps + M_{C_y,B,B}}},$$

$$S_{\text{lim},B} = \min\left(S_{\max}, \max\left(S_{\min}, S_B\right)\right),$$

where the limits are defined as $S_{\max} = 10^{0.4}$ and $S_{\min} = 10^{-0.5}$, the final phase-aligning mixing matrix elements follow as

$$M_{\text{PA},B,A} = \left(S_{\text{lim},B} \cdot AES + (1 - AES)\right) \cdot M_{\text{mod},B,A},$$

where *AES* = (1 – **passiveDownmixFlag**).

### 11.4.2.6 Calculation of output data

The output signals for the current frame $F$ are formulated by linearly interpolating the mixing matrices from the previous frame to the current frame,

$$\mathbf{z}_{\text{ch}}^{F,n} = \left( \left( \frac{n+1}{L_n} \mathbf{M}_{\text{PA}}^{F} + \frac{L_n - n - 1}{L_n} \mathbf{M}_{\text{PA}}^{F-1} \right) \left( \mathbf{y}_{\text{in, ch}}^{F,n} \right)^{\text{T}} \right)^{\text{T}} \quad \text{for} \quad 0 \le n < L_n .$$

Note that the input audio for the above mixing procedure is the first half of the analysis window.

### 11.4.2.7 F/T-transform (hybrid QMF synthesis)

Note that the processing steps described above have to be carried out for each hybrid QMF band $k$ (recursively, for ascending $k$). In the following formulations the band index $k$ is reintroduced, i.e. $\mathbf{z}_{\text{ch}}^{F,n,k} = \mathbf{z}_{\text{ch}}^{F,n}$. The hybrid QMF frequency domain output signal $\mathbf{z}_{\text{ch}}^{F,n,k}$ is transformed to an $N_{\text{out}}$-channel time

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

domain signal frame of length $L$ time domain samples per output channel $B$, yielding the time domain output signal $\tilde{\mathbf{z}}_{\mathrm{ch}}^{F,\nu}$:

The hybrid synthesis

$$\hat{\mathbf{z}}_{\mathrm{ch}}^{F,n,k} = \mathrm{HybridSynthesis}\left(\mathbf{z}_{\mathrm{ch}}^{F,n,k}\right)$$

is carried out as defined in Figure 8.21 of ISO/IEC 14496-3:2009, i.e. by summing the sub-subbands of the three lowest QMF subbands to obtain the three lowest QMF subbands of the 64band QMF representation. The subsequent QMF synthesis

$$\tilde{\mathbf{z}}_{\mathrm{ch}}^{F,\nu} = \mathrm{QMFSynthesis}\left(\hat{\mathbf{z}}_{\mathrm{ch}}^{F,n,k}\right)$$

shall be carried out as defined in ISO/IEC 14496-3:2009, subclause 4.6.18.4.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

# 12 Higher Order Ambisonics (HOA)

## 12.1 Technical Overview

### 12.1.1 Block Diagram

A block diagram of the HOA decoder architecture is depicted in Figure 33. First, the input bit stream is de-multiplexed and decoded by the MPEG-H 3D Audio Core decoder into $I$ PCM transport channels plus the HOA bit stream that contains parameters required to recompose the full HOA representation from these PCM signals. In the successive spatial decoding component, first, the actual value range of these signals is reconstructed by the inverse gain control processing. In a next step, the $I$ signals are re-distributed to provide the $M$ predominant signals and $(I - M)$ HOA coefficient signals representing the more ambient HOA components.

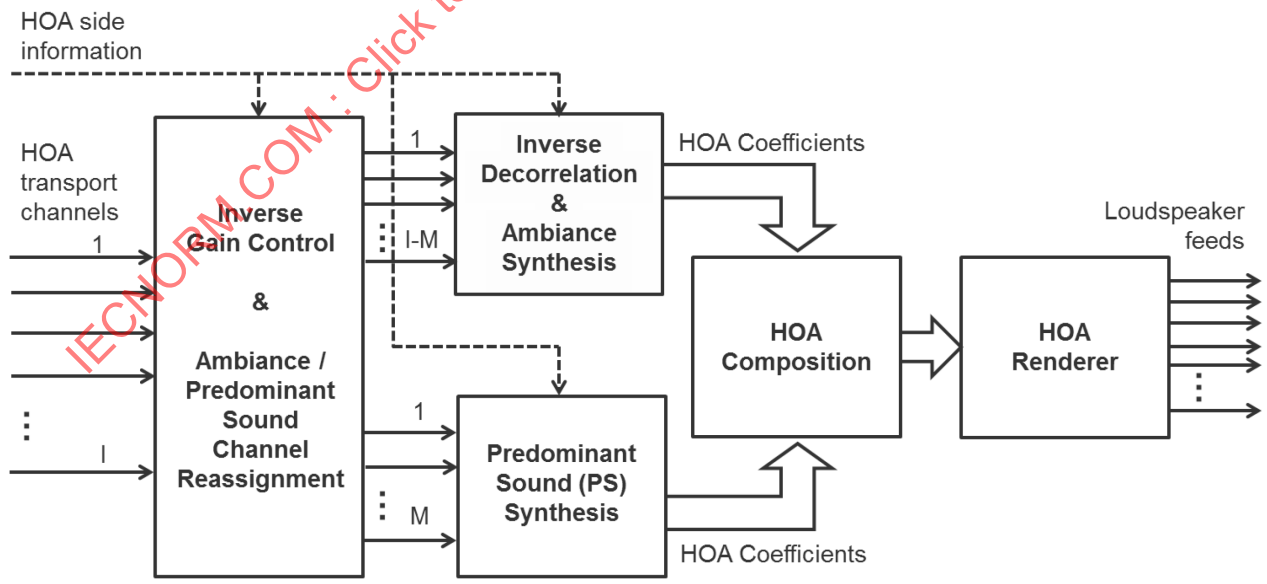



**Figure 33 — Simplified block diagram of the decoder**

© ISO/IEC 2015 – All rights reserved

The fixed subset of the $(I - M)$ HOA coefficient signals is re-correlated, this means the decorrelation at the HOA encoding stage is reversed. Next, all of the $(I - M)$ HOA coefficient signals are used to create the ambient HOA components. The predominant HOA components are synthesized from the $M$ predominant signals and the corresponding parameters. Finally, the predominant and the ambient HOA components are composed into the desired full HOA representation, which is then rendered to a given loudspeaker setup.

### 12.1.2 Overview of the decoder tools

#### 12.1.2.1 HOA Decoding Tools

The inputs of the HOA frame converter are the HOA configuration data HOAConfig() and the HOA Frame HOAFrame(), as it is depicted in Figure 34. The variables of Figure 34 are defined in 12.3 and 12.4.1.

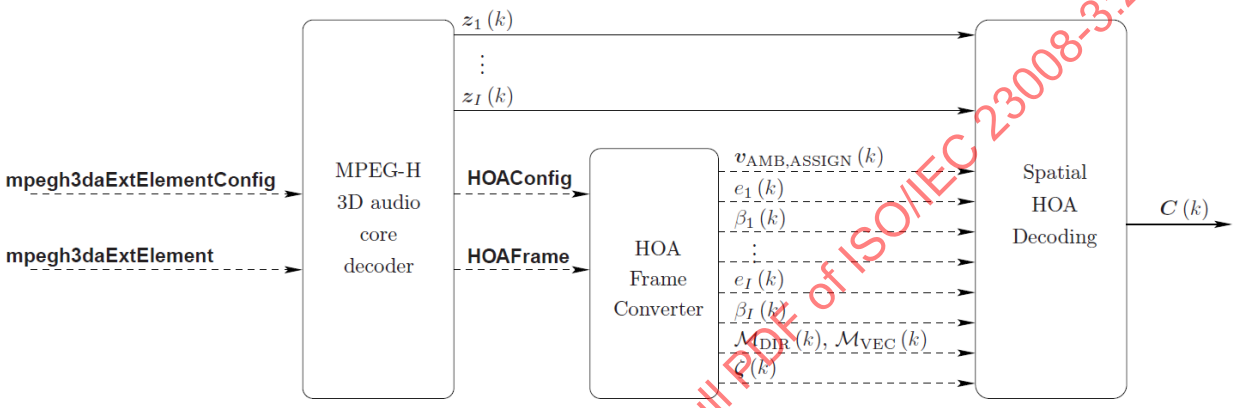



**Figure 34 — The architecture of the HOA decoder tools**

The HOA frame converter (Convert) and the MPEG-H 3D Audio Core decoder provide the data for the Spatial HOA Decoding.

The Spatial HOA Decoding re-creates the HOA time domain signals of the previous frame from the signals $z_i(k)$ and from the spatial side information provided by the HOA Frame Converter. The Spatial HOA decoding consists of the following coding tools, which are specified in 12.4.

— Inverse Gain Control

— Channel Reassignment

— Predominant Sound Synthesis

— Ambient Synthesis

— HOA Compositions

#### 12.1.2.2 HOA Renderer

The HOA Renderer converts the HOA signal matrix $\boldsymbol{C}(k)$ to the loudspeaker signals $\boldsymbol{W}_{\mathtt{SPEAKER}}(k)$ using the loudspeaker position matrix $\boldsymbol{\Omega}_{\mathtt{SPEAKER}}$ and the HOAConfig() (see 12.3.1) for its initialization.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

## 12.2 Syntax

### 12.2.1 Configuration of HOA elements

**Table 119 — Syntax of HOAConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HOAConfig()<br>{<br>  HoaOrder = escapedValue(3,5,0);<br>  NumOfHoaCoeffs = ( HoaOrder + 1 )^2;<br>  **UsesNfc;**<br>  if (UsesNfc) {<br>  **NfcReferenceDistance;**<br>  }<br>  HOADecoderConfig();<br>} | <br><br>**3,8**<br><br>**1**<br><br>**32** | <br><br>**uimsbf**<br><br>**bslbf**<br><br>**bslbf** |
| NOTE: HoaOrder = 30 … 38 are reserved. | | |

**Table 120 — Syntax of HOADecoderConfig()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HOADecoderConfig(numHOATransportChannels)<br>{<br>  MinAmbHoaOrder = escapedValue(3,5,0) – 1;<br>  MinNumOfCoeffsForAmbHOA = (MinAmbHoaOrder + 1)^2;<br>  NumOfAdditionalCoders = numHOATransportChannels –<br>  MinNumOfCoeffsForAmbHOA;<br>  **SingleLayer;**<br>  MaxNoOfDirSigsForPrediction = **MaxNoOfDirSigsForPrediction** + 1;<br>  NoOfBitsPerScalefactor = **NoOfBitsPerScalefactor** + 1**;**<br>  **CodedSpatialInterpolationTime;**<br>  **SpatialInterpolationMethod;**<br>  **CodedVVecLength;**<br>  **MaxGainCorrAmpExp;**<br>  **HOAFrameLengthIndicator;**<br>  MaxNumAddActiveAmbCoeffs = NumOfHoaCoeffs –<br>  MinNumOfCoeffsForAmbHOA;<br>  AmbAsignmBits = ceil( log2( MaxNumAddActiveAmbCoeffs ) );<br>  ActivePredIdsBits = ceil( log2( NumOfHoaCoeffs ) );<br>  i = 1;<br>  while( i * ActivePredIdsBits<br>  + ceil( log2( i ) ) < NumOfHoaCoeffs ){<br>  i++;<br>  }<br>  NumActivePredIdsBits = ceil( log2( max( 1, i – 1 ) ) );<br>  GainCorrPrevAmpExpBits = ceil( log2( ceil( log2(<br>  1.5 * NumOfHoaCoeffs ) )<br>  + MaxGainCorrAmpExp + 1 ) );<br>  for (i=0; i<NumOfAdditionalCoders; ++i){<br>  AmbCoeffTransitionState[i] = 3;<br>  }<br>} | <br><br>**3,8**<br><br><br><br>**1**<br>**2**<br>**4**<br>**3**<br>**1**<br>**2**<br>**3**<br>**2** | <br><br>**uimsbf**<br><br><br><br>**bslbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**bslbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf** |
| NOTE: MinAmbHoaOrder = 30 … 37 are reserved. HOAFrameLengthIndicator = 3 is reserved. CodedVVecLength = 3 is reserved. | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

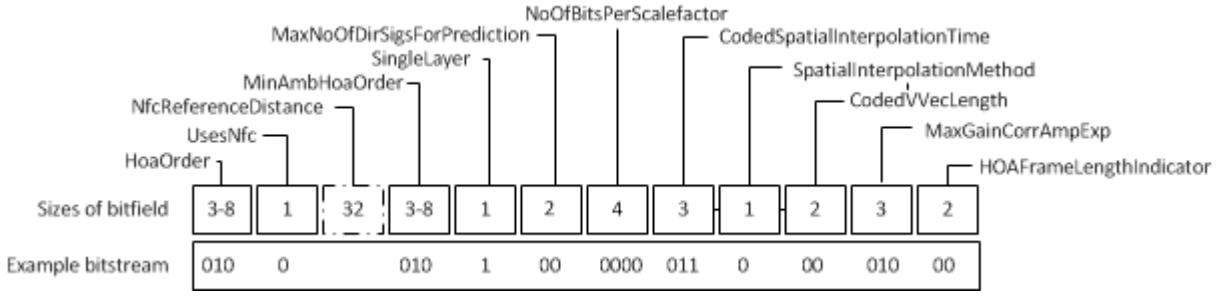



**Figure 35 — Example for HOAConfig()**

### 12.2.2 Payloads of HOA elements

**Table 121 — Syntax of HOAFrame**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HOAFrame()<br>{<br>NumOfDirSigs = 0;<br>NumOfVecSigs = 0;<br>NumOfContAddHoaChans = 0;<br>**hoaIndependencyFlag**; | **1** | **bslbf** |
| for(i=0; i< NumOfAdditionalCoders; ++i){<br>ChannelSideInfoData(i);<br>HOAGainCorrectionData(i);<br>switch ChannelType[i] {<br>case 0:<br>DirSigChannelIds[NumOfDirSigs] = i + 1;<br>NumOfDirSigs++;<br>break;<br>case 1:<br>VecSigChannelIds[NumOfVecSigs] = i + 1;<br>NumOfVecSigs++;<br>break;<br>case 2:<br>if (AmbCoeffTransitionState[i] == 0) {<br>ContAddHoaCoeff [NumOfContAddHoaChans] =<br>AmbCoeffIdx[i];<br>NumOfContAddHoaChans++;<br>}<br>break;<br>}<br>} | | |
| for ( i= NumOfAdditionalCoders;<br>i< NumHOATransportChannels; ++i){<br>HOAGainCorrectionData(i);<br>} | | |
| for(i=0; i< NumOfVecSigs; ++i){<br>VVectorData ( VecSigChannelIds(i) );<br>} | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
    if(NumOfDirSigs > 0){
        HOAPredictionInfo( DirSigChannelIds, NumOfDirSigs )
    }
}
```

NOTE: the encoder shall set hoaIndependencyFlag to 1 if usacIndependencyFlag (see mpegh3daFrame() in Table 26) is set to 1.

**Table 122 — Syntax of ChannelSideInfoData(i)**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| ChannelSideInfoData(i) | | |
| { | | |
| **ChannelType[i]** | **2** | **uimsbf** |
| switch ChannelType[i] | | |
| { | | |
| case 0: | | |
| **ActiveDirsIds[i];** | **10** | **uimsbf** |
| break; | | |
| case 1: | | |
| if(hoaIndependencyFlag){ | | |
| **NbitsQ**(k)[i] | **4** | **uimsbf** |
| if (NbitsQ(k)[i] == 4) { | | |
| **CodebkIdx(k)[i];** | **3** | **uimsbf** |
| } | | |
| elseif (NbitsQ(k)[i] >= 6) { | | |
| PFlag(k)[i] = 0; | | |
| **CbFlag**(k)[i]; | **1** | **bslbf** |
| } | | |
| } | | |
| else{ | | |
| **bA;** | **1** | **bslbf** |
| **bB;** | **1** | **bslbf** |
| if ((bA + bB) == 0){ | | |
| NbitsQ(k)[i] = NbitsQ(k-1)[i]; | | |
| PFlag(k)[i] = PFlag(k-1)[i]; | | |
| CbFlag(k)[i] = CbFlag(k-1)[i]; | | |
| CodebkIdx(k)[i] = CodebkIdx(k-1)[i]; | | |
| } | | |
| else{ | | |
| NbitsQ(k)[i] = (8*bA)+(4*bB)+**uintC**; | **2** | **uimsbf** |
| if (NbitsQ(k)[i] == 4) { | | |
| **CodebkIdx(k)[i]**; | **3** | **uimsbf** |
| } | | |
| elseif (NbitsQ(k)[i] >= 6) { | | |
| **PFlag**(k)[i]; | **1** | **bslbf** |
| **CbFlag**(k)[i]; | **1** | **bslbf** |
| } | | |
| } | | |
| } | | |
| break; | | |
| case 2: | | |
| AddAmbHoaInfoChannel(i); | | |
| break; | | |
| default: | | |
| } | | |
| } | | |
| NOTE CodebkIdx = 1 … 6 are reserved. | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 123 — ChannelType definition**

| ChannelType: |
|---|
| 0 : Direction-based Signal |
| 1 : Vector-based Signal |
| 2 : Additional Ambient HOA Coefficient |
| 3: Empty |

**Table 124 — Syntax of AddAmbHoaInfoChannel(i)**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HOAAddAmbInfoChannel(i) | | |
| { | | |
| if(hoaIndependencyFlag){ | | |
| **AmbCoeffTransitionState**[i]; | **2** | **uimsbf** |
| AmbCoeffIdx[i] = **CodedAmbCoeffIdx** + 1 + MinNumOfCoeffsForAmbHOA; | **AmbAsign mBits** | **uimsbf** |
| } | | |
| else { | | |
| if(**AmbCoeffIdxTransition** == 1) { | **1** | **bslbf** |
| if (AmbCoeffTransitionState[i] > 1) { | | |
| AmbCoeffTransitionState[i] = 1; | | |
| AmbCoeffIdx[i] = **CodedAmbCoeffIdx** + 1 + MinNumOfCoeffsForAmbHOA; | **AmbAsign mBits** | **uimsbf** |
| } | | |
| else { | | |
| AmbCoeffTransitionState[i] = 2; | | |
| } | | |
| } | | |
| else { | | |
| AmbCoeffTransitionState[i] = 0; | | |
| } | | |
| } | | |
| } | | |
| NOTE: | | |
| The AmbCoeffIdx of the preceding frame is used under the following conditions | | |
| if (AmbCoeffIdxTransitionState == 0 \|\| AmbCoeffIdxTransitionState == 2) | | |
| AmbCoeffTransitionState: | | |
| 0: No transition (continuous Additional Ambient HOA Coefficient) | | |
| 1: Fade-in of Additional Ambient HOA Coefficient | | |
| 2: Fade-out of Additional Ambient HOA Coefficient | | |
| 3: Initial value | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 125 — Syntax of HOAGainCorrectionData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HOAGainCorrectionData(i)<br>{<br>if(hoaIndependencyFlag){<br>GainCorrPrevAmpExp[i] = **GainCorrPrevAmpExp** - ceil( log2( 1.5 * NumHoaCoeffs ) );<br>}<br>n=0;<br>while(1) {<br>**CodedGainCorrectionExp[i][n]**<br>if(CodedGainCorrectionExp[i][n])<br>break;<br>n++;<br>}<br>**GainCorrectionException[i];**<br>} | **GainCorrPrevAmpExpBits**<br>**1**<br>**1** | **uimsbf**<br>**bslbf**<br>**bslbf** |

**Table 126 — Syntax of VVectorData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| VVectorData(i)<br>{<br>if (NbitsQ(k)[i] == 4){<br>NumVvecIndices = CodebkIdx(k)[i] +1;<br>If (CodebkIdx(k)[i] == 0) {<br>VvecIdx[0] = **VvecIdx** + 1;<br>WeightVal[0] = ((**SgnVal***2)-1);<br>} else {<br>**WeightIdx**;<br>nbitsIdx = ceil(log2(NumOfHoaCoeffs));<br>for (j=0; j< NumVvecIndices; ++j) {<br>VvecIdx[j] = **VvecIdx** + 1;<br>WeightVal[j] = ((**SgnVal***2)-1) * WeightValCdbk[CodebkIdx(k)[i]][WeightIdx][j];<br>}<br>}<br>}<br>else if (NbitsQ(k)[i] == 5) {<br>for (m=0; m< VVecLength; ++m){<br>aVal[i][m] = (**VecVal** / 128.0) – 1.0;<br>} | **10**<br>**1**<br>**8**<br>**nbitsIdx**<br>**1**<br>**8** | **uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| else if(NbitsQ(k)[i] >= 6) { | | |
| for (m=0; m< VVecLength; ++m){ | | |
| huffIdx = *huffSelect*(VVecCoeffId[m], PFlag[i], CbFlag[i]); | | |
| cid = *huffDecode*(NbitsQ[i], huffIdx, **huffVal**); | **dynamic** | **huffDecode** |
| aVal[i][m] = 0.0; | | |
| if ( cid > 0 ) { | | |
| aVal[i][m] = sgn = (**sgnVal** * 2) - 1; | **1** | **bslbf** |
| if (cid > 1) { | | |
| aVal[i][m] = sgn * (2.0^(cid -1 ) + **intAddVal**); | **cid-1** | **uimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| } | | |
| NOTE: See 12.4.1.10.2 for computation of VVecLength | | |

**Table 127 — Syntax of HOAPredictionInfo(DirSigChannelIds, NumOfDirSigs)**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| HOAPredictionInfo(DirSigChannelIds, NumOfDirSigs) | | |
| { | | |
| PredIdsBits = ceil( log2( NumOfDirSigs + 1 ) ); | | |
| if(**PSPredictionActive**){ | **1** | **bslbf** |
| NumActivePred = 0; | | |
| if(**KindOfCodedPredIds**){ | **1** | **bslbf** |
| NumActivePred = **NumActivePredIds** + 1; | **NumActivePredIdsBits** | **uimsbf** |
| i=0; | | |
| while( i < NumActivePred){ | | |
| PredIds[i] = **PredIds**[i] + 1; | **ActivePredIdsBits** | **uimsbf** |
| i++; | | |
| } | | |
| } | | |
| else{ | | |
| for (i=0; i<(HoaOrder +1)^2; i++) { | | |
| if(**ActivePred[i]**) { | **1** | **bslbf** |
| NumActivePred ++; | | |
| } | | |
| } | | |
| } | | |
| NumOfGains=0; | | |
| for (i=0; i<NumActivePred * MaxNoOfDirSigsForPrediction; i++) { | | |
| if (**PredDirSigIds[i]** > 0) { | **PredIdsBits** | **uimsbf** |
| PredDirSigIds[i] = | | |
| DirSigChannelIds[PredDirSigIds[i] - 1 ]; | | |
| NumOfGains++; | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
            n=0;
            for (i=0; i< NumOfGains; i++) {
                if (PredDirSigIds[i]>0) {
                    PredGains[n];                    NoOfBitsPerScalefactor    bslbf
                    n++;
                }
            }
        }
}
```

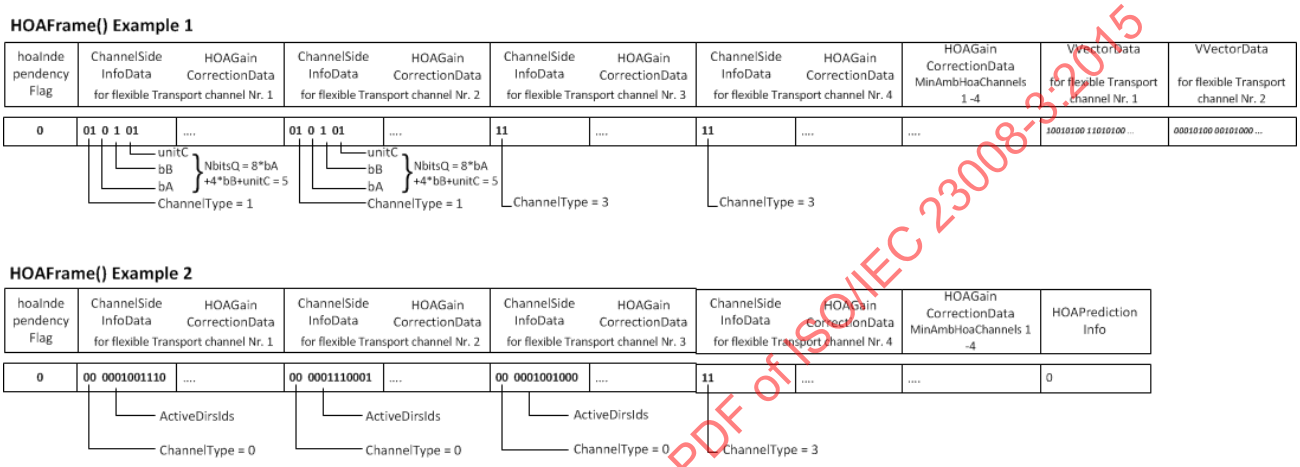



**Figure 36 — Two examples for HOAFrame()**

## 12.3 Data Structure

### 12.3.1 Definitions of HOA Config

HOAConfig() This element contains information about the audio content

**HoaOrder** This element determines the HOA order of the coded signal by HoaOrder = **HoaOrder** +1.

NumOfHoaCoeffs This element determines the number of HOA coefficients of the coded HOA representation, which is equal to the number of HOA coefficients to be reconstructed.

**UsesNfc** This element determines whether or not the HOA Near Field Compensation (NFC) has been applied to the coded signal.

**NfcReferenceDistance** This element determines the radius in meter that has been used for the HOA NFC (interpreted as float in IEEE 754 format in little-endian).

**HOADecoderConfig()** This element contains information to initialize the HOA spatial decoder.

**MinAmbHoaOrder** This element determines the minimum HOA order used for the coding of the ambient HOA representation by MinAmbHoaOrder = **MinAmbHoaOrder** - 1. The value -1 indicates that the number of decorrelated ambiance coefficients is equal to zero. Thus the HOA representation is transmitted without a residual ambiance HOA representation of lower order and all transport channels have a flexible channel type. The value 6 is used to extend the HOA order signaling.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

MinNumOfCoeffsForAmbHOA This element determines the minimum number of ambient HOA coefficients.

NumOfAdditionalCoders This element determines the number of additional transport channels used for coding the directional and/or additional HOA coefficients of the ambient component. These transport channels have a flexible ChannelType that is defined in Table 123.

**SingleLayer** This element indicates that the HOA signal is provided in a single layer according to Table 128.

**Table 128 — SingleLayer definition**

| Value | Meaning |
|---|---|
| 0 | /* reserved */ |
| 1 | HOA signal is provided in a single layer |

**MaxNoOfDirSigsForPrediction** This element determines the number of directional signals used for the prediction of the predominant sound components.

NoOfBitsPerScalefactor This element determines the number of bits for reading **PredGains[**n**]**.

**CodedSpatialInterpolationTime** This element determines the time of the spatio-temporal interpolation of the Vector-based directional signals as defined in Table 130.

**SpatialInterpolationMethod** This element determines the interpolation method applied during the spatio-temporal interpolation of the vector-based signals.

**CodedVVecLength** This element indicates the length of the transmitted data vector used to synthesize the vector-based signals.

**MaxGainCorrAmpExp** Gives the exponent of basis two of the maximum, accumulated amplification that has been used in the gain correction tool of the HOA spatial encoder for amplification of signals of the transport channel. This value is required to compute the number of bits for reading MaxGainCorrPrevAmpExp when the IndependencyFlag is set to true.

**HOAFrameLengthIndicator** Indicates the Frame Length $L$ (number of samples) of the HOA Spatial Decoding relative to the Core Coder Frame Length as defined in Table 129.

MaxNumAddActiveAmbCoeffs This element determines the maximum number of additional HOA channels that can be used additionally for the coding of the ambient HOA representation.

AmbAsignmBits This element determines the number of bits that are required for reading **CodedAmbCoeffIdx[**i**]**.

ActivePredIdsBits This element determines the number of bits for reading **ActivePredIds**.

NumActivePredIdsBits This element determines the number of bits for reading **NumActivePredIds**.

GainCorrPrevAmpExtBits This element determines the number of bits for reading MaxGainCorrPrevAmpExp.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

### 12.3.2 Definitions of HOA payload

#### 12.3.2.1 HOAFrame()

The HOAFrame() holds the information that is required to decode the $L$ samples of a HOA frame of order HoaOrder.

| | |
|---|---|
| HOAFrame() | This block of data contains the data required to decode one frame of $L$ samples of the coded HOA representation. |
| NumOfDirSigs | This elements determines the number of active directional signals in the current HOAFrame(). |
| NumOfVecSigs | This element determines the number of active vector-based signals in the current HOAFrame(). |
| NumOfContAddHoaChans | This element determines the number of additional ambient HOA channels the current HOAFrame(). |
| **hoaIndependencyFlag** | This flag signals that the current frame is an independent frame that can be decoded without having knowledge about the previous frame. Otherwise this flag is equal to the usacIndependencyFlag**.** |
| ChannelSideInfoData(i) | This payload holds the side information for the i-th of the NumOfAdditionalCoders channels of flexible ChannelType. |
| HOAGainCorrectionData(i) | This payload contains data for the inverse gain correction of channel i. |
| DirSigChannelIds[NumOfDirSigs] | This element stores the channel index of each active directional signal of the current frame. |
| VecSigChannelIds[NumOfVecSigs] | This element stores the channel index of each active vector-based signal of the current frame. |
| ContAddHoaCoeff [NumOfContAddHoaChans] | This element stores the HOA coefficient number of each additional ambient HOA channel for which the AmbCoeffTransitionState word is of value 0 in the current frame. |
| HOAPredictionInfo( DirSigChannelIds, NumOfDirSigs ) | This payload contains data for the prediction of dominant sound sources from the active directional signals of the current frame. |

#### 12.3.2.2 ChannelSideInfoData(i)

This payload holds the side information for the i-th channel. The size and the data of the payload depend on the type of the channel.

| | |
|---|---|
| **ChannelType[i]** | This element stores the type of the i-th channel which is defined in Table 123. |
| **ActiveDirsIds[i]** | This element indicates the direction of the active directional signal using an index of the 900 predefined, uniformly distributed points from Annex F.9. The code word 0 is used for signaling the end of a directional signal. |
| **PFlag[i]** | The prediction flag used for the Huffman decoding of the scalar quantized V-vector associated with the Vector-based signal of the i-th channel. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**CbFlag[i]** The codebook flag used for the Huffman decoding of the scalar quantized V-vector associated with the Vector-based signal of the i-th channel.

**CodebkIdx[i]** Signals the specific codebook used to dequantize the vector-quantized V-vector associated with the Vector-based signal of the i-th channel.

**NbitsQ[i]** The NbitsQ[i] value determines the decoding method of the V-Vector associated with the Vector-based signal of the i-th channel. An NbitsQ[i] value of 4 determines the decoding of a vector-quantized V-vector. The value 5 determines the decoding of a uniform 8bit scalar quantized V-vector. If the value is greater than 5, Huffman decoding of the V-Vector is determined by using the Huffman table index as signaled with the NbitsQ[i] value (the Huffman tables are provided in Annex F.13 to F.22).

The two MSBs 00 signal the reuse of the values NbitsQ[i], PFlag[i], CbFlag[i], and CodebkIdx[i] from the previous frame (k-1).

**bA, bB** The msb (bA) and second msb (bB) of the NbitsQ[i] field.

**uintC** The code word of the remaining two bits of the NbitsQ[i] field.

AddAmbHoaInfoChannel(i) This payload holds the information for additional ambient HOA coefficients.

#### 12.3.2.3 AddAmbHoaInfoChannel(i)

This payload contains the information required to add an additional ambient HOA coefficient to the reconstructed HOA representation.

AmbCoeffTransitionState This decoder-internal variable tracks the state of the life-cycle of an additional ambient HOA coefficient. Those states are Fade-in, Continous State, and Fade-out. The AmbCoeffIdxTransition signals a change of the state in the bitstream. When an additional ambient HOA coefficient is faded in, the CodedAmbCoeffIdx word is sent to signal the new AmbCoeffIdx. In all other states, the AmbCoeffIdx of the previous frame is used.

**AmbCoeffIdxTransition** This element indicates that in this frame an additional ambient HOA coefficient is either being faded in or faded out. This flag will update the decoder-internal AmbCoeffTransitionState variable for this transport channel accordingly (see Table 6).

**CodedAmbCoeffIdx** This element reads the coded index of the additional ambient HOA coefficient.

AmbCoeffIdx[i] This element determines the index of the HOA signal where channel i contributes to the reconstructed HOA representation.

#### 12.3.2.4 HOAGainCorrectionData(i)

This structure comprises the parameters for the inverse gain correction of the spatial HOA decoding tool for the i-th transport channel.

HOAGainCorrectionData() This payload comprises the parameters for the inverse gain correction.

GainCorrPrevAmpExp[i] This element gives the amplification as an exponent to the basis of two that has been applied to the signal of the transport channel i in the previous frame. It is only required to send if the IndependencyFlag is set to true for starting

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

decoding at the current frame. The value range of the element is from –ceil(log2(1.5*NumOfHoaCoeffs) to MaxGainCorrAmpExp, where the lower bound is the maximal amplitude of any transport channel signal. This bound is computed from the assumption that all spatial domain signals of the encoded HOA representation have absolute amplitudes less than one. The spatial domain signals are computed by the multiplication of the HOA representation with the inverse of the modematrix created from the uniformly distributed positions from F.2 to F.11 corresponding to the **HoaOrder**.

**CodedGainCorrectionExp[i][n]** The index n addresses the bits of the run length code to determine the exponent used for the inverse gain correction of the i-th channels.

**GainCorrectionException[i]** indicates the gain correction exception state for each of the i-th channels.

#### 12.3.2.5 VVectorData( VecSigChannelIds(i) )

This structure contains the coded V-vector data used for the vector-based signal synthesis. The subfunctions used for the decoding are specified in 12.4.1.10.

| | |
|---|---|
| VVec(k)[i] | This is the V-vector for the k-th HOAframe() for the i-th channel. |
| VVecLength | This variable indicates the number of vector elements to read out. |
| VVecCoeffId | This vector contains the indices of the transmitted V-vector coefficients. |
| **VecVal** | An integer value between 0 and 255. |
| **aVal** | A temporary variable used during decoding of the VVectorData. |
| **huffVal** | A Huffman code word, to be Huffman-decoded. |
| **sgnVal** | This is the coded sign value used during decoding. |
| **intAddVal** | This is additional integer value used during decoding. |
| NumVvecIndices | The number of vectors used to dequantize a vector-quantized V-vector. |
| **WeightIdx** | The index in WeightValCdbk used to dequantize a vector-quantized V-vector. |
| nBitsW | Field size for reading **WeightIdx** to decode a vector-quantized V-vector. |
| WeightValCdbk | Codebook which contains a vector of positive real-valued weighting coefficients. Only necessary if NumVvecIndices is > 1. The WeightValCdbk with 256 entries is provided in Annex F.12. |
| **VvecIdx** | An index for VecDict, used to dequantize a vector-quantized V-vector. |
| nbitsIdx | Field size for reading **VvecIdx** to decode a vector-quantized V-vector. |
| WeightVal | A real-valued weighting coefficient to decode a vector-quantized V-vector. |

#### 12.3.2.6 HOAPredictionInfo( DirSigChannelIds, NumOfDirSigs )

The data of this payload provides information for the Predominant Sound Synthesis in the HOA Spatial Decoding tool. It includes parameters for the prediction of the predominant sound component from the active directional signals

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| PredIdsBits | This helper variable holds the number of bits for reading **PredDirSigIds[]**. The number of bits is adapted to the currently active number of directional signals NumOfDirSigs, which is determined in HOAFrame(). |
| **PSPredictionActive** | This element determines whether or not the spatial prediction tool contributes to the decoded HOA representation. |
| **KindOfCodedPredIds** | This element indicated the method for reading **ActivePred**[]. |
| **NumActivePredIds** | This element indicates the number of active prediction indices from the NumOfHoaCoeffs uniformly distributed position indices from F.2 to F.11. The table is selected according to HOAOrder. |
| NumActivePred | This element determines the number of active prediction indices. |
| **PredIds[idx]** | This array indicates the active prediction indices from F.2 to F.11. The table is selected according to HOAOrder. |
| **ActivePred[idx]** | This element indicates whether or not (‘1’ or ‘0’) the prediction is computed for each of the NumOfHoaCoeffs uniformly distributed positions from F.2 to F.11. The table is selected according to HOAOrder. |
| **PredDirSigIds[n]** | This array indicates the index of the transport channel of the active directional signals for each active prediction. |
| NumOfGains | The variable indicates the number of prediction gains provided by the bit stream. Only prediction gains that are greater than zero are provided. The maximal number of prediction gains is equal to NumActivePred times MaxNoOfDirSigsForPrediction. |
| **PredGains[n]** | This array holds the NumOfGains prediction gains. |

## 12.4 HOA Tool Description

### 12.4.1 HOA Frame Converter

#### 12.4.1.1 General

The HOA Frame Converter, shown in Figure 34, converts the parameters from the HOAFrame() payload and from the HOAAUHeader() payload to the parameters required for the HOA decoding tools.

#### 12.4.1.2 Global Parameter

The HOA Frame Converter defines the following variables:

| | |
|---|---|
| $I$ =NumHOATransportChannels | Number of transport signals used as input for the Spatial Decoding Tool |
| $i \in \{1, \ldots, I\}$ | Index for the transport signals |
| $N$ = HoaOrder | Ambisonics order of the coded HOA signal |
| $O = (N+1)^2$ | Number of HOA coefficients |
| $N_{\mathrm{MIN}}$ = **MinAmbHoaOrder** | Minimum order of the transmitted ambient HOA representation |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $O_{MIN} = (N_{MIN}+1)^2$ | Minimum number of ambient HOA coefficients |
| $d$ | Direction index |
| $D_{PRED}$ = MaxNoOfDirSigsForPrediction | Maximum number of directional signals used for the prediction of dominant sound sources |
| $D$ = NumOfAdditionalCoders | Number of channels with flexible channel types |
| $B_{SC}$ =NoOfBitsPerScaleFactor | Number of bits per scale factor |
| $Q = 900$ | Number of direction indices |
| $q \in \{1, \ldots, Q\}$ | Quantization index of a direction |
| $L$ | Frame length in samples for the HOA spatial decoding. The frame length $L$ depends on outputFrameLength, which is the number of output samples of the core decoder for decoding one frame (it can be 768, 1024, 2048 or 4096 output samples long). The Output frame length can be found in Table 70 in 6.1.1 of ISO/IEC 23003-3:2012. Furthermore, $L$ depends on the value of **HOAFrameLengthIndicator**. The definition of $L$ can be found in Table 129. |

**Table 129 — Value of frame length $L$ depending on HOAFrameLengthIndicator and outputFrameLength**

| **outputFrameLength** | **HOAFrameLengthIndicator** | | | |
|---|---|---|---|---|
| | **$00_2$** | **$01_2$** | **$10_2$** | **$11_2$** |
| **768** | 768 | 768 | 768 | reserved |
| **1024** | 1024 | 1024 | 1024 | |
| **2048** | 2048 | 1024 | 1024 | |
| **4096** | 4096 | 2048 | 1024 | |

| | |
|---|---|
| $l$ | Sample index |
| $k$ | Frame index |
| $r_{max}$ | Determines the maximum speaker distance of the listening setup. |
| $E$ = **CodedVVecLength** | Coded V-vector length |
| VecDict[cdbLen] | Codebook with *cdbLen* codebook entries containing vectors of HOA expansion coefficients, used to decode a vector-quantized V-vector. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

#### 12.4.1.3 Helper functions

```
bool isMemberOf(val, array, arraySize)
{
    idx = 0;
    for( idx = 0; idx < arraySize; idx++ )
    {
        if(array[idx] == val)
            return true;
    }
    return false;
}
```

#### 12.4.1.4 Prediction Parameters $\zeta(k)$

The set of predictions parameters consists of three arrays $\zeta(k) = \{p_{\mathrm{TYPE}}(k), P_{\mathrm{IND}}(k), P_{\mathrm{Q,F}}(k)\}$. The array $p_{\mathrm{TYPE}}(k)$ consists of $O$ entries, $P_{\mathrm{IND}}(k)$ and $P_{\mathrm{Q,F}}(k)$ are $D_{\mathrm{PRED}}$x$O$ matrices.

```
for (q=1; q ≤ O; q++) {
    p_TYPE(k)[q] = 0;
    for (d=1; d ≤ D_PRED; d++) {
        P_IND(k)[d][q] = 0;
        P_Q,F(k)[d][q] = 0;
    }
}
if (PSPredictionActive) {
    if (!KindOfCodedPredIds){
        i=0;
        for (q=0; q < O; q++) {
            if (ActivePred[q]) {
                p_TYPE(k)[q+1] = 1;
                i++;
            }
        }
    }
    else {
        for (i=0; i < NumActivePred; i++) {
            p_TYPE(k)[PredIds[i]] = 1;
        }
    }
}

i=0;
j=0;
for (q=1; q ≤ O; q++) {
    for (d=1; d ≤ D_PRED; d++) {
        if (p_TYPE(k)[i]>0) {
            P_IND(k)[d][q] = PredDirSigIds[i];
            i++;
            if (PredDirSigIds[i] != 0) {
                P_Q,F(k)[d][q] = PredGains[j];
                j++;
            }
        }
    }
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 12.4.1.5 Assigment vector $v_{\mathrm{AMB,ASSIGN}}(k)$

```
for (i=0; i < NumOfAdditionalCoders; i++) {
    if (ChannelType[i]==2) {
        v_AMB,ASSIGN(k)[i+1] = AmbCoeffIdx[i];
    }
    else {
        v_AMB,ASSIGN(k)[i+1]=0;
    }
}
nIdx = 1;
for (i= NumOfAdditionalCoders; i < NumHOATransportChannels; i++) {
    v_AMB,ASSIGN(k)[i+1]=nIdx;
    nIdx++;
}
```

#### 12.4.1.6 Tuple set $\mathcal{M}_{\mathrm{DIR}}(k)$

```
M_DIR(k)=∅;
for (i=0; i < NumOfAdditionalCoders; i++) {
    if (ChannelType[i]==0){
        M_DIR(k)=M_DIR(k) ∪ (i + 1, ActiveDirsIds[i]);
    }
}
```

#### 12.4.1.7 The sets $\mathcal{I}_{\mathrm{E}}(k)$, $\mathcal{I}_{\mathrm{D}}(k)$ and $\mathcal{I}_{\mathrm{U}}(k)$

```
I_E(k) = ∅;
I_D(k) = ∅;
I_U(k) = {1, ... O_MIN};
for (i=0; i < NumOfAdditionalCoders; i++) {
    switch(AmbCoeffTransitionState[i]) {
        case 0:
        {
                    I_U(k)=I_U(k) ∪ {AmbCoeffIdx[i]};
                    break;
        }
        case 1:
        {
                    I_E(k)=I_E(k) ∪ {AmbCoeffIdx[i]};
                    break;
        }
        case 2:
        {
                    I_D(k)=I_D(k) ∪ {AmbCoeffIdx[i]};
        }

    }
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

#### 12.4.1.8 Gain Correction Exponents $e_i(k)$

```
RunIdx = 0;
for (i = 0; i < I; i++) {
    if (IndependencyFlag) {
        g_IGC,i(k − 1) = pow( 2, GainCorrPrevAmpExp[i] );
       }
    TmpCodedExp = 0;
    FoundOne = false;

    while (!FoundOne) {
        FoundOne = CodedGainCorrectionExp[i][RunIdx];
        TmpCodedExp++;
        RunIdx++;
    }

    switch (TmpCodedExp) {
        case 1:
        {
            e_i+1(k) = 0;
            break;
        }
        case 2:
        {
            e_i+1(k) = -1;
            break;
        }
        default:
        {
            e_i+1(k) = TmpCodedExp-2;
        }
    }
}
```

#### 12.4.1.9 Gain Correction Exception Flag $\beta_i(k)$

```
for (i=0; i < I; i++) {
    β_i+1(k) = GainCorrectionException[i];
}
```

#### 12.4.1.10 Decoding of V-vector

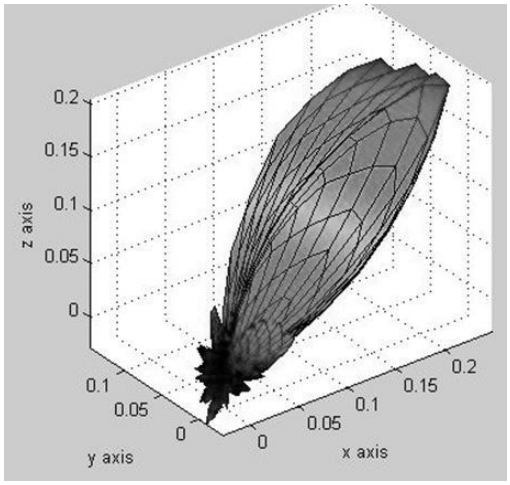
12.4.1.10.1 **General**

A V-vector represents the spatial distribution of the soundfield for a particular vector-based predominant sound and is decomposed from the HOA frame using a vector-based synthesis approach. Various examples of the V-vectors are illustrated in the following Figure 37.

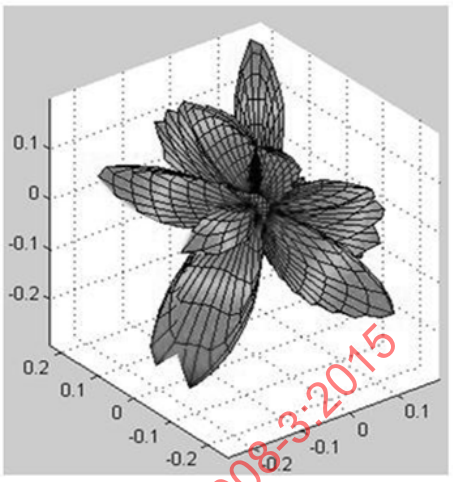
IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

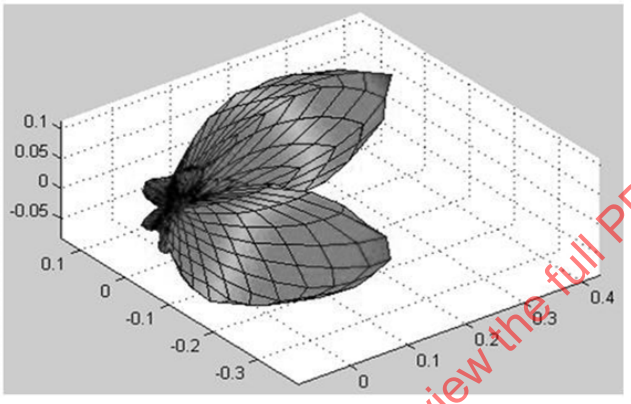
Bee sound comes from azimuth=0° and elevation=45°.



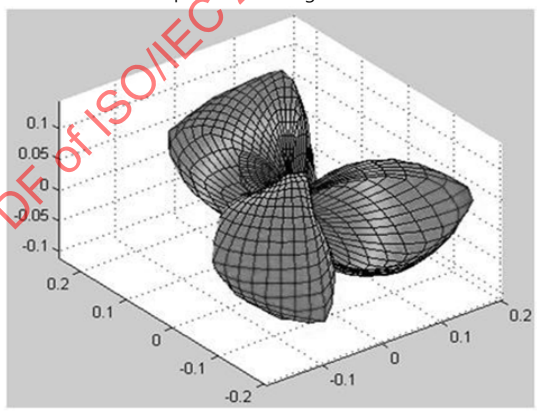

Helicopter sound comes from the sky.



Modern electronic music comes from two different directions.



People are shouting in a stadium.



IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Figure 37 — Illustrations of V-vectors**

These vectors were either scalar-quantized or vector-quantized. If the vectors are scalar quantized, an optional prediction from the V-vector of frame k-1 as well as efficient Huffman can be employed as shown in the following Figure 38.

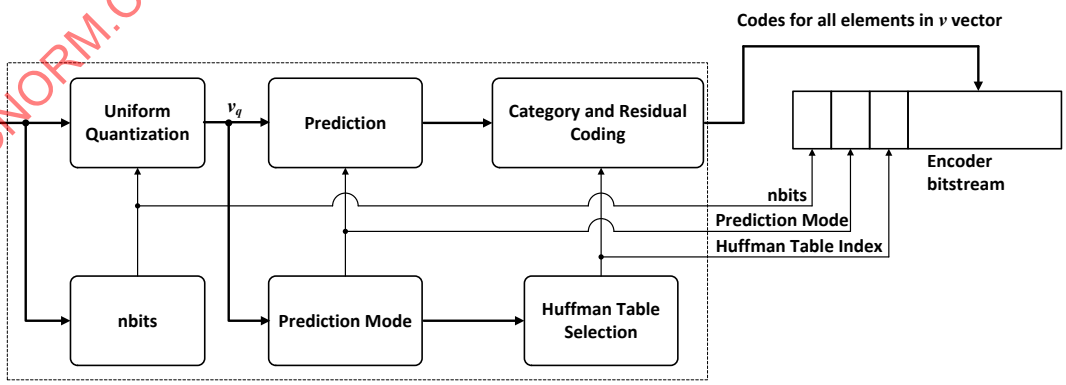



**Figure 38 — Coding of V-vector**

To decode a scalar-quantized V-vector, the nbits, prediction mode and Huffman Table index are obtained from the bitstream as specified below in 12.4.1.10.2, 12.4.1.10.3and 12.4.1.10.4. The bitstream is formulated to specify the compressed version of this spatial component (V-vector) along with the *nbits*, *prediction mode* and

© ISO/IEC 2015 – All rights reserved

*Huffman table index*. The compressed V-vector is Huffman decoded, predicted and dequantized to obtain a reconstructed V-vector, where the V-vectors for a $k^{th}$ frame are denoted as $v(k)$ as part of the tuple set $\mathcal{M}_{\mathrm{VEC}}(k)$. In the text below, *nbits* denotes the quantization step size. Portions of *nbits* are denoted as *unitC*, *bB* and *bA* below. The prediction mode is denoted as *PFlag* and the Huffman table index is denoted as *huffIdx*. The Huffman table index is not explicitly signaled in the bitstream but is derived as specified in 12.4.1.10.4 below based on a *CbFlag* specified in the bitstream.

When decoding vector-quantized V-vectors, a V-vector is represented by a weighted summation of pre-defined vectors as given by

$$V \approx \sum_{i=1}^{I} \omega_i \Omega_i$$

Where $\omega_i$ and $\Omega_i$ are an i-th weighting value and the corresponding vectors, respectively. Once the V-vector is reconstructed, a rescaling of the V-vector is performed, so that the Euclidian norm is of size N+1.

An example of the vector-quantized V-vector decoding is illustrated in Figure 39. As shown in Figure 39 (a), an original V-vector is represented by a mixture of multiple vectors. Thus, the original V-vector is estimated by a weighted sum as shown in Figure 39 (b) where a weighting vector is shown in Figure 39 (e). Figure 39 (c) and (f) illustrate the cases that only $I_S$ ($I_S \leq I$) highest weighting values are selected. Vector quantization is performed for the selected weighting values (Figure 39 (g)). The codebook indices and weights are signalled to the decoder, so that the V-vector as illustrated in Figure 39 (d) can be constructed.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

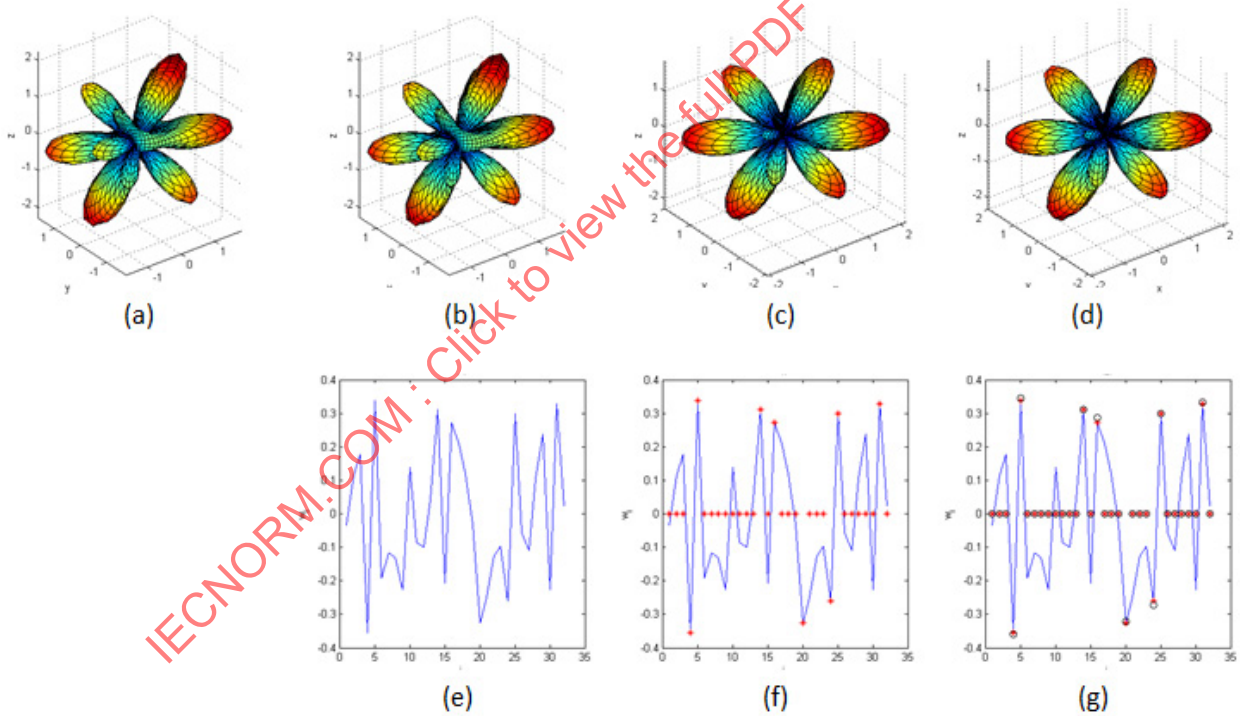



**Figure 39 — Example for vector-quantization of V-vector**

The size of the Vector codebook depends on the value NumVvecIndices and on the HOA order. If the variable NumVvecIndices is set to 1, the vector codebook containing HOA expansion coefficients derived from Annex F is used. If NumVvecIndices is larger than 1, the Vector codebook with $O$ vector is used in combination with 256x8 weighting values (Table in Annex F.12). For the HOA order 4, the Vector codebook with 32 entries as derived from the Table in Annex F.6 is used.


© ISO/IEC 2015 – All rights reserved

12.4.1.10.2 **VVecLength and VVecCoeffId**

The codedVVecLength word indicates:

0) Complete vector length (`NumOfHoaCoeffs` elements). Indicates that all of the coefficients for the predominant vectors (`NumOfHoaCoeffs`) are specified.

1) Vector elements 1 to `MinNumOfCoeffsForAmbHOA` and all elements defined in `ContAddHoaCoeff` are not transmitted. Indicates that only those coefficients of the predominant vector corresponding to the number greater than a MinNumOfCoeffsForAmbHOA are specified. Further those NumOfContAddAmbHoaChan coefficients identified in ContAddAmbHoaChan are subtracted. The list ContAddAmbHoaChan specifies additional channels corresponding to an order that exceeds the order MinAmbHoaOrder.

2) Vector elements 1 to `MinNumOfCoeffsForAmbHOA` are not transmitted. Indicates that those coefficients of the predominant vectors corresponding to the number greater than a MinNumOfCoeffsForAmbHOA are specified.

Both the VVecLength as well as the VVecCoeffId list are valid for all VVectors within one HOAFrame.

For the assignment algorithm below a helper function is defined as follows:

```
switch CodedVVecLength{
    case 0:
        VVecLength = NumOfHoaCoeffs;
        for (m=0; m<VVecLength; ++m) {
            VVecCoeffId[m] = m;
        }
        break;
    case 1:
        VVecLength = NumOfHoaCoeffs – MinNumOfCoeffsForAmbHOA - NumOfContAddHoaChans;
        CoeffIdx = MinNumOfCoeffsForAmbHOA+1;
        for (m=0; m<VVecLength; ++m) {
            bIsInArray = isMemberOf(CoeffIdx, ContAddHoaCoeff, NumOfContAddHoaChans);
            while (bIsInArray) {
                CoeffIdx++;
                bIsInArray = isMemberOf(CoeffIdx, ContAddHoaCoeff, NumOfContAddHoaChans);
            }
            VVecCoeffId[m] = CoeffIdx-1;
        }
        break;
    case 2:
        VVecLength = NumOfHoaCoeffs – MinNumOfCoeffsForAmbHOA;
        for (m=0; m< VVecLength; ++m) {
            VVecCoeffId[m] = m + MinNumOfCoeffsForAmbHOA;
        }
}
```

The first switch statement with the three cases (cases 0-2) thus provides a way by which to determine the predominant vector length in terms of the number (VVecLength) and indices of coefficients (VVecCoeffId).

12.4.1.10.3 **huffSelect `huffIdx`**

```
huffIdx = 5;
if (CbFlag(k)[i] == 1) {
    huffIdx = min(3, max(1, ceil(sqrt(VVecCoeffId[m]+1) - 1)));
}
else if (PFlag(k)[i] == 1) {
    huffIdx = 4;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

```
}
```

#### 12.4.1.10.4 **huffDecode cid**

```
thisTable = huffmanTable[NbitsQ[i]].codebook[huffIdx];
huffWordFound = 0;
word = 0;
while (huffWordFound == 0) {
    word = concatenate( word, hufVal);
    for ( l = 0 ; l < thisTable.length ; l++ ) {
        if ( word == thisTable [l] ) {
            huffWordFound = 1;
            cid = word->symbol;
        }
    }
}
```

#### 12.4.1.10.5 **Conversion to VVec element**

The kind of dequantization of the V-vector is signalled by the word NbitsQ. The NbitsQ value of 4 indicates vector-quantization. When NbitsQ equals 5, a uniform 8 bit scalar dequantization is performed. In contrast, an NbitsQ value of greater or equal to 6 indicates the application of Huffman decoding of a scalar-quantized V-vector. The prediction mode is denoted as the PFlag, while the CbFlag represents a Huffman Table information bit.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
if (NbitsQ(k)[i] == 4) {
    if (NumVvecIndices == 1) {
        for (m=0; m< VVecLength; ++m) {
            idx = VVecCoeffID[m];
            v^(i)_VVecCoeffId[m](k) = WeightVal[0] * VecDict[900].[VvecIdx[0]][idx];
        }
    } else {
        cdbLen = O;
        if (N==4) {
            cdbLen = 32;
        }
        for (m=0; m<O; ++m) {
            TmpVVec[m] = 0;
            for (j=0; j< NumVvecIndices; ++j) {
                TmpVVec[m] += WeightVal[j] * VecDict[cdbLen].[VvecIdx[j]][m];
            }
        }
        FNorm = 0.0;
        for (m=0; m<O; ++m) {
            FNorm += TmpVVec[m] * TmpVVec[m];
        }
        FNorm = (N+1)/sqrt(FNorm);
        for (m=0; m< VVecLength; ++m) {
               idx = VVecCoeffID[m];
               v^(i)_VVecCoeffId[m](k)= TmpVVec[idx] * FNorm;
        }
    }
}
elseif (NbitsQ(k)[i] == 5) {
    for (m=0; m<VVecLength; ++m) {
         v^(i)_VVecCoeffId[m](k) =    (N+1)*aVal[i][m];
    }
}
elseif (NbitsQ(k)[i] >= 6) {
    for (m=0; m<VVecLength; ++m) {
        v^(i)_VVecCoeffId[m](k) = (N+1) * (2^(16 - NbitsQ(k)[i])*aVal[i][m])/2^15;
        if (PFlag(k)[i] == 1) {
            v^(i)_VVecCoeffId[m](k) += v^(i)_VVecCoeffId[m](k − 1);
        }
    }
}
```

12.4.1.10.6 **Tuple set $\mathcal{M}_{\mathrm{VEC}}(k)$**

```
M_VEC(k)=∅;
for (i=0; i < NumOfAdditionalCoders; i++) {
    if (ChannelType[i]==1) {
        M_VEC(k)=M_VEC(k) ∪ (i + 1, v^(i)(k));
    }
}
```

### 12.4.2 Spatial HOA decoding

#### 12.4.2.1 General Architecture

The architecture of the spatial HOA decoder is depicted in Figure 40.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

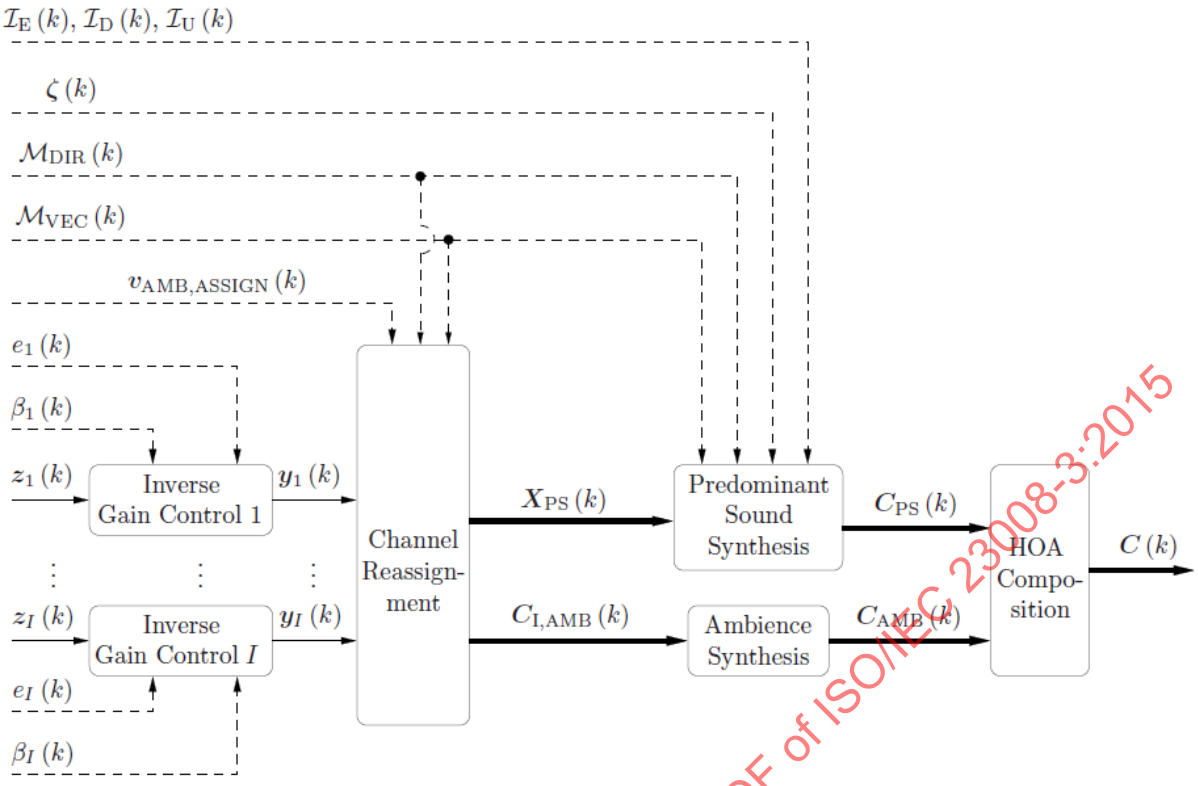



**Figure 40 — Architecture of spatial HOA decoder**

For each $i \in \{1, \ldots, I\}$, the $k$-th frame of the $i$-th perceptually decoded signal, i.e. $\boldsymbol{z}_i(k)$, is input to an Inverse Gain Control processing block together with the associated gain correction exponent $e_i(k)$ and gain correction exception flag $\beta_i(k)$. The $i$-th Inverse Gain Control processing provides a gain corrected signal frame $\boldsymbol{y}_i(k)$.

All of the $I$ gain corrected signal frames $\boldsymbol{y}_i(k)$, $i \in \{1, \ldots, I\}$, are passed together with the assignment vector $\boldsymbol{v}_{AMB,ASSIGN}(k)$ and the tuple sets $\mathcal{M}_{DIR}(k)$ and $\mathcal{M}_{VEC}(k)$ to the Channel Reassignment processing block, where they are redistributed to create the frame $\boldsymbol{X}_{PS}(k)$ of all predominant sound signals (i.e., all directional and vector-based signals) and the frame $\boldsymbol{C}_{I,AMB}(k)$ of an intermediate representation of the ambient HOA component.

The meaning of the input parameters to the Channel Reassignment is as follows. The assignment vector $\boldsymbol{v}_{AMB,ASSIGN}(k)$ indicates for each transport channel the index of a possibly contained coefficient sequence of the ambient HOA component. The tuple set

$$\mathcal{M}_{DIR}(k) := \left\{ \left( i, \Omega_{TEST,QUANT,i}(k) \right) \middle| \, i \text{ is index of an active direction for } (k+1)^{th} \text{ and } (k)^{th} \text{ frame} \right\}$$

consists of tuples of which the first element $i$ denotes the index of an active direction and of which the second element $\boldsymbol{\Omega}_{TEST,QUANT,i}(k)$ denotes the respective quantized direction defined in F.9. In other words, the first element of the tuple indicates the index $i$ of the gain corrected signal frame $\boldsymbol{y}_i(k)$ that is supposed to represent the directional signal related to the quantized direction $\boldsymbol{\Omega}_{TEST,QUANT,i}(k)$ given by the second element of the tuple. The quantized directions $\boldsymbol{\Omega}_{TEST,QUANT,i}(k)$ are assumed to have been determined at the spatial HOA encoding stage by an analysis of the input HOA representation. Such an analysis can be exemplarily based on the search of energetically dominant components of the sound scene using the directional power distribution (see Annex C.5.3 for details). For the reason, that directions are always computed with respect to two successive frames due to overlap add processing, there occurs the special case that for the last frame of the activity period for a directional signal there is actually no direction, which is signaled by setting the respective quantized direction to zero.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

The tuple set $\mathcal{M}_{\mathrm{VEC}}(k)$

$$\mathcal{M}_{\mathrm{VEC}}(k) := \left\{ \left( i, \boldsymbol{v}^{(i)}(k) \right) \middle| \; i \text{ is the index of a vector found for } (k+1)^{th} \text{ and } (k)^{th} \text{ frame} \right\}$$

consists of tuples of which the first element indicates the index $i$ of the gain corrected signal frame $\boldsymbol{y}_i(k)$ that represents the signal to be reconstructed by a vector $\boldsymbol{v}^{(i)}(k)$, which is given by the second element of the tuple. The vector $\boldsymbol{v}^{(i)}(k)$ is scalar-dequantized or vector-dequantized as specified in 12.4.1.10 and represents information about the spatial distributions (directions, widths, shapes) of the active signal $\boldsymbol{y}_i(k)$ in the reconstructed HOA frame $\boldsymbol{C}(k)$. It is assumed that $\boldsymbol{v}^{(i)}(k)$ has an Euclidean norm of $N+1$.

In the Predominant Sound Synthesis processing block the HOA representation of the predominant sound component $\boldsymbol{C}_{\mathrm{PS}}(k)$ is computed from the frame $\boldsymbol{X}_{\mathrm{PS}}(k)$ of all predominant sound signals. It uses the tuple sets $\mathcal{M}_{\mathrm{DIR}}(k)$ and $\mathcal{M}_{\mathrm{VEC}}(k)$, the set $\boldsymbol{\zeta}(k)$ of prediction parameters and the sets $\mathcal{I}_{\mathrm{E}}(k)$, $\mathcal{I}_{\mathrm{D}}(k)$, $\mathcal{I}_{\mathrm{U}}(k)$ of coefficient indices of the ambient HOA component, which have to be enabled, disabled and to remain active in the $(k)^{th}$ frame.

In the Ambience Synthesis processing block, the ambient HOA component frame $\boldsymbol{C}_{\mathrm{AMB}}(k)$ is created from the frame $\boldsymbol{C}_{\mathrm{I,AMB}}(k)$ of the intermediate representation of the ambient HOA component. Note that this processing also comprises an inverse spatial transform to invert the spatial transform applied in the encoder (see Annex C.5.3.3.2) for decorrelating the first $O_{\mathrm{MIN}}$ coefficients of the ambient HOA component.

Finally, in the HOA Composition processing block the ambient HOA component frame $\boldsymbol{C}_{\mathrm{AMB}}(k)$ and the frame $\boldsymbol{C}_{\mathrm{PS}}(k)$ of the predominant sound HOA component are superimposed to provide the decoded HOA frame $\boldsymbol{C}(k)$. In the following, the individual processing blocks are described in more detail.

#### 12.4.2.2 Inverse Gain Control

The goal of the Inverse Gain Control (IGC) processing block is to invert the gain modifications performed to the signals before perceptual encoding at the HOA encoding stage in order to recreate their initial value range.

For that purpose, a fixed template transition window function

$$\mathbf{f}_{\mathrm{IGC}} := [f_{\mathrm{IGC}}(1) \quad f_{\mathrm{IGC}}(2) \quad \ldots \quad f_{\mathrm{IGC}}(L)]$$

is employed,whose elements are defined by

$$f_{\mathrm{IGC}}(l) := \frac{1}{4}\cos\left(\frac{\pi(l-1)}{L-1}\right) + \frac{3}{4} \qquad \text{for} \quad l = 1, \ldots, L.$$

Assuming the input frame $\boldsymbol{z}_i(k)$ and output frame $\boldsymbol{y}_i(k)$ to be expressed through their samples by

$$\boldsymbol{z}_i(k) := [z_i(k,1) \quad z_i(k,2) \quad \ldots \quad z_i(k,L)]$$

$$\boldsymbol{y}_i(k) := [y_i(k,1) \quad y_i(k,2) \quad \ldots \quad y_i(k,L)],$$

the computation of the samples of the output frame is given by

$$y_i(k,l) = \begin{cases} \frac{z_i(k,l)}{g_{\mathrm{IGC},i}(k-1)} \cdot [f_{\mathrm{IGC}}(l)]^{-s_i(k)} & \text{if } \beta_i(k) = 0 \\ \frac{z_i(k,l)}{g_{\mathrm{IGC},i}(k-1)} \cdot 2^{s_i(k)} & \text{if } \beta_i(k) = 1 \end{cases} \qquad \text{for } i = 1, \ldots, I, l = 1, \ldots, L.$$

The factor $g_{\mathrm{IGC},i}(k-1)$ is initialized by

$$g_{\mathrm{IGC},i}(0) := 1 \qquad \text{for} \quad i = 1, \ldots, I,$$

and is recursively updated in the $k$-th frame by

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$g_{\mathrm{IGC},i}(k) = g_{\mathrm{IGC},i}(k-1) \cdot 2^{-e_i(k)} \quad \text{for } i = 1, \ldots, I.$$

For random access the factor $g_{\mathrm{IGC},i}(k-1)$ is provided explicitely in HOAGainCorrectionData() if the hoaIndependencyFlag has been signaled as true.

#### 12.4.2.3 Channel Reassignment

The Channel Reassignment processing block has the purpose to create the frame $\boldsymbol{X}_{\mathrm{PS}}(k)$ of all predominant sound signals and the frame $\boldsymbol{C}_{\mathrm{I,AMB}}(k)$ of an intermediate representation of the ambient HOA component from the gain corrected signal frames $\boldsymbol{y}_i(k)$, $i \in \{1, \ldots, I\}$, and the assignment vector $\boldsymbol{v}_{\mathrm{AMB,ASSIGN}}(k)$, which indicates for each transport channel the index of a possibly contained coefficient sequence of the ambient HOA component. Additionally, the sets $\mathcal{I}_{\mathrm{DIR,ACT}}(k)$ and $\mathcal{I}_{\mathrm{VEC,ACT}}(k)$ are used, which are supposed to contain the first elements of all tuples of $\mathcal{M}_{\mathrm{DIR}}(k)$ and $\mathcal{M}_{\mathrm{VEC}}(k)$, respectively. It is important to note that these two sets are assumed to be disjoint.

For the actual assignment the following steps are performed.

1. The frame $\boldsymbol{X}_{\mathrm{PS}}(k)$ of all predominant sound signals is assumed to be composed of the individual predominant sound signal frames according to

   $$\boldsymbol{X}_{\mathrm{PS}}(k) = \begin{bmatrix} x_{\mathrm{PS},1}(k) \\ x_{\mathrm{PS},2}(k) \\ \vdots \\ x_{\mathrm{PS},D}(k) \end{bmatrix},$$

   where each frame is supposed to consist of its samples according to

   $$x_{\mathrm{PS},i}(k) = [x_{\mathrm{PS},i}(k,1) \;\; x_{\mathrm{PS},i}(k,2) \ldots \; x_{\mathrm{PS},i}(k,L)] \quad \text{for } i = 1, \ldots, J,$$

   where $J = I - O_{\mathrm{MIN}}$.

   The sample values of the directional signal frames are computed as follows:

   $$x_{\mathrm{PS},i}(k,l) = \begin{cases} y_i(k,l) & \text{if } i \in \mathcal{I}_{\mathrm{DIR}}(k) \cup \mathcal{I}_{\mathrm{VEC}}(k) \\ 0 & \text{else} \end{cases} \quad \text{for } i = 1, \ldots, J, \quad l = 1, \ldots, L.$$

2. The frame $\boldsymbol{C}_{\mathrm{I,AMB}}(k)$ of an intermediate representation of the ambient HOA omponent is assumed to be composed according to

   $$\boldsymbol{C}_{\mathrm{I,AMB}}(k) = \begin{bmatrix} \boldsymbol{c}_{\mathrm{I,AMB},1}(k) \\ \boldsymbol{c}_{\mathrm{I,AMB},2}(k) \\ \vdots \\ \boldsymbol{c}_{\mathrm{I,AMB},O}(k) \end{bmatrix}$$

   with

   $$\boldsymbol{c}_{\mathrm{I,AMB},n}(k) = [c_{\mathrm{I,AMB},n}(k,1) \quad c_{\mathrm{I,AMB},n}(k,2) \quad \ldots \quad c_{\mathrm{I,AMB},n}(k,L)] \quad \text{for } n = 1, \ldots, O.$$

   The sample values of the intermediate representation of the ambient HOA component are obtained as follows:

$$c_{\mathrm{I,AMB},n}(k,l) = \begin{cases} y_i(k,l) & \text{if } \exists i \in \{1, \ldots, I\} \text{ such that } v_{\mathrm{AMB,ASSIGN},i}(k) = n \\ 0 & \text{else} \end{cases}.$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 12.4.2.4 Predominant Sound Synthesis

##### 12.4.2.4.1 General

The purpose of the Predominant Sound Synthesis is to create the frame $\boldsymbol{C}_{\mathrm{PS}}(k)$ of the HOA representation of the predominant sound component from the frame $\boldsymbol{X}_{\mathrm{PS}}(k)$ of all predominant sound signals using the tuple set $\mathcal{M}_{\mathrm{DIR}}(k)$ and $\mathcal{M}_{\mathrm{VEC}}(k)$, the set $\boldsymbol{\zeta}(k)$ of prediction parameters, and the sets $\mathcal{I}_{\mathrm{E}}(k)$, $\mathcal{I}_{\mathrm{D}}(k)$, and $\mathcal{I}_{\mathrm{U}}(k)$. As illustrated in Figure 41, the processing can be subdivided into four processing steps, which are described in the following.

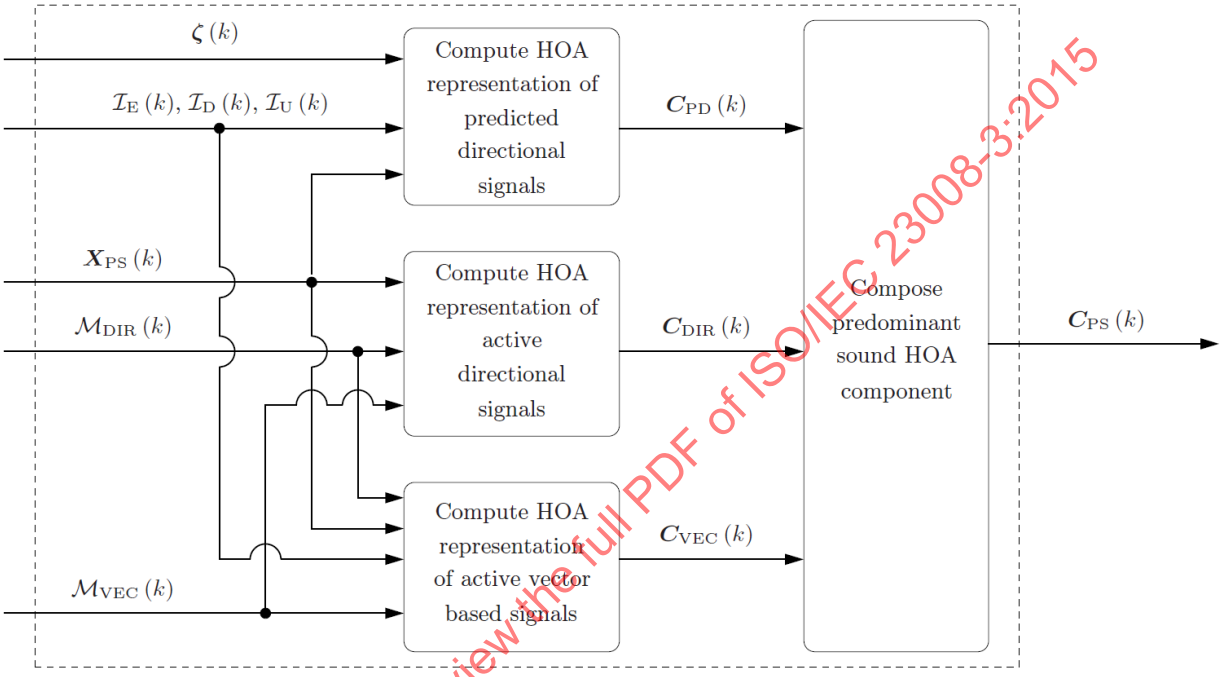



**Figure 41 — Predominant sound synthesis**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

##### 12.4.2.4.2 Compute HOA representation of active directional signals

In order to avoid artifacts due to changes of the directions between successive frames, the computation of the HOA representation from the directional signals is based on the concept of overlap add. Hence, the HOA representation $\boldsymbol{C}_{\mathrm{DIR}}(k)$ of active directional signals is computed as the sum of a faded out component and a faded in component:

$$\boldsymbol{C}_{\mathrm{DIR}}(k) = \boldsymbol{C}_{\mathrm{DIR,OUT}}(k) + \boldsymbol{C}_{\mathrm{DIR,IN}}(k).$$

To compute the two individual components, in a first step the instantaneous signal frames for directional signal indices $d \in \mathcal{I}_{\mathrm{DIR}}(k_1)$ and directional signal frame index $k_2$ are defined by

$$\boldsymbol{C}_{\mathrm{DIR,I}}^{(d)}(k_1; k_2) := \boldsymbol{\Psi}^{(N,29)}\big|_{\Omega_{\mathrm{TEST,QUANT},d}(k_1)} \boldsymbol{x}_{\mathrm{DIR},d}(k_2),$$

where $\boldsymbol{\Psi}^{(N,29)}$ denotes the mode matrix of order $N$ with respect to the predefined directions $\boldsymbol{\Omega}_n^{(29)}, n = 1, \ldots, Q = (29+1)^2 = 900$, is defined in Annex F.1.5 and $\boldsymbol{\Psi}^{(N,29)}|_q$ denotes the $q$-th column vector of $\boldsymbol{\Psi}^{(N,29)}$.

The sample values of the faded out and faded in directional HOA components are then determined by

© ISO/IEC 2015 – All rights reserved

$$c_{\mathrm{DIR,OUT},i}(k,l) = \sum_{d \in \mathcal{I}_{\mathrm{DIR,NZ}}(k-1)} c^{(d)}_{\mathrm{DIR,L},i}(k-1;k,l) \cdot \begin{cases} w_{\mathrm{DIR}}(L+l) & \text{if } d \in \mathcal{I}_{\mathrm{DIR}}(k) \\ w_{\mathrm{VEC}}(L+l) & \text{if } d \in \mathcal{I}_{\mathrm{VEC}}(k) \\ 1 & \text{else} \end{cases}$$

$$c_{\mathrm{DIR,IN},i}(k,l) = \sum_{d \in \mathcal{I}_{\mathrm{DIR,NZ}}(k)} c^{(d)}_{\mathrm{DIR,L},i}(k;k,l) \cdot \begin{cases} w_{\mathrm{DIR}}(l) & \text{if } d \in \mathcal{I}_{\mathrm{DIR}}(k-1) \cup \mathcal{I}_{\mathrm{VEC}}(k-1) \\ 1 & \text{else} \end{cases},$$

where $\mathcal{I}_{\mathrm{DIR,NZ}}(k)$ denotes the set of those first elements of $\mathcal{M}_{\mathrm{DIR}}(k)$ where the corresponding second element is non-zero. The fading of the instantaneous HOA representations for the overlap add operation is accomplished with two different fading windows

$$\boldsymbol{w}_{\mathrm{DIR}} := [w_{\mathrm{DIR}}(1) \quad w_{\mathrm{DIR}}(2) \quad \dots \quad w_{\mathrm{DIR}}(2L)]$$

$$\boldsymbol{w}_{\mathrm{VEC}} := [w_{\mathrm{VEC}}(1) \quad w_{\mathrm{VEC}}(2) \quad \dots \quad w_{\mathrm{VEC}}(2L)]$$

whose elements are defined by

$$w_{\mathrm{DIR}}(l) := \frac{1}{2}\left[1 - \cos\left(2\pi \frac{l-1}{2L}\right)\right]$$

$$w_{\mathrm{VEC}}(l) := \begin{cases} \dfrac{l-1}{L_{\mathrm{IP}}-1} & \text{if } 1 \le l \le L_{\mathrm{IP}} \wedge \mathrm{SpatialInterpolationMethod} = 0 \\ \dfrac{1}{2}\left[1 - \cos\left(2\pi \dfrac{l-1}{2L_{\mathrm{IP}}}\right)\right] & \text{if } 1 \le l \le L_{\mathrm{IP}} \wedge \mathrm{SpatialInterpolationMethod} = 1 \\ 1 & \text{if } L_{\mathrm{IP}}+1 \le l \le L \\ 1 - \dfrac{l-L-1}{L_{\mathrm{IP}}-1} & \text{if } L+1 \le l \le L+L_{\mathrm{IP}} \wedge \mathrm{SpatialInterpolationMethod} = 0 \\ 1 - \dfrac{1}{2}\left[1 - \cos\left(2\pi \dfrac{l-L-1}{2L_{\mathrm{IP}}}\right)\right] & \text{if } L+1 \le l \le L+L_{\mathrm{IP}} \wedge \mathrm{SpatialInterpolationMethod} = 1 \\ 0 & \text{if } L+L_{\mathrm{IP}}+1 \le l \le 2L \end{cases},$$

where $L_{\mathrm{IP}}$ is defined in Table 130.

**Table 130 — Decoding of codedSpatialInterpolationTime into $L_{\mathrm{IP}}$**

| $L$ | CodedSpatialInterpolationTime | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **768** | 0 | 32 | 64 | 128 | 256 | 384 | 512 | 768 |
| **1024** | 0 | 64 | 128 | 256 | 384 | 512 | 768 | 1024 |
| **2048** | 0 | 128 | 256 | 512 | 768 | 1024 | 1536 | 2048 |
| **4096** | 0 | 256 | 512 | 1024 | 1536 | 2048 | 3072 | 4096 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

12.4.2.4.3 **Compute HOA representation of predicted directional signals**

The parameter set $\boldsymbol{\zeta}(k) = \{\boldsymbol{p}_{\mathrm{TYPE}}(k), \boldsymbol{P}_{\mathrm{IND}}(k), \boldsymbol{P}_{\mathrm{Q,F}}(k)\}$ related to the spatial prediction of the directional signals consists of the following components:

— The vector $\boldsymbol{p}_{\mathrm{TYPE}}(k)$, whose elements $p_{\mathrm{TYPE},n}(k)$, $n = 1, \dots, O$ indicate if for the $n$-th direction $\boldsymbol{\Omega}_n^{(N)}$, defined in F.2 to F.11 and the $k$-th and $(k+1)$-th frame a prediction is performed or not, and if so, then they also indicate the kind of prediction. In particular, the meaning of the elements is as follows:


© ISO/IEC 2015 – All rights reserved

$$p_{\mathrm{TYPE},n}(k) = \begin{cases} 0 & \text{for no prediction for the direction } \Omega_n^{(N)} \text{ and } k-\text{th and } (k+1)-\text{th frame} \\ 1 & \text{for a full band prediction for the direction } \Omega_n^{(N)} \text{ and } k-\text{th and } (k+1)-\text{th frame} \end{cases}$$

— The matrix $\boldsymbol{P}_{\mathrm{IND}}(k)$, whose elements $p_{\mathrm{IND},d,n}(k)$, $d = 1, \ldots, D_{\mathrm{PRED}}$, $n = 1, \ldots, O$ denote the indices from which directional signals the prediction for the direction $\Omega_n^{(N)}$ and the $k$-th and $(k+1)$-th frame has to be performed. If no prediction is to be performed for a direction $\Omega_n^{(N)}$, the corresponding column of the matrix $\boldsymbol{P}_{\mathrm{IND}}(k)$ consists of zeros. Further, if less than $D_{\mathrm{PRED}}$ directional signals are used for the prediction for a direction $\Omega_n^{(N)}$, the non-required elements in the $n$-th column of $\boldsymbol{P}_{\mathrm{IND}}(k)$ are also zero.

— The matrix $\boldsymbol{P}_{\mathrm{Q,F}}(k)$, which contains the corresponding quantized prediction factors $p_{\mathrm{Q,F},d,n}(k)$, $d = 1, \ldots, D_{\mathrm{PRED}}$, $n = 1, \ldots, O$.

Note that the prediction parameters $\boldsymbol{\zeta}(k)$ are related to the frames $k$ and $(k+1)$. Additionally, the following dependent quantity

$$b_{\mathrm{ACT}}(k) = \begin{cases} 1 & \text{if } \exists n \text{ such that } p_{\mathrm{TYPE},n}(k) = 0 \\ 0 & \text{else} \end{cases}$$

is introduced, which indicates whether a prediction is to be performed related to frames $k$ and $k+1$. Further, the quantized prediction factors $p_{\mathrm{Q,F},d,n}(k)$, $d = 1, \ldots, D_{\mathrm{PRED}}$, $n = 1, \ldots, O$, are dequantized to provide the actual prediction factors

$$p_{\mathrm{F},d,n}(k) = \left(p_{\mathrm{Q,F},d,n}(k) + \frac{1}{2}\right) \cdot 2^{-B_{\mathrm{SC}}+1}.$$

The idea behind the computation of the predicted HOA component is to represent it by directional signals (i.e. general plane wave functions) impinging from the predefined directions $\Omega_n^{(N)}$, $n = 1, \ldots, O$, and then transform this representation to an HOA representation. These directional signals impinging from the predefined directions $\Omega_n^{(N)}$, $n = 1, \ldots, O$, are predicted from the predominant sound signals $\boldsymbol{X}_{\mathrm{PS}}(k)$ using the parameter set $\boldsymbol{\zeta}(k)$ The computation of the predicted directional signals is based on the concept of overlap add in order to avoid artifacts due to changes of the prediction parameters between successive frames. Hence, the $k$-th frame of the predicted directional signals, denoted by $\mathbf{X}_{\mathrm{PD}}(k)$, is computed as the sum of a faded out component and a faded in component:

$$\mathbf{X}_{\mathrm{PD}}(k) = \mathbf{X}_{\mathrm{PD,OUT}}(k) + \mathbf{X}_{\mathrm{PD,IN}}(k).$$

The sample values $x_{\mathrm{PD,OUT},n}(k,l)$ and $x_{\mathrm{PD,IN},n}(k,l)$, $n = 1, \ldots, O$, $l = 1, \ldots, L$, of the faded out and faded in predicted directional signals are then computed by

$$x_{\mathrm{PD,OUT},n}(k,l) = w_{\mathrm{DIR}}(L+l) \cdot \begin{cases} 0 & \text{if } p_{\mathrm{TYPE},n}(k-1) = 0 \\ \sum_{d=1}^{D_{\mathrm{PRED}}} p_{\mathrm{F},d,n}(k-1) \cdot x_{\mathrm{PS},p_{\mathrm{IND},d,n}(k-1)}(k,l) & \text{if } p_{\mathrm{TYPE},n}(k-1) = 1 \end{cases}$$

$$x_{\mathrm{PD,IN},n}(k,l) = w_{\mathrm{DIR}}(l) \cdot \begin{cases} 0 & \text{if } p_{\mathrm{TYPE},n}(k) = 0 \\ \sum_{d=1}^{D_{\mathrm{PRED}}} p_{\mathrm{F},d,n}(k) \cdot x_{\mathrm{PS},p_{\mathrm{IND},d,n}(k)}(k,l) & \text{if } p_{\mathrm{TYPE},n}(k) = 1 \end{cases}$$

In a next step, the predicted directional signals are transformed to the HOA domain by

$$\boldsymbol{C}_{\mathrm{PD,I}}(k) = \boldsymbol{\Psi}^{(N,N)} \cdot \boldsymbol{X}_{\mathrm{PD}}(k),$$

where $\boldsymbol{\Psi}^{(N,N)}$ denotes the mode matrix of order $N$ with respect to the predefined directions $\Omega_n^{(N)}, n = 1, \ldots, O$, defined in Annex F.1.5, Assuming the preliminary HOA representation $\boldsymbol{C}_{\mathrm{PD,I}}(k)$ of the predicted directional signals to be expressed by means of its samples by

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$\boldsymbol{C}_{\mathrm{PD,I}}(k) = \begin{bmatrix} c_{\mathrm{PD,I},1}(k,1) & \dots & c_{\mathrm{PD,I},1}(k,L) \\ \vdots & \ddots & \vdots \\ c_{\mathrm{PD,I},O}(k,1) & \dots & c_{\mathrm{PD,I},O}(k,L) \end{bmatrix},$$

the samples of the final output HOA representation

$$\boldsymbol{C}_{\mathrm{PD}}(k) = \begin{bmatrix} c_{\mathrm{PD},1}(k,1) & \dots & c_{\mathrm{PD},1}(k,L) \\ \vdots & \ddots & \vdots \\ c_{\mathrm{PD},O}(k,1) & \dots & c_{\mathrm{PD},O}(k,L) \end{bmatrix}$$

of the predicted directional signals are computed

$$c_{\mathrm{PD},n}(k,l) = \begin{cases} 0 & \text{if } n \in \mathcal{I}_{\mathrm{U}}(k) \\ c_{\mathrm{PD,I},n}(k,l) \cdot w_{\mathrm{DIR}}(l) & \text{if } n \in \mathcal{I}_{\mathrm{D}}(k) \wedge b_{\mathrm{ACT}}(k-1) = 1 \\ c_{\mathrm{PD,I},n}(k,l) \cdot w_{\mathrm{DIR}}(L+l) & \text{if } n \in \mathcal{I}_{\mathrm{E}}(k) \wedge b_{\mathrm{ACT}}(k) = 1 \\ c_{\mathrm{PD,I},n}(k,l) & \text{else} \end{cases} \quad \text{for } n = 1,\dots,O, l = 1,\dots,L$$

#### 12.4.2.4.4 **Compute HOA representation of active vector-based signals**

##### **12.4.2.4.4.1 General**

The computation of the HOA representation of the vector-based signals is conceptualized by the interpolation of V-vectors which describes the distribution of the predominant sound components in space (e.g., directions, shapes, and widths). Because a V-vector already contains all relevant spatial information about the predominant sound components, a spatial prediction, as specified in 12.4.2.4.3, is not used for vector-based signals. The interpolation of the V-vector is carried out over time. This spatio-temporal interpolation ensures a continuous and smooth evolution of the soundfield across frame boundaries, that lends itself to perceptually transparent quality. In this manner, the HOA signal is re-composed from the original decomposition comprised of the interpolated vectors and the corresponding predominant signals.

The following Figure 42 illustrates how the HOA coefficients are reformulated from the vector-based signals. Scalar dequantization is first performed with respect to each V-vector to generate $\mathcal{M}_{\mathrm{VEC}}(k)$, where the *i*th individual vectors of the current frame may be denoted as $\boldsymbol{v}_{\mathrm{I}}^{(i)}(k)$. The V-vectors are decomposed from the HOA coefficients typically employing a linear invertible transform (see Annex C.5.3.2.1). Spatio-temporal interpolation is performed with respect to the $\mathcal{M}_{\mathrm{VEC}}(k)$ and $\mathcal{M}_{\mathrm{VEC}}(k-1)$ (which denotes V-vectors from a previous frame with individual vectors of $\mathcal{M}_{\mathrm{VEC}}(k-1)$ denoted as $\boldsymbol{v}_{\mathrm{O}}^{(i)}(k)$). The interpolation method is controlled by $w_{\mathrm{VEC}}(l)$, as described in more detail below. Following interpolation, the *i*th interpolated V-vector ($\overline{\boldsymbol{v}^{(i)}(k,l)}$) are then mutliplied by the *i*th predominant signal, $X_{PS}^{(i)}(k,l)$, to produce the *i*th column of the HOA representation ($\boldsymbol{c}_{\mathrm{VEC}}^{(i)}(k,l)$). The column vectors are then summed to formulate the HOA representation of the vector-based signals. In this way, the HOA ceofficients are reconstructed for a frame.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

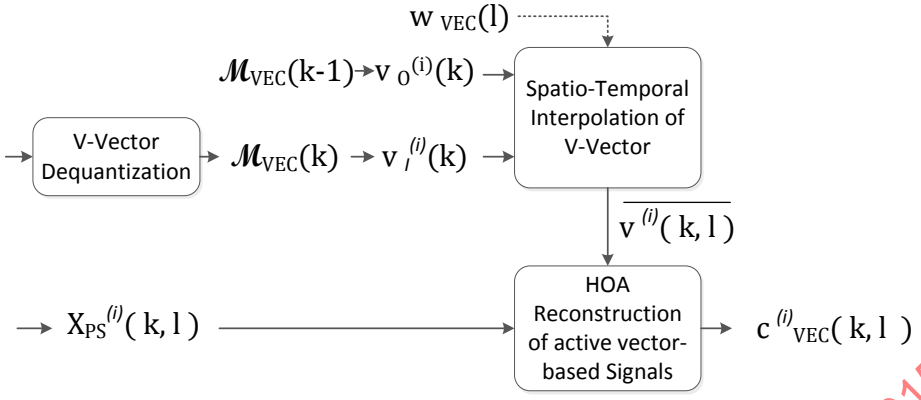



**Figure 42 — Computation of vector-based signals**

#### 12.4.2.4.4.2 Spatio-temporal interpolation of V-vectors

An interpolation matrix $\overline{\boldsymbol{V}^{(i)}(k)} = [\overline{\boldsymbol{v}^{(i)}(k,1)}, \ \overline{\boldsymbol{v}^{(i)}(k,2)}, \ \ldots \ \overline{\boldsymbol{v}^{(i)}(k,L)},]$ is first computed for each index $i \in \mathcal{I}_{\mathrm{VEC}}(k) \cup \mathcal{I}_{\mathrm{VEC}}(k-1)$ of a vector-based signal that is active in the $k$-th or $(k-1)$-th frame. Its columns $\overline{\boldsymbol{v}^{(i)}(k,l)}$ represent for each sample $1 \leq l \leq L$ of a frame an interpolated vector defined by

$$\overline{\boldsymbol{v}^{(i)}(k,l)} = w_{\mathrm{I}}^{(i)}(l)\,\boldsymbol{v}_{\mathrm{I}}^{(i)}(k) + \left(1 - w_{\mathrm{I}}^{(i)}(l)\right)\boldsymbol{v}_{0}^{(i)}(k).$$

In the equation above, $\boldsymbol{v}_{\mathrm{I}}^{(i)}(k)$ and $\boldsymbol{v}_{0}^{(i)}(k)$ denote the vectors between those the interpolation takes place, defined by

$$\boldsymbol{v}_{\mathrm{I}}^{(i)}(k) = \begin{cases} \boldsymbol{v}^{(i)}(k) & \text{if } i \in \mathcal{I}_{\mathrm{VEC}}(k) \\ \boldsymbol{0} & \text{else} \end{cases}$$

$$\boldsymbol{v}_{0}^{(i)}(k) = \begin{cases} \boldsymbol{v}^{(i)}(k-1) & \text{if } i \in \mathcal{I}_{\mathrm{VEC}}(k-1), \\ \boldsymbol{0} & \text{else} \end{cases}$$

where $\boldsymbol{0}$ denotes the zero vector. This is shown in the following Figure 43.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

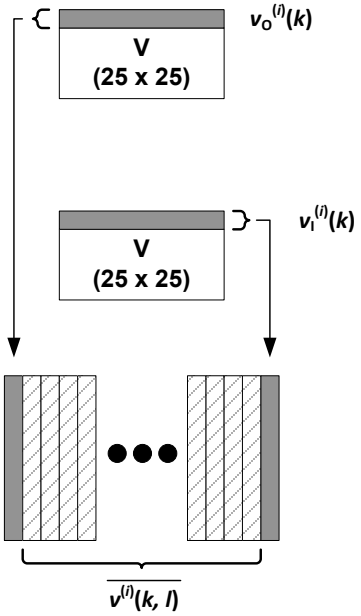



**Figure 43 — Interpolation of V-vectors**

Further, the interpolation function $w_I^{(i)}(l)$ is defined by

$$w_I^{(i)}(l) = \begin{cases} w_{DIR}(l) & \text{if } i \in \mathcal{I}_{VEC}(k-1) \cap \mathcal{I}_{DIR}(k) \\ w_{VEC}(l) & \text{else} \end{cases},$$

where

$$w_{VEC}(l) = \begin{cases} \dfrac{l-1}{L_{IP}-1} & \text{for } 1 \le l \le L_{IP} \text{ if } \textbf{SpatialInterpolationMethod} = 0 \\ \dfrac{1}{2}\left[1 - \cos\left(2\pi \dfrac{l-1}{2L_{IP}}\right)\right] & \text{for } 1 \le l \le L_{IP} \text{ if } \textbf{SpatialInterpolationMethod} = 1 \\ 1 & \text{for } L_{IP}+1 \le l \le L \end{cases},$$

and where, $L_{IP}$ is indicated by the variable CodedSpatialInterpolationTime as given by Table 130.

The HOA representation $\boldsymbol{C}_{VEC}^{(i)}(k) = \left[\boldsymbol{c}_{VEC}^{(i)}(k,1) \quad \boldsymbol{c}_{VEC}^{(i)}(k,2) \quad \ldots \quad \boldsymbol{c}_{VEC}^{(i)}(k,L)\right]$ for each $i^{th}$ vector-based signal $\boldsymbol{x}_{PS,i}(k), i \in \mathcal{I}_{VEC}(k) \cup \mathcal{I}_{VEC}(k-1)$, is a matrix of dimension $(N+1)^2 \times L$, whose columns are given by:

$$\boldsymbol{c}_{VEC}^{(i)}(k,l) = \overline{\boldsymbol{v}^{(i)}(k,l)}\, x_{PS,i}(k,l).$$

The complete HOA representation of the vector-based signals can be computed by summing the HOA contribution from each individual vector-based signal as follows:

$$\boldsymbol{C}_{VEC}(k) = \sum_{i \in \mathcal{I}_{VEC}(k) \cup \mathcal{I}_{VEC}(k-1)} \boldsymbol{C}_{VEC}^{(i)}(k).$$

In the case, that $E$, the CodedVVecLength, has a value of 1, the following operations have to be processed:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

— If there are coefficient sequences of the ambient HOA component that are explicitly additionally transmitted and faded in during the $k$-th frame (of which the indices are contained in the set $\mathcal{I}_{\mathrm{E}}(k)$), the respective coefficient sequences of the HOA representation $\tilde{\boldsymbol{c}}_{\mathrm{VEC}}(k)$ have to be faded out using the fade-out part of the window $w_{\mathrm{DIR}}$.

— If there are coefficient sequences of the ambient HOA component that are explicitly additionally transmitted and faded out during the $k$-th frame (of which the indices are contained in the set $\mathcal{I}_{\mathrm{D}}(k)$), the respective coefficient sequences of the HOA representation $\tilde{\boldsymbol{c}}_{\mathrm{VEC}}(k)$ have to be faded in using the fade-in part of the window $w_{\mathrm{DIR}}$.

— Hence, the final HOA representation of the vector-based signals is obtained by

$$c_{\mathrm{VEC},n}(k,L) = \begin{cases} \tilde{c}_{\mathrm{VEC},n}(k,L) \cdot w_{\mathrm{DIR}}(l) & \text{if } n \in \mathcal{I}_{\mathrm{D}}(k) \wedge E = 1 \\ \tilde{c}_{\mathrm{VEC},n}(k,L) \cdot w_{\mathrm{DIR}}(L+l) & \text{if } n \in \mathcal{I}_{\mathrm{E}}(k) \wedge E = 1 \\ \tilde{c}_{\mathrm{VEC},n}(k,L) & \text{else} \end{cases} \qquad \text{for } 1 \leq l \leq L\,.$$

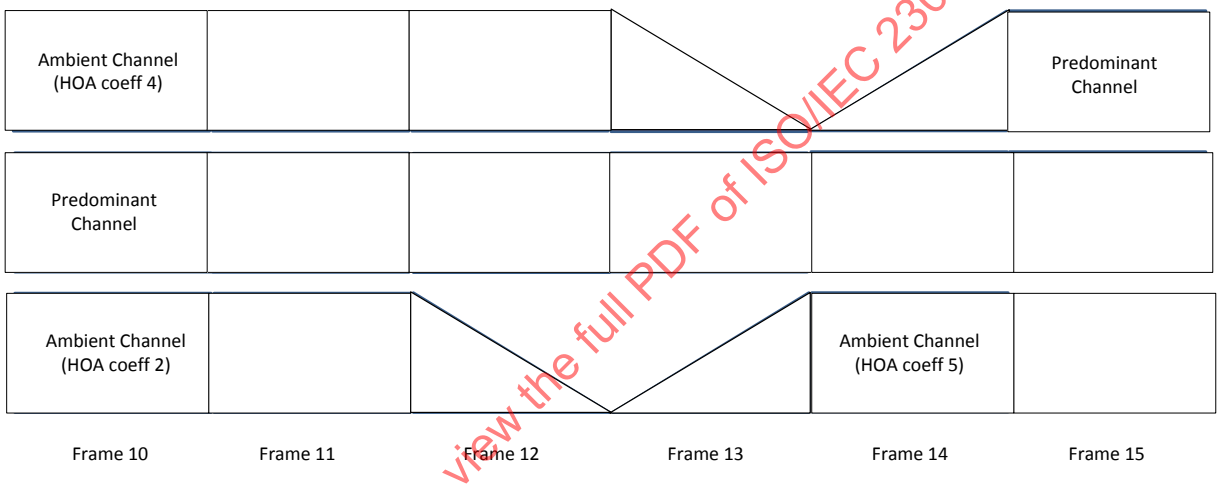



**Figure 44 — Example Fade-in and Fade-Out of Components**

The above Figure 44 exemplifies how the ambient HOA coefficient channels 4, 2, and 5 and a predominant-signal channel are faded-in and faded-out as described above. The ambient HOA coefficient channel 4 undergoes a period of transition during frame 13 with the predominant channel undergoing transition during frame 14 to replace the ambient HOA coefficient channel 4 in the bitstream. The second predominant channel remains unchanged through frames 10-15. The ambient HOA coefficient channel 2 undergoes a transition during frame 12 with the ambient HOA coefficient channel 5 undergoing a transition during frame 13 to replace the ambient HOA coefficient channel 2 in the bitstream.

During the above-described periods of transition, the AmbCoeffTransition flag specified in Table 124 indicates a transition of the respective channel. Given the previous state of the AmbCoeffTransitionMode, the respective channel either performs a fade-in or a fade-out operation.

The foregoing is represented by the following pseudo-code:

*1. Reconstructing newly introduced distinct components (if any)*

```
if exist(newTransportChannelPredSound) {
    for (l= 0; l<L; ++l) {
        c_VEC^(i)(k,l) = X_PS^(i)(k,l) * v_I^(i)(k)
    }
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

*2. Reconstructing continuous distinct components (if any) and apply spatio-temporal interpolation*

```
if exist(contTransportChannelPredSound) {
    for (l= 0; l<L; ++l) {
        c_VEC^(i)(k,l) += v^(i)(k,l) * X_PS^(i)(k,l);
    }
}
```

*3. If transitional ambient HOA coefficients are present in the frame, apply fade in/fade out to* $\boldsymbol{C}_{VEC}$

```
foreach n ∈ I_E(k) {
    for (l =0; l<L; ++l) {
        C̃_VEC,n(k,l) *= w_DIR(L+l);
    }
}

foreach n ∈ I_D(k) {
    for (l =0; l<L; ++l) {
        C̃_VEC,n(k,l) *= w_DIR(l);
    }
}
```

The foregoing pseudo-code describes how to process the reconstructed vector-based sound components $\tilde{\boldsymbol{C}}_{\mathrm{VEC}}$ when an ambient HOA coefficient is in transition in case $E$ has a value of 1.

The first section provides pseudo-code for reconstructing newly introduced predominant components when present. The second section provides pseudo-code for reconstructing predominant components that continuing from the previous to the current frame and applying spatio-temporal interpolation. In section three of the pseudo-code the HOA coefficient reconstructed from the vector-based sound component are faded-out or faded-in with regards to the additional HOA ambient channels that are arise or retire in frame *k*.

Identifying which of the ambient HOA coefficients that are in transition as described above enables the identification of those of the HOA coefficients that are included in or excluded from the bitstream. Various syntax elements, such as the CodedAmbCoeffIdx, MinNumOfCoeffsForAmbHOA, and AmbCoeffIdxTransition, are used to derive AmbCoeffIdx[i], which identifies those of the ambient HOA coefficients included in the bitstream. Based on these syntax elements, the ambient HOA coefficients are identified and parsed from the bitstream.

#### 12.4.2.4.5 Compose complete predominant sound component

The complete predominant sound HOA component, $\boldsymbol{C}_{\mathrm{PS}}(k)$, is obtained as the sum of the HOA component $\boldsymbol{C}_{\mathrm{DIR}}(k)$ of the directional signals, the HOA component $\boldsymbol{C}_{\mathrm{PD}}(k)$ of the predicted directional signals and the HOA component of the vector-based signals $\boldsymbol{C}_{\mathrm{VEC}}(k)$ by

$$\boldsymbol{C}_{\mathrm{PS}}(k) = \boldsymbol{C}_{\mathrm{DIR}}(k) + \boldsymbol{C}_{\mathrm{PD}}(k) + \boldsymbol{C}_{\mathrm{VEC}}(k).$$

Note that the HOA components $\boldsymbol{C}_{\mathrm{DIR}}(k), \boldsymbol{C}_{\mathrm{PD}}(k)$, and $\boldsymbol{C}_{\mathrm{VEC}}(k)$ are zero in case they are not processed as defined in the ChannelSideInfoData (Table 122) for frame *k*.

### 12.4.2.5 Ambience Synthesis

The ambient HOA component frame $\boldsymbol{C}_{\mathrm{AMB}}(k)$ is assumed to be composed according to

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$\boldsymbol{C}_{\mathrm{AMB}}(k) = \begin{bmatrix} \boldsymbol{c}_{\mathrm{AMB},1}(k) \\ \boldsymbol{c}_{\mathrm{AMB},2}(k) \\ \vdots \\ \boldsymbol{c}_{\mathrm{AMB},O}(k) \end{bmatrix}$$

with

$$\boldsymbol{c}_{\mathrm{AMB},n}(k) = [c_{\mathrm{AMB},n}(k,1) \quad c_{\mathrm{AMB},n}(k,2) \quad \dots \quad c_{\mathrm{AMB},n}(k,L)] \quad \text{for} \quad n = 1,\dots,O.$$

The first $O_{\mathrm{MIN}}$ coefficients of the ambient HOA component are obtained by

$$\begin{bmatrix} \boldsymbol{c}_{\mathrm{AMB},1}(k) \\ \boldsymbol{c}_{\mathrm{AMB},2}(k) \\ \vdots \\ \boldsymbol{c}_{\mathrm{AMB},O_{\mathrm{MIN}}}(k) \end{bmatrix} = \boldsymbol{\Psi}^{(N_{\mathrm{MIN}},N_{\mathrm{MIN}})} \cdot \begin{bmatrix} \boldsymbol{c}_{\mathrm{I,AMB},1}(k) \\ \boldsymbol{c}_{\mathrm{I,AMB},2}(k) \\ \vdots \\ \boldsymbol{c}_{\mathrm{I,AMB},O_{\mathrm{MIN}}}(k) \end{bmatrix},$$

where $\boldsymbol{\Psi}^{(N_{\mathrm{MIN}},N_{\mathrm{MIN}})}$ denotes the mode matrix of order $N_{\mathrm{MIN}}$ with respect to the predefined directions $\boldsymbol{\Omega}_n^{(N_{\mathrm{MIN}})}, n = 1,\dots,O_{\mathrm{MIN}}$, defined in Annex F.1.5. Note that the multiplication by the mode matrix $\boldsymbol{\Psi}^{(N_{\mathrm{MIN}},N_{\mathrm{MIN}})}$ represents the inverse spatial transform intended to invert the spatial transform applied in the encoder (see Annex C.5.3.3.2) for decorrelating the first $O_{\mathrm{MIN}}$ coefficients of the ambient HOA component. The sample values of the remaining coefficients of the ambient HOA component are set according to

$$c_{\mathrm{AMB},n}(k,l) = c_{\mathrm{I,AMB},n}(k,l) \quad \text{for} \quad O_{\mathrm{MIN}} < n \leq O \ .$$

#### 12.4.2.6 HOA composition

Finally the decoded HOA frame $\boldsymbol{C}(k)$ is computed by

$$\boldsymbol{C}(k) = \boldsymbol{C}_{\mathrm{AMB}}(k) + \boldsymbol{C}_{\mathrm{DIR}}(k) + \boldsymbol{C}_{\mathrm{PD}}(k) + \boldsymbol{C}_{\mathrm{VEC}}(k) = \boldsymbol{C}_{\mathrm{AMB}}(k) + \boldsymbol{C}_{\mathrm{PS}}(k).$$

### 12.4.3 HOA Renderer

#### 12.4.3.1 General

This section describes the conversion of the decoded Higher Order Ambisonics (HOA) representation to loudspeaker signals. In 12.4.2.4.2 the renderer architecture is presented; the design process of the rendering matrix for flexible rendering is given in 12.4.3.3. Clause 16 gives information about a gain and delay compensation for non-spherical placed loudspeaker setups and 12.4.3.4 explains NFC processing.

#### 12.4.3.2 Architecture overview

The conversion process is shown in Figure 45. The decoded HOA representation $\boldsymbol{C}$, here described as a matrix of $(N+1)^2$ rows and $T$ columns. $N$ denotes the HOA order (HoaOrder) and $T$ expresses the block size in number of samples. $\boldsymbol{C}$ is converted to the representation of loudspeaker signals $\boldsymbol{W}$ of size $L \times T$, where $L$ is the number of loudspeaker channels, by multiplication with the rendering matrix $\boldsymbol{D}$: $\boldsymbol{W} = \boldsymbol{D}\,\boldsymbol{C}$. A comprehensive description of the expected HOA format ($i.e.\ \boldsymbol{C}$) can be found in Annex F.1 and C.5.1.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

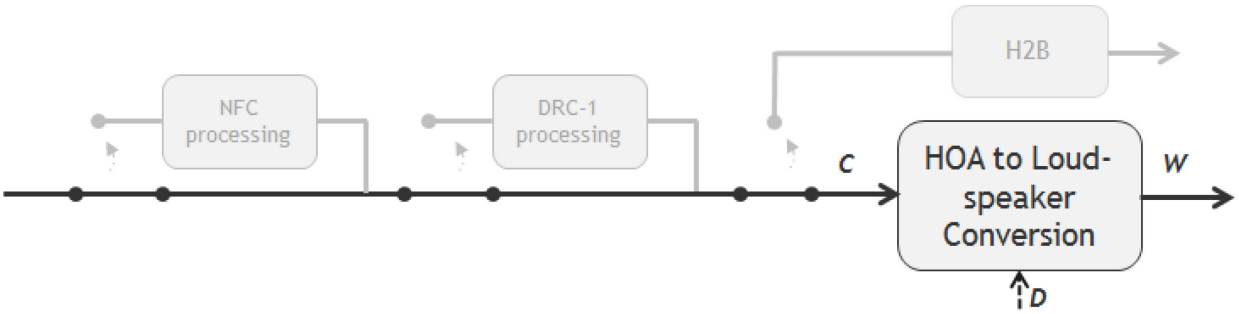



**Figure 45 — The HOA conversion process. In light gray: Preprocessing block NFC processing (should be switched active if UseNfc is true and the speaker listener distance $r_{max}$ is smaller NfcReference Distance), pre-processing block DRC-1 for HOA and as an alternative to rendering to loudspeakers, a computational efficient binaural rendering directly using the HOA coefficients (H2B, see 13.3.1). Computational more efficient but mathematically equivalent ways to implement the processing chain may be found in Annex G.**

The rendering process assumes that the loudspeakers are positioned with equal distances to the sweet spot (spherical setup) or that a gain and delay compensation is performed as post processing of the speaker signals before play out to create a virtual spherical setup. Clause 16 gives an informative description. A Peak Limiter should be implemented as post-processing.

The rendering state diagram is shown in Figure 46. On creation the renderer receives the positions of the loudspeaker setup $\Omega_{SPEAKER}$. $\Omega_{SPEAKER} = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_L]$, a matrix of size $3xL$, where L gives the number of loudspeakers and $\mathbf{r}_l = [\eta_l, \theta_l, \phi_l]^T = [\eta_l, \hat{\Omega}_l^T]^T$ is the position vector of speaker $l$. $\eta_l$ is the distance from the sweet spot (listening position) to speaker $l$ and $\hat{\Omega}_l^T = [\theta_l, \phi_l]$ are the related spherical angles. For the definition of the spherical coordinate system see Annex F.1.1, Figure F.1. Note that the definition of the spherical coordinate system used here differs from the one used within other parts of the standard. The inclination angle $\theta_l$ here is related to the elevation angle in rad by $\theta_l = \frac{\pi}{2} - \theta_{l,elevation}$.

For all HOA supported orders $N$ the renderer can create rendering matrices and store these in a data base or calculate these matrices on the fly when needed (see 12.4.3.3 for details on matrix construction). In case a HOA rendering matrix was signalled via HoaRenderingMatrixSet() (5.2.2.5) which matches the actual reproduction setup according to the matching rules defined in 10.3.1, the signalled HOA rendering matrix shall be applied for the rendering process.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

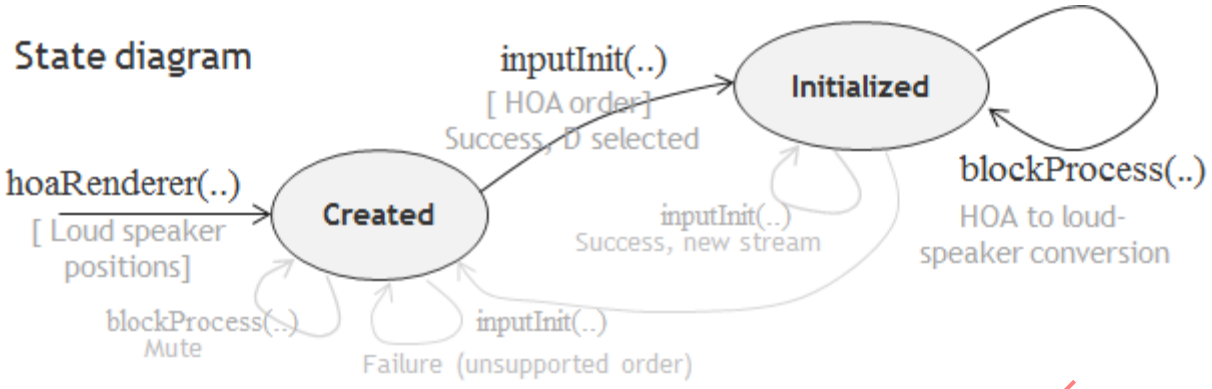



**Figure 46 — Renderer state diagram. There are two states of initialization. First the rendering matrices are created depending on the loudspeaker positions. A matrix suited for content to process is then selected using content meta data information**.

When new content is streamed to the renderer, it receives content information extracted from HOAConfig (). The renderer will try to select a rendering matrix $\boldsymbol{D}$ using the HOAOrder and create/select NFC filters for pre-processing if necessary, i.e. if UsesNfc is active in the bit stream and **NfcReferenceDistance** > $r_{max}$, with $r_{max}$ the maximum speaker distance of the listening setup. If successful, the renderer will reach state *Initialized*. Rendering block processing of HOA data then creates $\boldsymbol{L}$ speaker output signals (blockProcess()).

#### 12.4.3.3 Matrix design for flexible rendering

12.4.3.3.1 **General**

This section describes the design of energy preserving rendering matrices, where the number of HOA coefficients $(N+1)^2$ can be larger than the number of loudspeakers $L$. Energy preservation describes the characteristics that the HOA signal's loudness is preserved independent of the speaker setup and that constant amplitude spatial sweeps can be perceived equal loud after rendering. The section is divided into two parts: First the design for 3D speaker setups are described, then the design method for 2D speaker setups is presented. The 2D design method makes use of the 3D method because it uses virtual loud speakers placed at the pole positions of a (virtual) spherical speaker setup.

12.4.3.3.2 **Matrix design for 3D loudspeaker setups**

##### 12.4.3.3.2.1 Building Block Overview

The building blocks are shown in Figure 47. For each supported HOA order $N$ a matrix $\boldsymbol{D}$ is created. Main input to the design are the loudspeaker positions which are here indicated by $\mathfrak{D}_L$ as directions. $L$ loudspeaker directions are given by $\mathfrak{D}_L = [\,\hat{\Omega}_1, \ldots, \hat{\Omega}_L]$ with spherical angle $\hat{\Omega}_l = \left[\hat{\theta}_l, \hat{\phi}_l\right]^T$. A spherical setup is assumed and the speaker distances are neglected. $\hat{\theta}_l$ indicates the inclination and $\hat{\phi}_l$ the speaker azimuth, both in rad. The related coordinate system in Cartesian coordinated is centered at the sweet spot, the X axis points towards the ideal Center speaker position, the Y-axis to the left and the Z-axis to the top. An ideally placed Center speaker thus would have a $[\frac{\pi}{2}, 0]$position.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

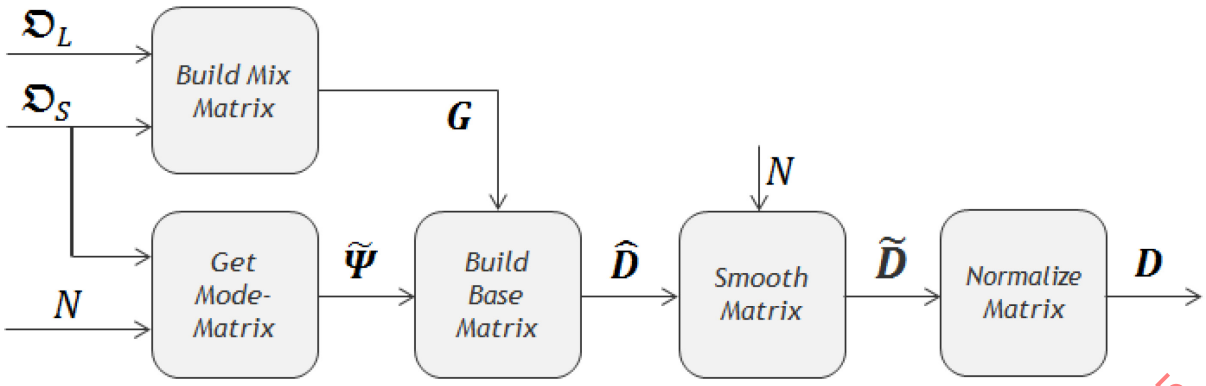



**Figure 47 — Building blocks of the matrix design procedure**

Additional input to the design process are ideal spherical design positions $\mathfrak{D}_S = [\Omega_1, \ldots, \Omega_S]$ with $\Omega_s = [\theta_s, \phi_s]^T$, characterized that they sample the unit sphere very regularly. A spherical grid of a $S = 324$ positions is defined, which enable the construction of matrices up to HOA order $N = 9$. The position tables are defined in F.11.

#### 12.4.3.3.2.2 Building block – Get Mode Matrix

A mode matrix $\widetilde{\Psi}$ related to $\mathfrak{D}_S$ and N is calculated or read from memory. $\widetilde{\Psi} = [y_1, \ldots y_S]$ with $y_s = [S_0^0(\Omega_s),\ S_1^{-1}(\Omega_s), \ldots, S_N^N(\Omega_S)]^H$, where vector $y_s$ holds the real valued spherical harmonic coefficients $S_n^m(\Omega_s)$. For the calculation of $S_n^m(\Omega_s)$ see the definition in Annex F.1.3.

#### 12.4.3.3.2.3 Building block – Build Mix Matrix

A Mix Matrix $G = [g_1, \ldots, g_S]$ of size $LxS$ is created which holds gain vectors $g_s$ to mix a direction $\Omega_s$ with speakers positioned at $\mathfrak{D}_L$. The method of robust panning is presented here which showed to be ideal for the following processing steps. The principle is depicted in Figure 48. For each and every plane wave from direction $\Omega_s$, $L$ panning gains in vector $g_s$ ($g_s = [g_{s,1}, \ldots, g_{s,L}]^T$), intended to attenuate plane waves emitted from $\mathfrak{D}_L$ (speakers) are created in a way that the error recorded at virtual microphones becomes minimal. Let $m_s \in C^{Mx1}$ denote $M$ microphone signals receiving the sound radiated from sources placed at $\Omega_s$ and $m_L \in C^{Mx1}$ the sound from sources placed at $\mathfrak{D}_L$. The virtual microphone signals are given by the plane wave transfer functions and an excitation signal $u$ by $m_s = h_{Ms} u$ and $m_L = H_{ML}\, g_s\, u$. The error $|m_s - m_L|_2^2$ becomes independent of $u$.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

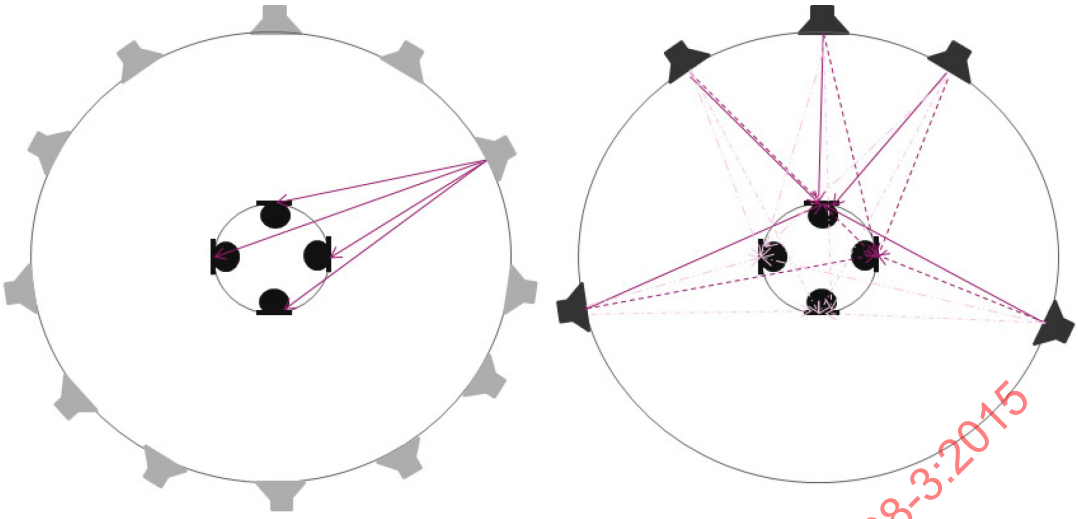


**Figure 48 — The principle of robust panning: Virtual microphones capture sound from direction $\Omega_s$ as a plane wave. Panning gains are calculated such that the sound recorded from loudspeaker directions $\mathfrak{D}_L = [\,\tilde{\Omega}_1, \ldots, \tilde{\Omega}_L]$ becomes as close as possible compared to first measurement.**

Minimizing the error term $\varepsilon_1 = |h_{Ms} - H_{ML}\, g_s|_2^2$ leads to the LMS-error solution for $g_s$. To make the gain vector more robust a regularization parameter is introduced that increases the costs if gains become large for sources positioned far away from direction $\Omega_s$:

*Minimize*: $\varepsilon_2 = |h_{Ms} - H_{ML}\, g_s|_2^2 + \beta\, |\mathcal{B}^T g_s|_2^2$ .

The LMS solution of this equation is given by:

$$g_s = \left(H_{ML}^H H_{ML} + \beta\, \mathcal{B}^H \mathcal{B}\right)^{-1} H_{ML}^H \; h_{Ms},$$

With $\mathcal{B}$ as $LxL$ diagonal matrix with diagonal elements of vector $\mathcal{b}$: $\mathcal{B} = diag(\mathcal{b})$; $(\,)^T$ is transposed, and $(\,)^H$ is transposed complex. $\mathcal{b} = [\mathcal{b}_1, \ldots, \mathcal{b}_L]^T$, where $\mathcal{b}_l$ needs to become larger the bigger the spherical angle of direction $\Omega_s$ to speaker direction $\tilde{\Omega}_l$ gets: $\mathcal{b}_l \sim (\measuredangle \Omega_s, \tilde{\Omega}_l)$ .

The transfer functions $h_{Ms}$, $H_{ML}$ are functions of frequency. To make the gains $g_s$ independent of frequency, the virtual microphone radius is selected as a function of frequency.

**Detailed description:**

1. A test frequency $f = 1000Hz$ is defined. Calculation of $k = \frac{2\pi f}{c_s}$,
   with $c_s$ is the speed of sound (340m/s). $\beta$ is selected dependent of the HOA order of the content to be rendered, see Table 119.

2. The spherical angles of virtual microphone positions $\mathfrak{D}_M$, with $\mathfrak{D}_M = [\,\ddot{\Omega}_1, \ldots, \ddot{\Omega}_M]$ with $\ddot{\Omega}_m = [\ddot{\theta}_m, \ddot{\phi}_m]^T$, are defined with $M = 256$ positions in Annex F.11. The microphone position radius is calculated by $r_M = \frac{0.4667\left(\sqrt{L} - 1\right)}{k}$
   with $L$ the number of loudspeakers.

3. Calculation of $H_{ML} \in C^{M\,x\,L}$ with matrix elements
   $H_{m,l} = e^{i\,k\,r_M\,\cos(\gamma_{l,m})}$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

where $\gamma_{l,m}$ is the spherical angle between $\ddot{\Omega}_m$ and $\hat{\Omega}_l$ :

$\cos(\gamma_{l,m}) = \cos(\hat{\theta}_l)\cos(\ddot{\theta}_m) + \sin(\hat{\theta}_l)\sin(\ddot{\theta}_m)\cos(\hat{\phi}_l - \ddot{\phi}_m)$. This is the plane wave transfer function and equal to $e^{i(k\hat{X}_l^T)\ddot{x}_m}$ where $\hat{X}_l$ are the speaker positions in Cartesian coordinates (with radius 1) and $\ddot{X}_m$ the microphone positions in Cartesian coordinates (with radius $r_M$).

4. Loop: $for(s = 1, s \leq S, s++)$

   a. Calculation of $h_{Ms} \in C^{Mx1}$ with vector elements

   $h_{m,s} = e^{ikr_M \cos(\hat{\gamma}_{s,m})}$

   where $\hat{\gamma}_{s,m}$ is the spherical angle between $\ddot{\Omega}_m$ and $\Omega_s$ :

   $\cos(\hat{\gamma}_{s,m}) = \cos(\theta_s)\cos(\ddot{\theta}_m) + \sin(\theta_s)\sin(\ddot{\theta}_m)\cos(\phi_s - \ddot{\phi}_m)$.

   b. Calculation of $B = diag(\mathscr{b})$ with:

   $\mathscr{b}_l = 1 - (0.5 + 0.5\cos(\bar{\gamma}_{s,l}))^2$

   and $\cos(\bar{\gamma}_{s,l}) = \cos(\theta_s)\cos(\hat{\theta}_l) + \sin(\theta_s)\sin(\hat{\theta}_l)\cos(\phi_s - \hat{\phi}_l)$.

   c. Calculation of $g_s = \left(H_{ML}^H H_{ML} + \beta B^H B\right)^{-1} H_{ML}^H h_{Ms}$.

   d. The $L$ gain values are complex with a very small imaginary part. We use only real valued gains: $\hat{g}_s = Re(g_s)$.

   e. Fill in the gain vector into the Mix Matrix :

   $G = [\hat{g}_1, .., \hat{g}_s, \ldots]$

**Table 131 — Values of regularization parameter β depending on HOA content order**.

| HOA order $N$ | 1 | 2 | 3 | 4 | >4 |
|---|---|---|---|---|---|
| $\beta$ | 1000 | 1000 | 100 | 100 | 50 |

#### 12.4.3.3.2.4 Building block – Build Base Matrix

The compact singular value decomposition of the matrix product of the mode matrix and the transposed mixing matrix is calculated by:

$U S V^H = \Psi G^T$

Matrix $S = diag(S_1, \ldots, S_K)$ is diagonal with the singular values as diagonal elements. Let $S_{max}$ denote the maximal singular value.

A new diagonal matrix with $\hat{S} = diag(\hat{S}_1, \ldots, \hat{S}_K)$ is created with $\hat{S}_k = \begin{cases} 1 & if\ S_k/S_{max} \geq a \\ 0 & else \end{cases}$

A threshold value $a$ of –60dB was selected: $a = 10^{-3}$.

The Base Matrix is calculated as follows: $\hat{D} = V \hat{S} U^H$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 12.4.3.3.2.5 Building block – Smooth Matrix

The task of this building block is to smooth the directive properties of the renderer, i.e. to attenuate the back and side lobes of the speaker panning pattern with the cost of widening the main lobe. The smoothed rendering matrix $\tilde{\boldsymbol{D}}$ is created by:

$$\tilde{\boldsymbol{D}} = \hat{\boldsymbol{D}}\, diag\,(\mathfrak{w}),$$

where $\mathfrak{w}$ is a gain vector of size $(N+1)^2$. The process can be compared to windowing in time-frequency processing and is equal to a left convolution on $\mathcal{S}^2$ (on the unit sphere) in spatial domain. The smoothing gains of vector $\mathfrak{w}$ are constructed from a helper vector $\mathfrak{v}$ with $N$+1 elements.

To construct vector $\mathfrak{v}$ two different approaches are used:

— If $L \geq (N+1)^2$, i.e. if the number of loudspeakers is larger or equal to the number of HOA coefficients, so called $max_r_E$ coefficients are used. Algorithm:

  — Given are the rightmost zeros of the Lengendre Polynomials of increasing order *N*, starting with order 1 to order 13 in vector $\boldsymbol{Z}$: $\boldsymbol{Z}$ = `[0.0, 0.5574, 0.7746, 0.8611, 0.9062, 0.9325, 0.9491, 0.9603, 0.9682, 0.9739, 0.9782, 0.9816, 0.9842];`

  — select value $rE = z_{N+1}$ form $\boldsymbol{Z}$ . (In C++ notation rE = z[N])

  — set $\mathfrak{v}_1 = 1$ and get the remaining values by calculating the Legendre polynomial $P_n()$ of order n for the value rE:
  $\text{for}(n = 1; n < N; n++)$
  $\mathfrak{v}_{n+1} = P_n(rE)$

— if $L < (N+1)^2$ a Kaiser-Bessel right half-window design is used to calculate the coefficients $\mathfrak{v}_n$:

$$\mathfrak{v}_n = \frac{I_0\left(width\sqrt{1-\left(\frac{2(n+N-1)}{len-1}-1\right)^2}\right)}{I_0(width)} \quad \text{for } n = 1 \ldots N+1$$

where $I_0()$ denotes the zero-order Modified Bessel function of first kind with parameters $len = 2N+1$ and $width = N+1$.

Vector $\mathfrak{w}$ is now constructed from $\mathfrak{v}_n$ :$\mathfrak{w} = [\,\mathfrak{v}_1, \mathfrak{v}_2, \mathfrak{v}_2, \mathfrak{v}_2, \mathfrak{v}_3, \mathfrak{v}_3, \mathfrak{v}_3, \mathfrak{v}_3, \mathfrak{v}_3, \mathfrak{v}_4, \ldots, \mathfrak{v}_N]^T$,

which is be computed by:

$$\text{for } (n = 0; n \leq N, n++)$$

$$\text{for}(m = -n; n \leq n; m++)$$

$$\mathfrak{w}_{n^2+n+m+1} = \mathfrak{v}_{n+1}$$

#### 12.4.3.3.2.6 Building block – Normalize Matrix

The rendering matrix is derived by normalization by its Frobenius Norm:

$$\boldsymbol{D} = \frac{\tilde{\boldsymbol{D}}}{||\,\tilde{\boldsymbol{D}}||_f},$$

where $||\,\tilde{\boldsymbol{D}}||_f$ denotes the Frobenius matrix norm, $||\,\tilde{\boldsymbol{D}}||_f = \sum_{l=1}^{L}\sum_{n=1}^{(N+1)^2}\tilde{D}_{l,n}^2$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

12.4.3.3.3 **Matrix design for 2D loudspeaker setups**

A 2D speaker setup is detected if all speaker elevation positions are within 7 degrees to the horizontal plane or expressed in speaker inclinations $\hat{\theta}_l$ :

$$if\ (\hat{\theta}_l > \frac{\pi}{180}(90-7)\ \&\&\ \hat{\theta} < \frac{\pi}{180}(90+7)) \text{ is true for all speakers } l = 1,2,..,L.$$

Then two more (virtual) speakers are added so that the new number of speakers $L_2$ becomes $L_2 = L + 2$ and the new speaker directions are given by:

$\mathfrak{D}_{L_2} = [\hat{\Omega}_1,\dots,\hat{\Omega}_L\dots,\hat{\Omega}_{L+1}\dots,\hat{\Omega}_{L+2}]$ with spherical angle $\hat{\Omega}_l = [\hat{\theta}_l, \hat{\phi}_l]^T$ and $\hat{\Omega}_{L+1} = [0,0]^T$ and $\hat{\Omega}_{L+2} = [\pi, 0]^T$.

A matrix $\widehat{\boldsymbol{D}_2} \in R^{L_2 x (N+1)^2}$ is designed using the design method for 3D loudspeaker setups with the $\mathfrak{D}_{L_2}$ speaker position directions.

A matrix $\boldsymbol{D}_2 \in R^{L x (N+1)^2}$ with matrix elements $D_{2_{l,n}}$ is created from $\widehat{\boldsymbol{D}_2}$ by:

$$D_{2_{l,n}} = \widehat{D_2}_{l,n} + g\left(\widehat{D_2}_{L+1,n} + \widehat{D_2}_{L+2,n}\right),$$

for $= 1,2,..,L$ , $n = 1,2,..,(N+1)^2$ and $g = \frac{1}{\sqrt{L}}$.

Finally the rendering matrix $\boldsymbol{D}$ is derived by repeating the building block – normalize matrix:

$$\boldsymbol{D} = \frac{\boldsymbol{D}_2}{||\boldsymbol{D}_2||_f},$$

where $||\boldsymbol{D}_2||_f$ denotes the Frobenius matrix norm, $||\boldsymbol{D}_2||_f = \sum_{l=1}^{L}\sum_{n=1}^{(N+1)^2} D_{2_{l,n}}^2$.

### 12.4.3.4 NFC Processing

This section describes the design of Near-Field Compensating Filters. These are aimed to be applied to HOA signals before the building block called “HOA to Loudspeaker Conversion” in Figure 45.

NFC compensation is advantageously used for spherical microphone recordings or artificial mixing with very close sound sources. Let’s consider a HOA content of maximum order N. For a given HOA component of order index n, $0 \le n \le N$, NFC HOA uses $\mathrm{d_n} = \frac{\mathrm{h_0^{(2)}(kr_s)}}{\mathrm{h_n^{(2)}(kr_s)}} \frac{\mathrm{h_n^{(2)}(kr_{NFC})}}{\mathrm{h_0^{(2)}(kr_{NFC})}}$, with $\mathrm{h_n^{(2)}}$ as the spherical Hankel functions of the second kind, $\mathrm{r_s}$ is the source-sweet spot distance, and $\mathrm{r_{NFC}}$ is the compensation radius ($\mathrm{r_{NFC}}$ = NfcReferenceDistance).

The transfer function $\mathrm{d_n}$ introduced above applies to any HOA signal of order n. In the renderer, it is implemented in time domain as an n-th order IIR filter under the Direct Form II, with n/2 second order sections (or “cells”) for even n, or (n-1)/2 second order sections plus one first order section for odd n:

$$H_n(z) = \prod_{q=1}^{n/2} \frac{b_0^q + b_1^q z^{-1} + b_2^q z^{-2}}{a_0^q + a_1^q z^{-1} + a_2^q z^{-2}} \times \frac{b_0^{\frac{n+1}{2}} + b_1^{\frac{n+1}{2}} z^{-1}}{a_0^{\frac{n+1}{2}} + a_1^{\frac{n+1}{2}} z^{-1}}$$

$$= g\prod_{q=1}^{n/2} \frac{1 + b'^q_1 z^{-1} + b'^q_2 z^{-2}}{1 + a'^q_1 z^{-1} + a'^q_2 z^{-2}} \times \frac{1 + b'^{\frac{n+1}{2}}_1 z^{-1}}{1 + a'^{\frac{n+1}{2}}_1 z^{-1}}$$

the right factor (first order cell) being present only for odd orders n.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

For a given order index n, filter coefficients $a_i^q$ are computed as follows:

1. Set $\tau = r_{REC}/c$ with the sound speed c having a typical value of 340m/s.
2. Set $\alpha = 4 f_s \tau$
3. For second order cells (for 1≤q≤n/2), derive coefficients $a_0^q$, $a_1^q$ and $a_2^q$ from conjugate complex roots $X_{n,q}$ and $X_{n,n-q+1} = X_{n,q}^*$ exhibited in Table 132:

$$a_0^q = 1 - 2\frac{\mathrm{Re}(X_{n,q})}{\alpha} + \frac{|X_{n,q}|^2}{\alpha^2}$$

$$a_1^q = -2\left(1 - \frac{|X_{n,q}|^2}{\alpha^2}\right)$$

$$a_2^q = 1 + 2\frac{\mathrm{Re}(X_{n,q})}{\alpha} + \frac{|X_{n,q}|^2}{\alpha^2}$$

4. For odd order filters, derive the coefficients of the additional first order cell as follows:

$$a_0^{\frac{n+1}{2}} = 1 - \frac{X_{n,(n+1)/2}}{\alpha}, \quad a_1^{\frac{n+1}{2}} = -\left(1 + \frac{X_{n,(n+1)/2}}{\alpha}\right)$$

Filter coefficients $b_i^q$ are computed according the same procedure, where coefficients $a_i^q$ just have to be replaced by $b_i^q$ and step 1 is replaced by:

1. Set $\tau = r_{src}/c$

**Table 132 — Approximative values $X_{nq}$ of the roots of generalized Bessel polynomials for the first few orders n.**

| n | q | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | -2 | | |
| 2 | -3.0000+1.7321*j* | | |
| 3 | -3.6778+3.5088*j* | -4.6444 | |
| 4 | -4.2076+5.3148*j* | -5.7924+1.7345*i* | |
| 5 | -4.6493+7.1420*j* | -6.7039+3.4853*j* | -7.2935 |
| 6 | -5.0319+8.9853*j* | -7.4714+5.2525*j* | -8.4967+1.7350*j* |

# 13 Binaural Renderer

## 13.1 Introduction

Two binaural rendering tools are specified in subsequent sections

- Time-Domain Binaural Renderer
- Frequency-Domain Binaural Renderer

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

## 13.2 Frequency-Domain Binaural Renderer

### 13.2.1 Introduction

The Frequency-Domain Binaural Renderer can be used for generating the 3D Audio headphone signal for all types of input content (Channel and/or Object and/or HOA). The Frequency-Domain Binaural Renderer takes loudspeaker feeds as input signals.

The frequency-domain binaural processing is carried out as a decoder process converting the decoded signal into a binaural downmix signal that provides a surround sound experience when listened to over headphones.

The binaural renderer has as input the decoded data stream. The signal is processed by a QMF analysis filterbank as outlined in 4.B.18.2 of ISO/IEC 14496-3:2009 with the modifications stated in 8.6.4.2 of ISO/IEC 14496-3:2009. The renderer can also process QMF domain input data; the analysis filterbank is then omitted.

The binaural room impulse responses (BRIRs) are represented as complex QMF domain filters. The conversion from time domain binaural room impulse responses to the complex QMF filter representation is outlined in ISO/IEC 23003-1:2006, Annex B.

The frequency-domain binaural renderer consists of three processing blocks, a variable order filtering in frequency domain (VOFF), a sparse frequency reverberator (SFR), and a QMF domain Tapped-Delay Line (QTDL). Figure 49 illustrates the processing time-frequency regions for the three processing blocks.

The QMF domain BRIRs are truncated such that they only contain direct sound and early reflections (D&E). The transition point from early reflections to late reverberation $N_{\mathrm{Filter}}$ is determined frequency-dependent by a bandwise reverberation time analysis. The QMF domain audio signals and the QMF domain D&E BRIRs are then processed by a bandwise partitioned fast convolution to perform the binaural processing. Since filter order of the QMF domain D&E BRIRs is frequency dependent, this processing is referred to as *variable order filtering in frequency domain* (VOFF).

A QMF domain sparse frequency reverberator (SFR) is used to generate 2-channel QMF domain late reverberation. The reverberation module uses a set of frequency-dependent reverberation times and energy values to adapt the characteristics of the reverberation. The waveforms of the reverberation are based on a stereo downmix of the audio input signal and it is adaptively scaled in amplitude depending on an analysis of the audio signal.

A QMF domain Tapped-Delay Line (QTDL) processing is used for high frequency bands. The QTDL processing module exploits a time lag and a gain per each band per each channel to mimic the most significant binaural cues in the band. The VOFF and the QMF domain reverberator are turned off in the bands where the QTDL processing is used.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

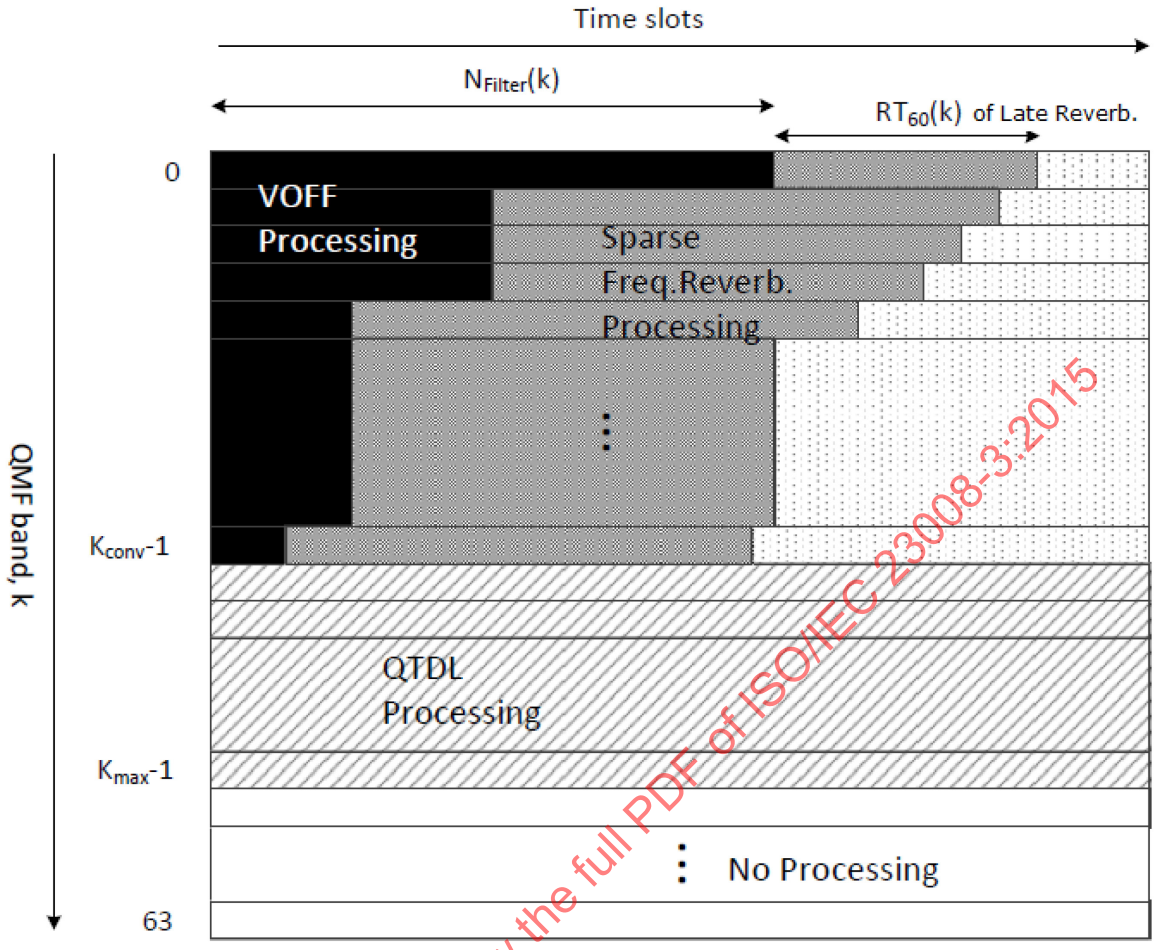



**Figure 49 — Graphical representation of the processing regions for the three processing blocks of the binaural renderer**

The 2-channel VOFF output and the 2-channel SFR output are then combined and mixed with the QTDL processed output and finally, two QMF synthesis filter banks compute the binaural time domain output signals as outlined in 4.6.18.4.2 of ISO/IEC 14496-3:2009. The renderer can also produce QMF domain output data; the synthesis filterbank is then omitted.

An overview of the QMF domain binaural processing is given in Figure 50.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

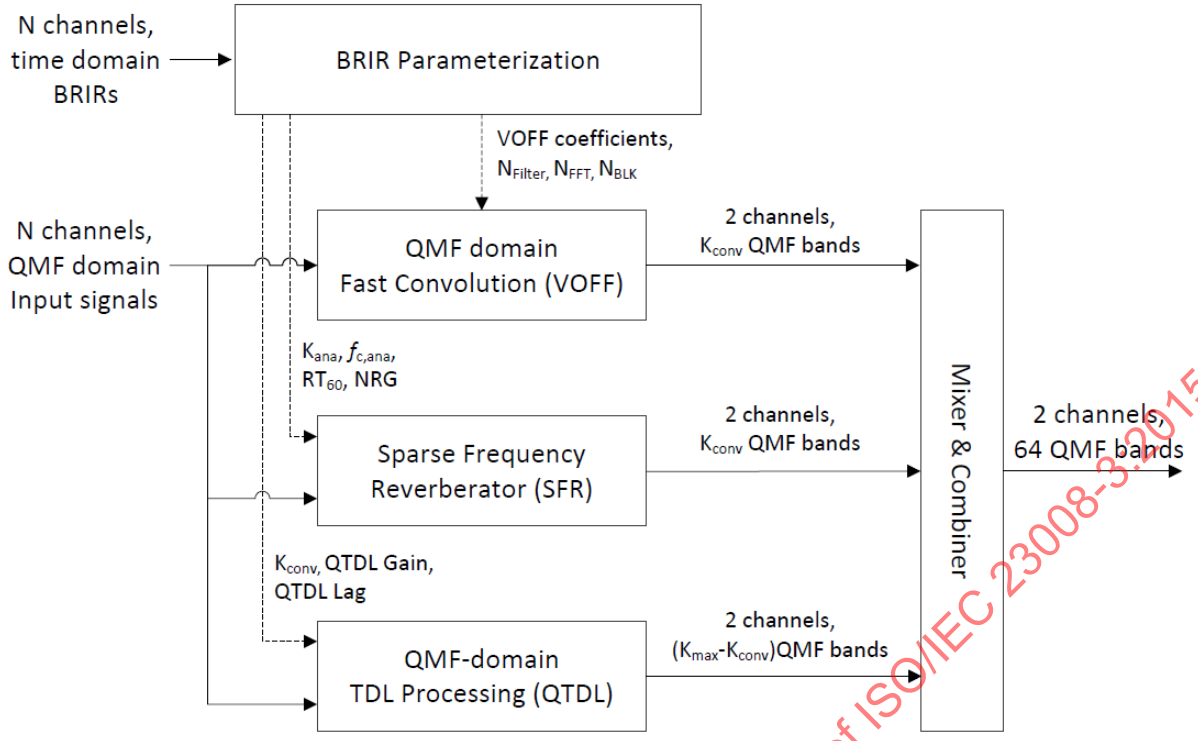



**Figure 50 — Overview of the binaural processing**

### 13.2.2 Definitions

Audio signals that are fed into the binaural renderer are referred to as *input signals* in the following. Audio signals that are the result of the binaural processing are referred to as *output signals*. The input signals of the binaural renderer are audio output signals of the core decoder.

The following variable definitions are used:

| | |
|---|---|
| $\alpha$ | Azimuth angle |
| $a$ | Factor in downmix matrix |
| $b_{\mathrm{reg}}$ | Axis intercept of a regression line |
| $\beta$ | Elevation angle |
| $c$ | BRIR output channel index (left/right), $c \in \{0,1\}$ |
| $c_{corr}$ | Correlation coefficient |
| $c_{\mathrm{eq},k}$ | Energy equalizing factor |
| $\mathbf{c}_{\mathrm{scale}} = [c_{\mathrm{scale},0}, c_{\mathrm{scale},1}]$ | Scaling factors |
| $\tilde{\mathbf{c}}_{\mathrm{scale}} = [\tilde{c}_{\mathrm{scale},0}, \tilde{c}_{\mathrm{scale},1}]$ | Smoothed scaling factors |
| $c_{60}$ | Real-valued correction factor for the calculation of the RT60 reverberation time |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $\mathrm{EDC}$ | Energy decay curve |
| $d_{\mathrm{FB}}$ | Frequency-dependent group delay of filter bank |
| $d$ | Delay in time domain samples |
| $d_{\mathrm{init}}$ | Initial delay of a BRIR set in time domain samples, propagation time |
| $d_{i,m}^{k}$ | QTDL lag |
| $f_{c,\mathrm{ana}}$ | Center frequencies of the late reverberation analysis bands |
| $f_s$ | Sampling rate |
| $g_{i,m,real}^{k}$ | Real value of QTDL Gain |
| $g_{i,m,imag}^{k}$ | Imaginary value of QTDL Gain |
| $\tilde{h}$ | 1-channel time domain impulse response |
| $\tilde{\mathbf{h}}_0^v = [\tilde{h}_{0,0}^v \cdots \tilde{h}_{0,N_{\mathrm{BRIR}}-1}^v]$ | Time domain representation of the left impulse response set excluding the LFE channels |
| $\tilde{\mathbf{h}}_1^v = [\tilde{h}_{1,0}^v \cdots \tilde{h}_{1,N_{\mathrm{BRIR}}-1}^v]$ | Time domain representation of the right impulse response set excluding the LFE channels |
| $\hat{\mathbf{h}}_0^{n,k} = [\hat{h}_{0,0}^{n,k} \cdots \hat{h}_{0,N_{\mathrm{BRIR}}-1}^{n,k}]$ | Complex valued QMF domain representation of the left impulse response set excluding the LFE channels |
| $\hat{\mathbf{h}}_1^{n,k} = [\hat{h}_{1,0}^{n,k} \cdots \hat{h}_{1,N_{\mathrm{BRIR}}-1}^{n,k}]$ | Complex valued QMF domain representation of the right impulse response set excluding the LFE channels |
| $\breve{\mathbf{H}}_0^{n,k,l} = [\breve{H}_{0,0}^{n,k,l} \cdots \breve{H}_{0,N_{\mathrm{BRIR}}-1}^{n,k,l}]$ | VOFF Coefficient for output channel 0 (Left) - Block-based pseudo-FFT domain representation of the left impulse response set excluding the LFE channels |
| $\breve{\boldsymbol{H}}_1^{n,k,l} = [\breve{H}_{1,0}^{n,k,l} \cdots \breve{H}_{1,N_{\mathrm{BRIR}}-1}^{n,k,l}]$ | VOFF Coefficient for output channel 1 (Right) - Block-based pseudo-FFT domain representation of the right impulse response set excluding the LFE channels |
| $\tilde{h}_{k_{\mathrm{FB}}}$ | Filtered version of a 1-channel time domain impulse response |
| $\tilde{h}_{k_{\mathrm{FB}},\mathrm{dB}}$ | Filtered normalized squared logarithmic 1-channel BRIR in analysis band $k_{\mathrm{FB}}$ |
| $\acute{h}_{k_{\mathrm{FB}},\mathrm{dB}}$ | Filtered smoothed logarithmic 1-channel BRIR in analysis band $k_{\mathrm{FB}}$ |
| $\tilde{h}_{k_{\mathrm{FB}},\mathrm{norm}}$ | Normalized filtered 1-channel BRIR |
| $K_{\mathrm{ana}}$ | Number of analysis frequency bands |
| $K_{\mathrm{ch}}$ | Number of BRIR pairs in a BRIR data set (number of measured positions) |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $K_{\text{conv}}$ | Number of QMF bands used for convolution |
| $K_{\text{LFE}}$ | Number of LFE BRIR pairs in a BRIR data set |
| $K_{LR}$ | Number of late reverberation analysis bands that are fed into the reverberator |
| $K_{\text{max}}$ | Number of QMF bands used for binaural processing |
| $K_{\text{trans}}$ | Number of QMF bands used in the analysis of the transition from early to late reflections |
| $k$ | QMF domain frequency band index |
| $K$ | Number of QMF domain frequency bands, $K = 64$ |
| $k_{\text{FB}}$ | Frequency band index of the late reverberation analysis |
| $k_s$ | Number of windows for the smoothing of a BRIR |
| $L$ | Length of audio frame in time domain samples, $L$ = outputFrameLength |
| $l_{\text{ana}}$ | Analysis length of the late reverberation analysis |
| $L_{\text{BRIR}}$ | Length of the measured BRIRs in time domain samples |
| $L_n$ | Length of an audio frame in QMF domain time slots. |
| $\mathbf{m}_{\text{conv}}$ | Vector to signal which channel of the input signal corresponds to which BRIR pair in the BRIR data set |
| $\mathrm{M}_{\text{DMX}}$ | Downmix matrix |
| $m_{\text{minv}}$ | Frequency-dependent minimum value in dB |
| $m_{\text{reg}}$ | Gradient of a regression line |
| $n$ | QMF domain time slots index |
| $N_{\text{Blk}}$ | Number of blocks |
| $N_{\text{BRIR}}$ | Number of BRIRs used in the parameterization and processing |
| $N_{\text{DMX,act}}$ | Number of active downmix channels |
| $n_{\text{DS}}$ | Maximum amplitude of a smoothed logarithmic BRIR |
| $\tilde{n}_{\text{DS}}$ | Mean amplitude around maximum of a smoothed logarithmic BRIR |
| $\mathrm{N}_{\text{Frm}}$ | Number of sub-frames |
| $N_{\text{in}}$ | Number of input audio channels |
| $N_{\text{out}}$ | Number of output audio channels, $N_{\text{out}} = 2$ |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $N_{\mathrm{Filter}}[k]$ | VOFF filter order corresponding to the k-th QMF domain frequency band |
| $N_{\mathrm{FFT}}[k]$ | FFT length |
| $n_{\mathrm{noise}}$ | Noise level in dB |
| $\mathrm{NRG}_{\mathrm{LR}}$ | Energy of the late reverberation |
| $P_i$ | Position (Azimuth and elevation angle) of a BRIR or audio channel |
| $\mathrm{RT}_{30}$ , $\mathrm{RT}_{60}$ | Reverberation times |
| $v$ | Time domain sample index |
| $v_0, v_5, v_{35}$ | Time domain indices with specified amplitude in the EDC |
| $v_{\mathrm{cross}}$ | Crossing sample of two lines |
| $v_{\mathrm{DS}}$ | Index of the maximum of a smoothed logarithmic BRIR |
| $v_{\mathrm{end}}$ | Maximum frequency-independent sample index used for the late reverberation analysis |
| $v_{\mathrm{end},f}$ | Frequency-dependent ending time domain sample index of the late reverberation analysis |
| $v_{\mathrm{noise}}$ | Index where a smoothed logarithmic BRIR is smaller than $n_{\mathrm{noise}} + 2$ for the first time |
| $v_{\mathrm{start}}$ | Frequency-dependent starting time domain sample index of the late reverberation analysis |
| $v_{\mathrm{trans}}$ | Frequency-dependent transition from early reflections to late reverberation in time domain samples (beginning from the direct sound) |
| $\tilde{\mathbf{y}}^v = [\tilde{y}_0^v \ \cdots \ \tilde{y}_{N_{\mathrm{in}}-1}^v]$ | $N_{\mathrm{in}}$-channel time domain input audio signal |
| $\hat{\mathbf{y}}^{n,k} = [\hat{y}_0^{n,k} \ \cdots \ \hat{y}_{N_{\mathrm{in}}-1}^{n,k}]$ | $N_{\mathrm{in}}$-channel QMF domain input audio signal |
| $\hat{\mathbf{z}}^{n,k}$ | Combined QMF domain signal of convolution output, reverb generator output and QTDL processing output. |
| $\tilde{\mathbf{z}}^v = [\tilde{z}_0^v, \tilde{z}_1^v]$ | 2-channel time domain output signal |
| $\check{\mathbf{z}}_{\mathrm{conv}}^{n,k} = [\check{z}_{0,\mathrm{conv}}^{n,k}, \check{z}_{1,\mathrm{conv}}^{n,k}]$ | VOFF processed signal in QMF domain frequency band $k$ |
| $\hat{\mathbf{z}}_{\mathrm{rev}}^{n,k} = [\hat{z}_{0,\mathrm{rev}}^{n,k}, \hat{z}_{1,\mathrm{rev}}^{n,k}]$ | Intermediate reverberation signal generated by the reverberator module in QMF domain frequency band $k$ |
| $\hat{\mathbf{z}}_{\mathrm{QTDL}}^{n,k} = [\hat{z}_{0,\mathrm{QTDL}}^{n,k}, \hat{z}_{1,\mathrm{QTDL}}^{n,k}]$ | QTDL processed signal in QMF domain frequency band $k$ |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

### 13.2.3 Parameterization of Binaural Room Impulse Responses

#### 13.2.3.1 Introduction

The frequency-domain binaural renderer needs specific metadata information describing the properties of the BRIR set. These values are calculated by a parameterization procedure and are stored in a dedicated file in a defined order. The file is a binary file written with 32 bit per sample, float values, little-endian ordering.

The parameterization takes into account as input parameters.

— N channels, time domain BRIRs

— Corresponding elevation and azimuth angles

It returns four groups of parameters

— General metadata: Elevation and azimuth of the BRIRs, the propagation time $d_{\text{init}}$, the number of processing band $K_{\max}$, the number of LFEs $K_{\text{LFE}}$, the LFE channel indices, the sampling rate of the BRIRs

— VOFF Parameters: The left/right VOFF coefficients $[\tilde{\mathbf{H}}_0^{n,k,l}\ \tilde{\mathbf{H}}_1^{n,k,l}]$, the VOFF filter length $N_{\text{Filter}}$, the FFT size per band $N_{\text{FFT}}$ and the number of blocks per bands $N_{\text{Blk}}$

— Reverberator Parameters: the number of analysis bands $K_{\text{ana}}$ used in the analysis of late reverberation, the center frequencies of the late reverberation analysis bands $f_{c,\text{ana}}$, the reverberation time $\text{RT}_{60}$ and energy $\text{NRG}_{\text{LR}}$ of the late reverberation

— QTDL Parameters: the left/right QTDL gain $[g_{i,m,real}^{k}\ g_{i,m,imag}^{k}]$, the left/right QTDL time lag $d_{i,m}^{k}$, the maximum band used for convolution $K_{\text{conv}}$

An overview of the BRIR parameterization is given in Figure 51.

In the block *Propagation Time Calculation*, the propagation time of the BRIRs is calculated to truncate them at the direct sound. In the block “Filter Converter”, the truncated time domain BRIRs are transformed to QMF domain BRIRs according to Annex B of ISO/IEC 23003-1:2006. The prototype filter coefficients for the filter conversion are used according to Table B.1 of ISO/IEC 23003-1:2006.

In the block *VOFF Parameter Generation*, a frequency-dependent RT20 reverberation time analysis is performed to determine the frequency-dependent transition from early reflections to late reverberation. Next, the truncated complex-valued QMF domain BRIRs up to the frequency-dependent transition are transformed into a VOFF coefficients $[\tilde{\mathbf{h}}_0^{n,k,l}\ \tilde{\mathbf{h}}_1^{n,k,l}]$. The VOFF coefficients $[\tilde{\mathbf{h}}_0^{n,k,l}\ \tilde{\mathbf{h}}_1^{n,k,l}]$ will be used to perform a fast convolution in the complex-valued QMF domain.

In the block *Sparse Frequency Reverberator Parameter Generation* (SFR Parameter Generation), the time domain BRIRs are analyzed by a one-third octave filter bank to determine the characteristics of the late reverberation. The reverberation time RT60 of the late reverberation and energy of the reverberation are used to control the sparse frequency reverberator.

In the block *QTDL Parameter Generation*, the peak locations and square root of real and imaginary energy of QMF domain BRIRs are analyzed to obtain QTDL parameters used in QTDL processing.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

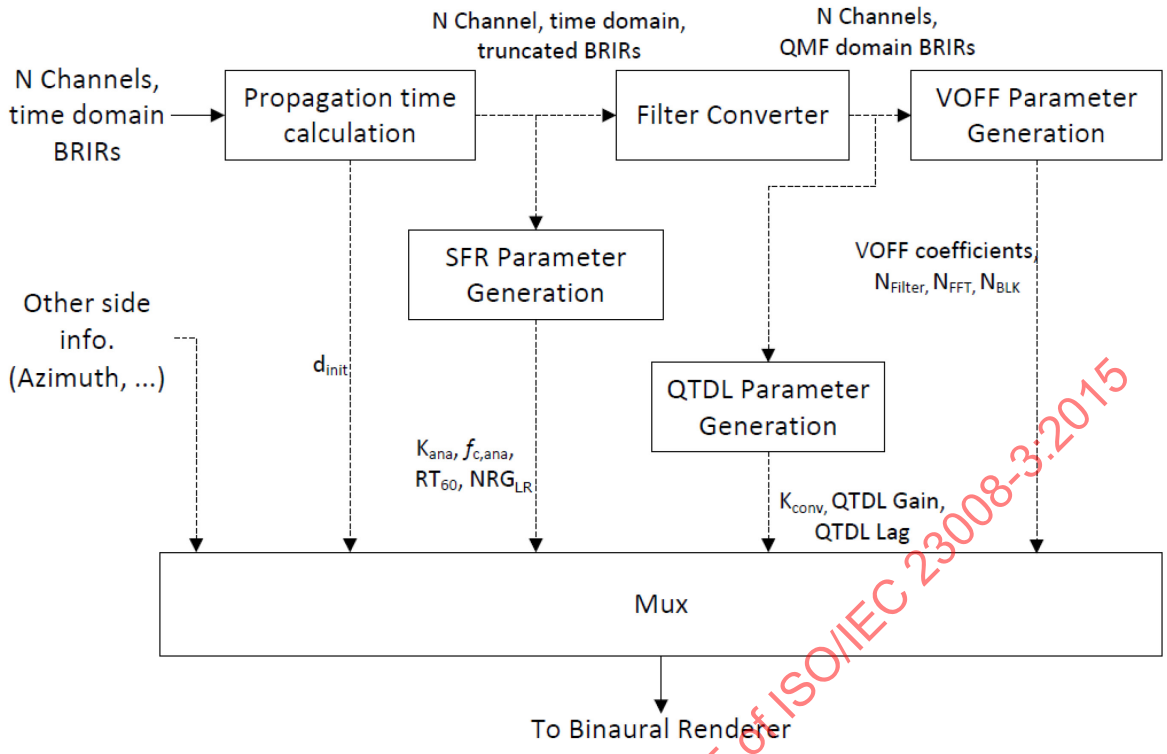



**Figure 51 — Details of BRIR Parameterization**

The analysis steps are specified in detail in the following sections.

#### 13.2.3.2 Propagation time $d_{\text{init}}$ calculation

The BRIRs contains a redundant portion between the 0th sample and the location of the impact of the direct sound. The time between the start of the BRIR and the direct sound is referred to as the propagation time. Since this portion only imposes additional delay for the binaural rendered output as well as additional complexity for processing, the BRIRs are trimmed to start at the direct sound. Therefore, the propagation time $d_{\text{init}}$ is determined by calculating the frame energy.

The following intermediate parameters are determined depending on the overall length of the input BRIRs:

If the length of the BRIRs in time domain samples exceeds 80ms

*flag_HRIR* = 0,

$L_{frm}$ = 32,

$N_{hop}$ = 8, and

$N_{iter} = 80e^{-3}f_s$ / $N_{hop}$

else

*flag_HRIR* = 1,

$L_{frm}$ = 8,

$N_{hop}$ = 1, and

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$N_{iter} = 80e^{-3} f_s / N_{hop}$

Then, the frame energy is obtained by

$$E(k) = \frac{1}{2N_{BRIR}} \sum_{m=1}^{N_{BRIR}} \sum_{i=0}^{1} \frac{1}{L_{frm}} \sum_{n=0}^{L_{frm}-1} h_{i,m}^{kN_{hop}+n} \quad k = 1, \ldots, N_{iter}$$

resulting in the propagation time

$$d_{\text{init}} = \frac{L_{frm}}{2} + N_{hop} * min\left[\arg_k \left(\frac{E(k)}{\max(E)} > -60dB\right)\right].$$

The sets of truncated left and right BRIRs (excluding the LFE channels) are then defined as

$$\mathbf{h}_0^v = \left[h_{0,0}^{v+d_{\text{init}}} \cdots h_{0,N_{\text{BRIR}}-1}^{v+d_{\text{init}}}\right] \text{ and } \mathbf{h}_1^v = \left[h_{1,0}^{v+d_{\text{init}}} \cdots h_{1,N_{\text{BRIR}}-1}^{v+d_{\text{init}}}\right],$$

with

$$N_{\text{BRIR}} = 2 \cdot K_{\text{ch}} - 2 \cdot K_{\text{LFE}} \text{ and } 0 \leq v < L_{\text{BRIR}} - d_{\text{init}}.$$

#### 13.2.3.3 Filter Conversion from time-domain BRIRs to QMF-domain BRIRs

The conversion of the time domain filters to the complex-valued QMF domain is carried out according to Annex B of ISO/IEC 23003-1:2006. The prototype filter coefficients for the filter conversion are used according to Table B.1 of ISO/IEC 23003-1:2006.

With the time domain representation of the impulse response sets excluding the LFE channels $[\mathbf{h}_0^v \; \mathbf{h}_1^v]$, a complex-valued QMF domain representation is obtained according to Annex B of ISO/IEC 23003-1:2006 as

$$\mathbf{h}_0^{n,k} = \left[h_{0,0}^{n,k} \cdots h_{0,N_{\text{BRIR}}-1}^{n,k}\right] \text{ and } \mathbf{h}_1^{n,k} = \left[h_{1,0}^{n,k} \cdots h_{1,N_{\text{BRIR}}-1}^{n,k}\right],$$

with

$$0 \leq n < n_{\text{end}}.$$

$$n_{\text{end}} = \lceil (L_{\text{BRIR}} - d_{\text{init}})/64 \rceil + 2$$

#### 13.2.3.4 Definition of the number of processing bands and convolution bands

The maximum processing band $K_{\max}$ and the number of convolution bands $K_{\text{conv}}$ can be received from Table 159 (**kMax** and **kConv**, respectively). In the bands 0 to $K_{\text{conv}}$ –1, the VOFF processing is carried out and in the bands $K_{\text{conv}}$ to $K_{\max}$ –1, the QTDL processing is carried out. When the QTDL processing is not used, $K_{\text{conv}}$ is overridden by $K_{\max}$. The SFR processing is carried out in the VOFF processing bands respectively when it is as illustrated in Figure 49.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 13.2.3.5 Parameter Generation for the Variable Order Filtering in Frequency domain (VOFF)

13.2.3.5.1 **General**

A frequency-dependent RT20 reverberation time analysis is carried out to determine filter order $N_{Filter}$ for the VOFF processing up to band $K_{conv} - 1$. The filter order $N_{Filter}$ for the VOFF is derived as specified in the following subclauses:

13.2.3.5.2 **Derivation of RT20**

The energy decay relief of a BRIR is obtained in the complex-valued QMF domain to derive the RT20 reverberation time as a measure for the transition time from the early reflections to the late reverberation.

The energy decay relief of a BRIR (channel $m$ of the BRIR set) in band k is defined as

$$EDR_{i,m}^{n,k} = \sum_{l=n}^{n_{end}-1} \left| \hat{h}_{i,m}^{l,k} \right|^2.$$

Then the averaged reverberation time RT20 is obtained by

$$RT_{20}^{k} = \frac{1}{2N_{BRIR}} \sum_{i=0}^{1} \sum_{m=0}^{N_{BRIR}-1} \min \left[ \arg_n \left( \frac{EDR_{i,m}^{n,k}}{EDR_{i,m}^{0,k}} < -20dB \right) \right].$$

13.2.3.5.3 **Filter Order Decision**

The VOFF filter order is determined by a logarithmic curve fitted reverberation time RT20. The curve fitting is not conducted for band 0. In the case of HRIR (f*lag_HRIR*=1), curve fitting is not applied.

To perform the curve fitting, two coefficients $a, and\ b$ should be obtained.

$$\bar{x} = \frac{1}{K_{max} - 1} \sum_{k=1}^{K_{max}-1} k$$

$$\bar{y} = \frac{1}{K_{max} - 1} \sum_{k=1}^{K_{max}-1} \log_2 RT_{20}^{k}$$

$$s_{xx} = \sum_{k=1}^{K_{max}-1} k^2 - (K_{max} - 1)\bar{x}^2$$

$$s_{xy} = \sum_{k=1}^{K_{max}-1} k(\log_2 RT_{20}^{k}) - (K_{max} - 1)\bar{x}\bar{y}$$

$$b = s_{xy}/s_{xx}$$

$$a = \bar{y} - b\bar{x}$$

The filter order $N_{Filter}$ for each band $k$ is derived as:

if $k$ = 0

$$N_{Filter}[0] = \min(2^{\lceil \log_2 RT_{20}^{0} \rceil}, n_{end})$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

else if $k < K_{\max}$

if *flag_HRIR* = 0

$$N_{Filter}[k] = \min(2^{\lfloor bk+a+0.5 \rfloor}, n_{end})$$

else

$$N_{Filter}[k] = \min(2^{\lfloor \log_2 \mathrm{RT}_{20}^{k}+0.5 \rfloor}, n_{end})$$

The transition $\nu_{\mathrm{trans}}[k]$ from direct sound and early reflections to the late reverberation tail in time domain samples is calculated from $N_{Filter}[k]$:

$$\nu_{\mathrm{trans}}[k] = 64 \cdot (N_{Filter}[k] - 2).$$

13.2.3.5.4 **VOFF Coefficient Generation**

The complex valued QMF domain BRIRs are truncated frequency-dependently at the corresponding value of the VOFF filter order $N_{\mathrm{Filter}}[k]$. The truncated complex valued QMF domain BRIRs are split into blocks to perform a block-wise fast convolution. These block-wise complex valued QMF domain BRIRs are therefore transformed to VOFF coefficients.

First, a frequency-dependent FFT size with a maximum of $2L_N$ is calculated according to

$$N_{\mathrm{FFT}}[k] = \min(2L_N, 2^{\lceil \log_2 2 \cdot N_{\mathrm{filter}}[k] \rceil}).$$

As the next step, the number of blocks per QMF band is determined

$$N_{\mathrm{Blk}}[k] = \frac{2^{\lceil \log_2 2 \cdot N_{\mathrm{filter}}[k] \rceil}}{N_{\mathrm{FFT}}[k]}.$$

The truncated complex valued QMF domain representations are obtained as

$$\tilde{\mathbf{h}}_c^{n,k} = \begin{cases} \hat{\mathbf{h}}_c^{s,k} & n < N_{filter}[k] \\ 0 & otherwise \end{cases}$$

If *flag_HRIR* == 1, the Sparse Frequency Reverberator is switched off. To prevent energy mismatch due to residue of BRIRs, energy compensation is carried out.

$$\tilde{\mathbf{h}}_c^{n,k} = \begin{cases} \sqrt{\frac{\sum_{s=0}^{s=n_{end}-1} real\left(\hat{\mathbf{h}}_c^{s,k}\right)^2}{\sum_{s=0}^{s=N_{\mathrm{filter}}[k]-1} real\left(\hat{\mathbf{h}}_c^{s,k}\right)^2}}\, real\left(\hat{\mathbf{h}}_c^{n,k}\right) + i \sqrt{\frac{\sum_{s=0}^{s=n_{end}-1} imag\left(\hat{\mathbf{h}}_c^{s,k}\right)^2}{\sum_{s=0}^{s=N_{\mathrm{filter}}[k]-1} imag\left(\hat{\mathbf{h}}_c^{n,k}\right)^2}}\, imag\left(\hat{\mathbf{h}}_c^{n,k}\right) & n < N_{filter}[k] \\ 0 & otherwise \end{cases}$$

The block-wise complex valued QMF domain representations are obtained by partitioning the truncated complex valued QMF domain representation.

$$\tilde{\mathbf{h}}_c^{n,k,l} = \begin{cases} \tilde{\mathbf{h}}_c^{(l-1)N_{FFT}[k]/2+n,k} & 0 \le n < N_{FFT}[k]/2 \\ 0 & otherwise \end{cases}$$

In each QMF band $k$ each block of the block-wise complex valued QMF domain representations are then transformed to a pseudo-FFT domain by a complex-valued FFT transform of length $N_{\mathrm{FFT}}[k]$ forming the block-based pseudo-FFT domain representation $\tilde{\boldsymbol{H}}_0^{n,k,l}$ (left ear) and $\tilde{\mathbf{H}}_1^{n,k,l}$ (right ear) for the blocks $0 \le l < N_{\mathrm{Blk}}[k]$. The $\tilde{\boldsymbol{H}}_c^{n,k,l}$ are the VOFF coefficients.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$\tilde{\mathbf{H}}_c^{n,k,l} = \boldsymbol{FFT}\left\{\mathbf{h}_c^{n,k,l}, N_{FFT}[k]\right\}$$

#### 13.2.3.6 Parameter Generation for the Sparse Frequency Reverberator

13.2.3.6.1 **General**

If *flag_HRIR* == 0, an analysis of the late reverberation parts of the BRIRs is carried out. Therefore, all of the BRIRs excluding the LFE BRIRs are filtered by a one-third octave filter bank.

13.2.3.6.2 **One-Third Octave Filterbank Analysis**

The one-third octave filter bank is realized by an infinite-impulse-response (IIR) filter with the coefficients given in Table 134 and Table 135. The filter coefficients are given in double precision. The filter is designed for a sampling frequency of 48 kHz and a maximum processing frequency of 18 kHz, resulting in 24 one-third octave bands.

If the BRIRs are sampled at a different sampling frequency, the same filter coefficients are used, resulting again in a maximum processing frequency of $0.75 \cdot f_s$.

The bands of the filtered BRIRs are delay aligned according to the group delay of the filters. The used delay numbers in time domain samples are given in Table 133.

**Table 133 — Delay for the delay alignment of the filtered BRIRs**

| Band $k'$ | Delay $d_{FB}[k']$ |
|---|---|
| 0 | 702 |
| 1 | 1448 |
| 2 | 1202 |
| 3 | 870 |
| 4 | 731 |
| 5 | 478 |
| 6 | 368 |
| 7 | 284 |
| 8 | 224 |
| 9 | 177 |
| 10 | 142 |
| 11 | 110 |
| 12 | 88 |
| 13 | 69 |
| 14 | 55 |
| 15 | 44 |
| 16 | 35 |
| 17 | 27 |
| 18 | 22 |
| 19 | 17 |
| 20 | 14 |
| 21 | 11 |
| 22 | 8 |
| 23 | 6 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

To delay align the frequency bands, they are zero-padded with the maximum delay number $d_{FB,max}$ at the beginning. Afterwards, each band is shifted to the beginning by its delay number and after that the first $d_{FB,max}$ samples are again cut away.

**Table 134 — Forward filter coefficients of the used 1/3 octave filter bank**

| Band | $f_{center}$[Hz] | Filter coefficients, forward path (B coefficients) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 99.21 | 0.000000002978966 | 0 | -0.000000008936898 | 0 | 0.000000008936898 | 0 | -0.000000002978966 |
| 1 | 125.00 | 0.000000005899584 | 0 | -0.000000017698751 | 0 | 0.000000017698751 | 0 | -0.000000005899584 |
| 2 | 157.49 | 0.000000011789497 | 0 | -0.000000035368490 | 0 | 0.000000035368490 | 0 | -0.000000011789497 |
| 3 | 198.43 | 0.000000023548606 | 0 | -0.000000070645819 | 0 | 0.000000070645819 | 0 | -0.000000023548606 |
| 4 | 250.00 | 0.000000047026089 | 0 | -0.000000141078268 | 0 | 0.000000141078268 | 0 | -0.000000047026089 |
| 5 | 314.98 | 0.000000093874874 | 0 | -0.000000281624623 | 0 | 0.000000281624623 | 0 | -0.000000093874874 |
| 6 | 396.85 | 0.000000187307829 | 0 | -0.000000561923486 | 0 | 0.000000561923486 | 0 | -0.000000187307829 |
| 7 | 500.00 | 0.000000373504653 | 0 | -0.000001120513960 | 0 | 0.000001120513960 | 0 | -0.000000373504653 |
| 8 | 629.96 | 0.000000744228866 | 0 | -0.000002232686597 | 0 | 0.000002232686597 | 0 | -0.000000744228866 |
| 9 | 793.70 | 0.000001481490914 | 0 | -0.000004444472741 | 0 | 0.000004444472741 | 0 | -0.000001481490914 |
| 10 | 1000.00 | 0.000002945568317 | 0 | -0.000008836704950 | 0 | 0.000008836704950 | 0 | -0.000002945568317 |
| 11 | 1259.92 | 0.000005847694618 | 0 | -0.000017543083853 | 0 | 0.000017543083853 | 0 | -0.000005847694618 |
| 12 | 1587.40 | 0.000011587291750 | 0 | -0.000034761875251 | 0 | 0.000034761875251 | 0 | -0.000011587291750 |
| 13 | 2000.00 | 0.000022906469387 | 0 | -0.000068719408160 | 0 | 0.000068719408160 | 0 | -0.000022906469387 |
| 14 | 2519.84 | 0.000045150610814 | 0 | -0.000135451832443 | 0 | 0.000135451832443 | 0 | -0.000045150610814 |
| 15 | 3174.80 | 0.000088673275133 | 0 | -0.000266019825400 | 0 | 0.000266019825400 | 0 | -0.000088673275133 |
| 16 | 4000.00 | 0.000173370214378 | 0 | -0.000520110643135 | 0 | 0.000520110643135 | 0 | -0.000173370214378 |
| 17 | 5039.68 | 0.000337103447126 | 0 | -0.001011310341379 | 0 | 0.001011310341379 | 0 | -0.000337103447126 |
| 18 | 6349.60 | 0.000651075235971 | 0 | -0.001953225707913 | 0 | 0.001953225707913 | 0 | -0.000651075235971 |
| 19 | 8000.00 | 0.001247284713280 | 0 | -0.003741854139840 | 0 | 0.003741854139840 | 0 | -0.001247284713280 |
| 20 | 10079.37 | 0.002366322106562 | 0 | -0.007098966319686 | 0 | 0.007098966319686 | 0 | -0.002366322106562 |
| 21 | 12699.21 | 0.004438169688363 | 0 | -0.013314509065088 | 0 | 0.013314509065088 | 0 | -0.004438169688363 |
| 22 | 16000.00 | 0.008214651510826 | 0 | -0.024643954532479 | 0 | 0.024643954532479 | 0 | -0.008214651510826 |
| 23 | 20158.74 | 0.014980703571592 | 0 | -0.044942110714776 | 0 | 0.044942110714776 | 0 | -0.014980703571592 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 135 — Backward filter coefficients of the used 1/3 octave filter bank**

| Band | $f_{center}$[Hz] | Filter coefficients, backward path (A coefficients) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 99.21 | 1 | -5.993751016763135 | 14.969279072784566 | -19.939603992021354 | 14.940647795274490 | - 5.970844821630246 | 0.994272962360464 |
| 1 | 125.00 | 1 | -5.991961445134535 | 14.960639671319811 | -19.922939882503549 | 14.924596293523440 | - 5.963124398568642 | 0.992789761382594 |
| 2 | 157.49 | 1 | -5.989609767964868 | 14.949371631318428 | -19.901379920011749 | 14.904008213044975 | - 5.953314349635694 | 0.990924193325314 |
| 3 | 198.43 | 1 | -5.986493089373522 | 14.934568071720706 | -19.873322981098088 | 14.877492822310906 | - 5.940823535229365 | 0.988578711974606 |
| 4 | 250.00 | 1 | -5.982322583166484 | 14.914956502895425 | -19.836561944151942 | 14.843176218160425 | - 5.924879679481255 | 0.985631486962872 |
| 5 | 314.98 | 1 | -5.976681836082384 | 14.888729126620820 | -19.788018121780837 | 14.798506980371531 | - 5.904466854668645 | 0.981930710405792 |
| 6 | 396.85 | 1 | -5.968963204438088 | 14.853287019657653 | -19.723347050471695 | 14.739973841057109 | - 5.878238363575107 | 0.977287777184542 |
| 7 | 500.00 | 1 | -5.958270126407491 | 14.804853333473961 | -19.636351633672902 | 14.662693694537232 | - 5.844394434554799 | 0.971469244020969 |
| 8 | 629.96 | 1 | -5.943266716356360 | 14.737888463499093 | -19.518109356078305 | 14.559810012283357 | - 5.800509649746865 | 0.964187554640025 |
| 9 | 793.70 | 1 | -5.921945879281351 | 14.644209432330609 | -19.355682487394688 | 14.421618545659840 | - 5.743289060834721 | 0.955090675488847 |
| 10 | 1000.00 | 1 | -5.891272053317806 | 14.511677079902135 | -19.130239207390304 | 14.234312652347755 | - 5.668224665450618 | 0.943751061239065 |
| 11 | 1259.92 | 1 | -5.846632613931893 | 14.322276215973289 | -18.814390396441560 | 13.978227786574614 | - 5.569116536160665 | 0.929654821150504 |
| 12 | 1587.40 | 1 | -5.781001286253467 | 14.049405018511159 | -18.368601439334121 | 13.625500348622651 | - 5.437419214533596 | 0.912192664215007 |
| 13 | 2000.00 | 1 | -5.683678077357811 | 13.654293601443985 | -17.736819219552288 | 13.137230968058500 | - 5.261383586695196 | 0.890655254306507 |
| 14 | 2519.84 | 1 | -5.538431150435243 | 13.081896267444138 | -16.842288758666236 | 12.460762205420412 | - 5.025009003406554 | 0.864237092185886 |
| 15 | 3174.80 | 1 | -5.320855496910868 | 12.257825487897835 | -15.586541172177160 | 11.528970962883021 | - 4.706945111668716 | 0.832054991343535 |
| 16 | 4000.00 | 1 | -4.994862939199035 | 11.090899392402983 | -13.858558427508518 | 10.266297975027744 | - 4.279774723365612 | 0.793189508854480 |
| 17 | 5039.68 | 1 | -4.508633849835456 | 9.492307997838553 | -11.567081708283867 | 8.611562042517182 | - 3.710717285192899 | 0.746759879904296 |
| 18 | 6349.60 | 1 | -3.791566479016410 | 7.433709378886649 | -8.711008375712691 | 6.575213188380568 | - 2.965921277956436 | 0.692044028301664 |
| 19 | 8000.00 | 1 | -2.756730866820556 | 5.078384616947228 | -5.476030603444230 | 4.351407566193580 | - 2.022277271752888 | 0.628653072944205 |
| 20 | 10079.37 | 1 | -1.319726797918420 | 3.001645881863994 | -2.235723204296885 | 2.473284706980526 | - 0.892513256112752 | 0.556761178207907 |
| 21 | 12699.21 | 1 | 0.543821839225891 | 2.367900783391379 | 0.842176405781307 | 1.858884029029895 | 0.331706060878868 | 0.477372283671088 |
| 22 | 16000.00 | 1 | 2.667654841026451 | 4.479530540428327 | 4.526976093111068 | 3.286179336952533 | 1.426496187396251 | 0.392570938427190 |
| 23 | 20158.74 | 1 | 4.472803785543190 | 8.630679353924112 | 9.234059565895342 | 5.792824924160126 | 2.020263319760948 | 0.305653299838751 |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

In addition to the 1/3 octave filter bank, an IIR low pass filtering is conducted. The filter coefficients are given in Table 136. The resulting 25 frequency bands are now addressed by the index $k_{FB}$ with the lowest band $k_{FB} = 0$ and the 24 bands from the 1/3 octave filter bank analysis $k_{FB} = 1$ to $k_{FB} = 24$. The filtered version of a 1-channel time domain impulse response $\tilde{h}$ in band $k_{FB}$ is called $\tilde{h}_{k_{FB}}$.

**Table 136 — Filter coefficients of the additional low pass filter**

| Forward Path, B coefficents | Backward Path, A coefficients |
|---|---|
| 4.07451850037432e-14 | 1 |
| 2.44471110022459e-13 | -5.95448186943460 |
| 6.11177775056149e-13 | 14.7734441411088 |
| 8.14903700074865e-13 | -19.5489429752463 |
| 6.11177775056149e-13 | 14.5509829161372 |
| 2.44471110022459e-13 | -5.77650412293260 |
| 4.07451850037432e-14 | 0.955501910370147 |

The transition values $\nu_{trans}[k]$ from the frequency-dependent RT20 reverberation time analysis are mapped from QMF bands to the 25 analysis frequency bands. For each of the analysis bands the transition from the one specific QMF band is used where the difference of the center frequency of the analysis band and the QMF band is minimal.

Then, the frequency-dependent value $\nu_{start}[k_{FB}] = \nu_{trans}[k_{FB}] + d_{init}$ (transition from early reflections to late reverberation in time domain samples plus the initial delay of the BRIR set) is used as a starting point for the following analysis.

The last maximally used sample for the analysis is calculated by $\nu_{end} = 2^{\lfloor \log_2(L_{BRIR}/4) \rfloor + 1}$.

A frequency-dependent terminal point sample is calculated:

$$\nu_{end,f}[k_{FB}] = \begin{cases} \nu_{end} & \text{if } k_{FB} > 3 \text{ and } k_{FB} < 22 \\ \nu_{end} - \nu_{start}[k_{FB}] & \text{else} \end{cases}$$

For each of the BRIRs which is not an LFE BRIR and has non-zero values and each of the 25 analysis bands the following steps are carried out between $\nu_{start}[k_{FB}]$ and $\nu_{end,f}[k_{FB}]$.

13.2.3.6.3 **Noise Floor Estimation**

A noise floor estimation is carried out for band $k_{FB} = 4$ or higher to determine the amount of noise in the filtered BRIRs. Therefore, the filtered BRIR is normalized, squared and the logarithm is calculated:

$$\tilde{h}_{k_{FB},dB}[\nu] = 10 \cdot \log_{10}\left(\frac{\tilde{h}_{k_{FB}}[\nu] + 10^{-20}}{\max(|\tilde{h}_{k_{FB}}|)}\right)^2$$

A smoothed logarithmic BRIR $\hat{h}_{k_{FB},dB}$ is then determined by calculating the mean value in overlapping windows (50% overlap) of 0.0025 seconds duration. The number of windows is $k_s$.

It is then determined where the smoothed logarithmic BRIR rises above a defined frequency-dependent minimum value in dB. The minimum value is defined as

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$m_{minv}[k_{FB}] = \begin{cases} -98 & k_{FB} = 0 \\ m_{minv}[k_{FB}-1] + 2 & k_{FB} < 22 \\ \left\lfloor \left( \frac{1}{k_s} \cdot \sum_{\nu=0}^{k_s-1} \acute{h}_{k_{FB},dB}[\nu] \right) + 0.5 \right\rfloor & \text{else} \end{cases}$$

resulting in a minimum value of -90dB in band 4.

The last 20% of the part of smoothed logarithmic BRIR that rises above the minimum value is used to determine the noise level $n_{noise}$ by a mean-value calculation.

If the smoothed logarithmic BRIR never rises above the minimum value, the noise level is determined to be the mean value of the whole smoothed logarithmic BRIR.

The time domain sample where the BRIR is assumed to have reached this noise level is then defined by calculating the crossing point of two lines, with the one line being parallel to the noise level plus 2 dB and the other line being a decaying regression line starting near the direct sound with a defined gradient.

The gradient of the regression line is defined as

$$m_{reg} = \frac{((n_{noise} + 2) - ((n_{DS} + \tilde{n}_{DS})/2)}{\left(\left\lfloor (\nu_{noise} + 1) \cdot \left(0.0025 \cdot \frac{f_s}{2}\right) + 0.5 \right\rfloor\right) - \left(\left\lfloor (\nu_{DS} + 1) \cdot \left(0.0025 \cdot \frac{f_s}{2}\right) + 0.5 \right\rfloor\right)}$$

with

$$\tilde{n}_{DS} = \frac{1}{l} \cdot \sum_{\nu=\max(0,\nu_{DS}-10)}^{\nu_{DS}+10} \acute{h}_{k_{FB},dB}[\nu]$$

$$l = \nu_{DS} + 10 - \max(0, \nu_{DS} - 10) + 1$$

$\nu_{DS}$ being the index of the maximum absolute value of the smoothed logarithmic BRIR, $n_{DS}$ the amplitude at $\nu_{DS}$ and $\nu_{noise}$ being the index where the smoothed logarithmic BRIR is smaller than $n_{noise} + 2$ for the first time. The axis intercept of the regression line is defined as

$$b_{reg} = \frac{(n_{DS} + \tilde{n}_{DS})}{2} - m_{reg} \cdot \left\lfloor (\nu_{DS} + 1) \cdot \left(0.0025 \cdot \frac{f_s}{2}\right) + 0.5 \right\rfloor.$$

The time domain sample $\nu_{cross}$ where the regression line crosses the parallel line for the first time specifies the beginning of the noise floor. This value is used to define the end of analysis in the corresponding frequency bands for the following parameterization steps.

$$\nu_{end,f}[k_{FB}] = \nu_{start}[k_{FB}] + \nu_{cross}[k_{FB}] \qquad \text{for } k_{FB} > 3$$

The analysis length $l_{ana}$ is defined as $\nu_{cross}[k_{FB}] + 1$.

13.2.3.6.4 **Determination of the RT60 reverberation time**

Between $\nu_{start}[k_{FB}]$ and $\nu_{end,f}[k_{FB}]$ the RT60 time is determined. Therefore the EDC (Energy Decay Curve, Schroeder integral) of the filtered BRIR is calculated by performing the following steps:

— The filtered BRIR is normalized

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$\tilde{h}_{k_{FB},norm}[\nu] = \frac{|\tilde{h}_{k_{FB}}[\nu]|}{\max(|\tilde{h}_{k_{FB}}|)}$$

— A factor of 1.0E-20 is added where $h_{k_{FB},norm}[\nu]$ is equal to zero

— The EDC is defined as

$$EDC[\nu] = 10 \cdot \log_{10}\left(\frac{\sum_{k=\nu}^{l_{ana}-1}\tilde{h}_{k_{FB},norm}[k]}{\sum_{k=0}^{l_{ana}-1}\tilde{h}_{k_{FB},norm}[k]}\right)$$

Index $\nu_0$ is associated to be the index of the sample where the EDC is bigger than zero for the last time. If such an index does not exist, then $\nu_0$ is set to zero.

Now, the index is determined where the EDC gets smaller than $EDC[\nu_0] - 5$ for the first time. This index is called index $\nu_5$. Next, the index is determined where the EDC gets smaller then $EDC[\nu_5] - 35$ for the first time. Accordingly, this index is called index $\nu_{35}$. The number of samples between $\nu_5$ and $\nu_{35}$ defines the reverberation time RT30. To gain the reverberation time RT60, the RT30 value is multiplied by a factor of $c_{60} = 2$.

An additional correction of the calculated RT60 reverberation time is introduced. If $\nu_{35}$ is bigger than $l_{ana} \cdot 0.95$ then $\nu_{35}$ is set to

$$\nu_{35} = \lfloor l_{ana} \cdot 0.95 + 0.5 \rfloor$$

and the correction factor is defined as

$$c_{60} = \frac{-65}{EDC[\nu_{35}]}.$$

If $\nu_5$ is bigger than $l_{ana} \cdot 0.5$, then $\nu_5$ is set to zero.

The frequency-dependent reverberation time RT60 is calculated by

$$RT_{60} = RT_{30} \cdot c_{60} = \left(\frac{\nu_{35}}{f_s} - \frac{\nu_5}{f_s}\right) \cdot c_{60}.$$

13.2.3.6.5 **Determination of the Energy**

The energy (absolute value squared) of each sample between $\nu_{start}[k_{FB}]$ and $\nu_{end}[k_{FB}]$ is summed up to give the total energy of the late reverberation:

$$NRG_{LR} = \sum_{\nu=0}^{l_{ana}-1} |\tilde{h}_{k_{FB}}[\nu]|^2.$$

The values of RT60 and energy are now averaged for each of analysis bands to reflect the characteristics of the whole BRIR set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 13.2.3.7 QTDL Parameter Generation

The QMF domain TDL processing is an efficient binaural rendering tool for the high frequency bands. In a QTDL-enabled QMF band, one tap delayed pulse mimics the most significant binaural cues of the interaural time difference (ITD) and the interaural level difference (ILD) in the band.

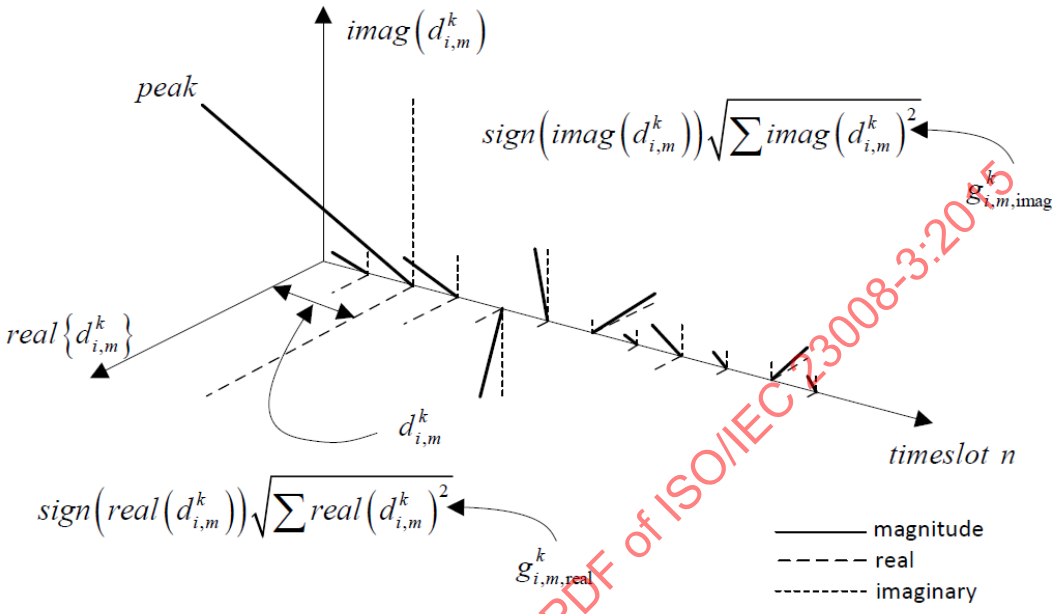



**Figure 52 — Geometric descriptions of QTDL parameters**

As described in Figure 52 the parameter generation for QTDL is carried out to obtain a QTDL lag parameter and two QTDL gain parameters per output channel (left/right), per band from $K_{\mathrm{conv}}$ to $K_{\max}$-1, and per input channel.

The QTDL lag $d_{i,m}^{k}$ is obtained as the location of the maximum peak of the magnitude of the complex-valued QMF domain representation.

$$d_{i,m}^{k} = \arg\max_{n}\left(|\mathrm{h}_{i,m}^{n,k}|^{2}\right)$$

The QTDL gain $g_{i,m}^{k}$ is the square root of the energy of the real/imaginary values of QMF domain representation.

$$g_{i,m,real}^{k} = \mathrm{sign}\left\{real\left(\mathrm{h}_{i,m}^{d_{i,m}^{k},k}\right)\right\}\sqrt{\sum_{l=0}^{n_{end}} real\left(\mathrm{h}_{i,m}^{l,k}\right)^{2}}$$

$$g_{i,m,imag}^{k} = \mathrm{sign}\left\{imag\left(\mathrm{h}_{i,m}^{d_{i,m}^{k},k}\right)\right\}\sqrt{\sum_{l=0}^{n_{end}} imag\left(\mathrm{h}_{i,m}^{l,k}\right)^{2}}$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

#### 13.2.3.8 Multiplexing of BRIR Parameters

The calculated BRIR metadata information is stored according to the defined interface (see 17.4.2).

### 13.2.4 Frequency-Domain Binaural Processing

The frequency-domain binaural renderer operates on contiguous, non-overlapping frames of length $L$ samples of the input audio signals and outputs one frame of $L$ samples. A standard frame length of $L$ = 4096 is defined.

#### 13.2.4.1 Initialization and Preprocessing

13.2.4.1.1 **General**

The initialization of the binaural renderer is carried out before the processing of the audio samples delivered by the core decoder takes place. The initialization consists of several processing steps.

13.2.4.1.2 **Reading of BRIR Metadata and pseudo-FFT BRIRs**

The binaural renderer reads BRIR metadata according to the defined interface (see 17.4.2).

#### 13.2.4.2 Audio signal processing

13.2.4.2.1 **General**

The audio processing of the binaural renderer obtains $N_{\mathrm{in}}$ input channels from the core decoder and generates a binaural output signal consisting of $N_{\mathrm{out}}$= 2 channels.

The processing takes as input

— the decoded audio data from the core decoder,

— the BRIR information from the parameterization, as is:

  — the general BRIR side information

  — the VOFF coefficients set and side information,

  — the frequency-dependent parameter set that is used by the QMF domain reverberator to generate the late reverberation,

  — and the QTDL gains and QTDL time lag set and side information

— and a parameter signaling the channel configuration of the audio data to process.

The channel configuration parameter is defined by the geometric position data (i.e. azimuth and elevation angles) associated with the input channels. The geometric data can either be signaled implicitly by CICPspeakerLayoutIdx or CICPspeakerIdx (see ISO/MPEG 23001-8) or explicitly.

13.2.4.2.2 **QMF analysis of the audio signal**

As the first processing step, the binaural renderer transforms $L$ time domain samples of the $N_{\mathrm{in}}$-channel time domain input signal $\tilde{\mathbf{y}}^{\mathrm{v}} = \lfloor \tilde{y}_0^{\mathrm{v}} \ \cdots \ \tilde{y}_{N_{\mathrm{in}}-1}^{\mathrm{v}} \rfloor$ to an $N_{\mathrm{in}}$ -channel QMF domain signal representation of dimension $L_{\mathrm{n}}$ QMF time slots (slot index $\mathrm{n}$) and $K$ = 64 frequency bands (band index $k$). In case of the standard framelength of $L$ =4096 samples, $L_{\mathrm{n}}$ is equal to 64 (i.e. $L_{\mathrm{n}}$ = $L$/64).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

A QMF analysis according to as outlined in 4.B.18.2 of ISO/IEC 14496-3:2009 with the modifications stated in 8.6.4.2 of ISO/IEC 14496-3:2009 is performed on a frame of the time domain signal $\tilde{\mathbf{y}}^{v}$ to gain a frame of the QMF domain signal $\hat{\mathbf{y}}^{n,k} = [\hat{y}_0^{n,k} \cdots \hat{y}_{N_{in}-1}^{n,k}]$ with $0 \leq k < K$ and $0 \leq n < L_n$.

Note that when the core decoder already uses QMF domain signals, thus, the QMF domain decoded audio data can be obtained; the QMF analysis step is omitted.

13.2.4.2.3 **Partitioned fast convolution for VOFF processing**

A band-wise partitioned fast convolution is carried out to process the QMF domain audio signal and the VOFF coefficients. Therefore an FFT analysis is carried out for each QMF frequency band $k$ with $0 \leq k < \mathrm{K}_{conv}$ with for each channel of the input audio signal.

The audio frames are split into sub-frames when the number of timeslots $L_n$ per frame is bigger than $\frac{N_{FFT}[k]}{2}$. The number of sub-frames is determined according to

$$\mathrm{N_{Frm}[k]} = \max\left(1, \frac{L_n}{\frac{N_{FFT}[k]}{2}}\right).$$

A vector $\mathbf{m}_{\mathrm{conv}}$ is used to signal which channel of the input signal corresponds to which BRIR pair in the BRIR data set. The audio channels and BRIRs are matched according to their position data. The BRIR position data is taken from the read metadata file, the position data of the audio channels is derived from the input parameter signaling the channel configuration.

If there are audio channels with no associated BRIR after the matching of the BRIR positions and the audio channel positions, a fallback BRIR is determined for these channels.

As a first step it is determined if a BRIR is available with same elevation and with a maximum azimuth deviation to the wanted position of +/- 20°. If such a BRIR is not available, the geometrically next BRIR is chosen. Which BRIR is the “geometrically next”, is determined to be the BRIR from the set of available BRIRs that has a minimum geometric distance to the wanted position. The geometric distance between two positions $P_1, P_2$ (each defined by azimuth $\alpha$ and elevation $\beta$) therefore is defined as

$$\Delta(P_1, P_2) = |\beta_1 - \beta_2| + |\alpha_1 - \alpha_2| .$$

For the set of known BRIR positions $[P_0, P_{N_{\mathrm{BRIR}}-1}]$ and a wanted position $P_W$, the geometrically next BRIR is the one where $\Delta(P_W, P_i)$ gets minimal for $\forall\, i \in [0, N_{\mathrm{BRIR}} - 1]$.

After a BRIR is determined for each audio channel, the partitioned fast convolution is conducted band-wise for all QMF frequency bands $k$ with $0 \leq k < K_{\mathrm{conv}}$ and $K_{\mathrm{conv}} = 32$. This processing step is computed independently of the sampling rate of the audio signal. If the audio sampling rate and the sampling rate of the BRIRs used for the generation of the VOFF coefficients does not correspond, they are nevertheless used for the VOFF processing. If this behavior is not wanted, the BRIRs have to be resampled to the wanted audio sampling rate and the parameterization has to be recalculated. Figure 53 outlines the partitioned fast convolution process.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

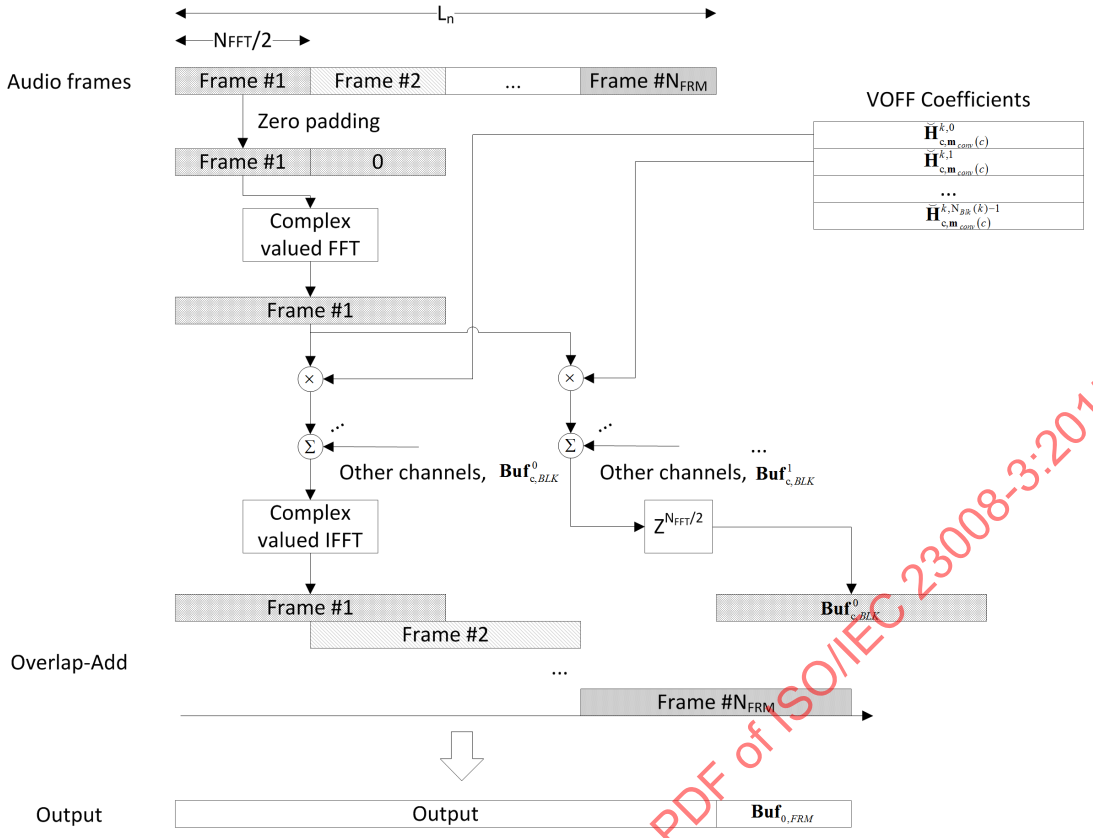



**Figure 53 — Outlines of the partitioned fast convolution process**

The sub-frames (referred to as “Audio frames” in Figure 53) of channel $c$ of the overall audio frame of length $L_N$ are zero-padded.

$$\hat{\mathbf{y}}_c^{i,k} = \left[ \hat{y}_c^{\frac{N_{FFT}}{2}i+0,k} \cdots \hat{y}_c^{\frac{N_{FFT}}{2}i+q,k} \cdots \hat{y}_c^{\frac{N_{FFT}}{2}(i+1)-1,k} \; 0 \cdots 0 \right]$$

The zero-padded sub-frames are transformed to the pseudo-FFT domain via complex-valued FFT.

$$\hat{\mathbf{Y}}_c^{i,k} = FFT\left\{\hat{\mathbf{y}}_c^{i,k}\right\} = [\hat{Y}_c^{0,i,k} \cdots \hat{Y}_c^{w,i,k} \cdots \hat{Y}_c^{N_{FFT}-1,i,k}]$$

To perform the block-wise fast convolution, the transformed audio frame $\hat{\mathbf{Y}}_c^{i,k}$ is complex multiplied by the VOFF coefficients of channel $\mathbf{m}_{\text{conv}}[c]$ and the complex multiplied outputs of all channels are summed up. From the second blocks, the complex multiplication results are shifted and stored to the buffers

$$\boldsymbol{Buf}_{0,BLK}^{l} = [Buf_{0,BLK}^{0,l} \cdots Buf_{0,BLK}^{N_{FFT}-1,l}] \text{ and } \boldsymbol{Buf}_{1,BLK}^{l} = [Buf_{1,BLK}^{0,l} \cdots Buf_{1,BLK}^{N_{FFT}-1,l}].$$

For $0 \le l < N_{BLK}(\mathrm{k}) - 1$

$$Buf_{0,BLK}^{w,l} = Buf_{0,BLK}^{w,l+1} + \sum_{c=0}^{N_{\text{in}}-1} \hat{Y}_c^{w,i,k} \tilde{\mathbf{H}}_{0,\mathbf{m}_{\text{conv}}(c)}^{w,k,l}$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$Buf_{1,BLK}^{w,l} = Buf_{1,BLK}^{w,l+1} + \sum_{c=0}^{N_{\text{in}}-1} \dot{\mathbf{Y}}_{\text{c}}^{w,i,k} \tilde{\mathbf{H}}_{1,\text{m}_{\text{conv}}(c)}^{w,k,l}$$

Then, the fast convoluted pseudo-FFT domain audio frame $[\dot{\mathbf{Z}}_{0,\text{conv}}^{w,i,k} \ \dot{\mathbf{Z}}_{1,\text{conv}}^{w,i,k}]$ is defined as

$$\dot{\mathbf{Z}}_{0,\text{conv}}^{w,i,k} = Buf_{0,BLK}^{w,0}$$

$$\dot{\mathbf{Z}}_{1,\text{conv}}^{w,i,k} = Buf_{1,BLK}^{w,0}$$

The fast convoluted pseudo-FFT domain audio frame $[\dot{\mathbf{Z}}_{0,\text{conv}}^{w,i,k} \ \dot{\mathbf{Z}}_{1,\text{conv}}^{w,i,k}]$ is then inverse transformed to the QMF domain and forms the fast convoluted QMF domain audio frame $[\dot{\mathbf{z}}_{0,\text{conv}}^{i,k} \ \dot{\mathbf{z}}_{1,\text{conv}}^{i,k}]$.

$$\dot{\mathbf{z}}_{0,\text{conv}}^{i,k} = IFFT\{[\dot{\mathrm{Z}}_{0,\text{conv}}^{0,i,k} \cdots \dot{\mathrm{Z}}_{0,\text{conv}}^{N_{FFT}-1,i,k}]\} = [\dot{\mathrm{z}}_{0,\text{conv}}^{0,i,k} \cdots \dot{\mathrm{z}}_{0,\text{conv}}^{N_{FFT}-1,i,k}]$$

$$\dot{\mathbf{z}}_{1,\text{conv}}^{i,k} = IFFT\{[\dot{\mathrm{Z}}_{1,\text{conv}}^{0,i,k} \cdots \dot{\mathrm{Z}}_{1,\text{conv}}^{N_{FFT}-1,i,k}]\} = [\dot{\mathrm{z}}_{1,\text{conv}}^{0,i,k} \cdots \dot{\mathrm{z}}_{1,\text{conv}}^{N_{FFT}-1,i,k}]$$

Finally, the fast convoluted QMF domain audio frame $[\dot{\mathbf{z}}_{0,\text{conv}}^{i,k} \ \dot{\mathbf{z}}_{1,\text{conv}}^{i,k}]$ is overlapped and saved.

For $0 \leq i < N_{FRM}(\mathrm{k}) - 1$ do:

$$\check{z}_{0,\text{conv}}^{\frac{N_{FFT}}{2}i+q,k} = \dot{\mathrm{z}}_{0,\text{conv}}^{q,i,k}$$

$$\check{z}_{1,\text{conv}}^{\frac{N_{FFT}}{2}i+q,k} = \dot{\mathrm{z}}_{1,\text{conv}}^{q,i,k}$$

For $i = N_{FRM}(\mathrm{k}) - 1$ do:

For $0 \leq q < N_{FFT}(\mathrm{k})/2 - 1$ do:

$$\check{z}_{0,\text{conv}}^{\frac{N_{FFT}}{2}i+q,k} = \dot{\mathrm{z}}_{0,\text{conv}}^{q,i,k}$$

$$\check{z}_{1,\text{conv}}^{\frac{N_{FFT}}{2}i+q,k} = \dot{\mathrm{z}}_{1,\text{conv}}^{q,i,k}$$

For $N_{FFT}(\mathrm{k})/2 - 1 \leq q < N_{FFT}(\mathrm{k}) - 1$ do:

$$Buf_{0,FRM}^{q} = \dot{\mathrm{z}}_{0,\text{conv}}^{q,i,k}$$

$$Buf_{1,BLK}^{q} = \dot{\mathrm{z}}_{1,\text{conv}}^{q,i,k}$$

The convolution results from each audio input channel with each BRIR pair are summed up in each QMF frequency band $\mathrm{k}$ with $0 \leq k < K_{\text{conv}}$ resulting in an intermediate 2-channel $K_{\text{conv}}$-band pseudo-FFT domain signal.

$\check{\mathbf{z}}_{\text{conv}}^{n,k} = [\check{z}_{0,\text{conv}}^{n,k}, \check{z}_{1,\text{conv}}^{n,k}]$ in QMF domain frequency band $k$.

Next, a bandwise FFT synthesis is carried out to transform the convolution result back to the QMF domain resulting in an intermediate 2-channel $K_{\text{conv}}$-band QMF domain signal with $\max(N_{\text{FFT}}[\mathrm{k}])$ time slots.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$\tilde{z}_{conv}^{n,k} = \left[\tilde{z}_{0,conv}^{n,k}, \tilde{z}_{1,conv}^{n,k}\right] \text{ with } 0 \le n < L_N \text{ and } 0 \le k < K_{\text{conv}}.$$

For each QMF domain input signal frame with $L_n = 64$ timeslots a convolution result signal frame with $L_n = 64$ timeslots is returned. The remaining timeslots are stored ($[Buf_{0,FRM}, Buf_{1,FRM}]$) and an overlap-add processing is carried out in the following frame(s).

13.2.4.2.4 **Generation of late reverberation**

As a second intermediate signal a QMF domain reverberation signal called $\hat{z}_{\text{rev}}^{n,k} = \left[\hat{z}_{0,\text{rev}}^{n,k}, \hat{z}_{1,\text{rev}}^{n,k}\right]$ is generated up to band $K_{\text{conv}} - 1$ by a reverberator module.

The reverberator takes as input

- a QMF domain stereo downmix of one frame of the input signal,
- the frequency-dependent RT60 reverberation time and the late reverberation energy values from the BRIR metadata information

and returns a 2-channel QMF domain late reverberation tail.

The RT60 reverberation time and late reverberation energy values are kept as read from the metadata file independent on the audio sampling rate; they are just remapped to different center frequencies if the audio sampling rate differs from the BRIR sampling rate that was used in the parameterization. Therefore, the center-frequencies of the late reverberation analysis bands from the BRIR metadata information are recalculated in a first step, if the audio sampling rate differs from the BRIR sampling rate that was used in the parameterization.

$$f_c[k] = \frac{f_c[k]}{f_{s,\text{BRIR}}} \cdot f_s \text{ with } 0 \le k < K_{ana}.$$

Next, the number of needed analysis bands $K_{LR}$ is determined. This is the number of analysis bands where

$$f_c[k] \le f_{c,max} \text{ with } f_{c,max} = 48.5 \cdot \left(\frac{f_s}{2} / 64\right).$$

Then, a QMF domain stereo downmix of one frame of the input signal $\hat{\mathbf{y}}^{n,k}$ is carried out to form the input of the reverberator by a weighted summation of the input signal channels. The weighting gains are contained in the downmix matrix $\mathbf{M}_{\text{DMX}}$. They are real-valued and non-negative and the downmix matrix is of dimension $N_{\text{out}} \times N_{\text{in}}$. It contains a non-zero value where a channel of the input signal is mapped to one of the two output channels.

The channels that represent loudspeaker positions on the left hemisphere are mapped to the left output channel and the channels that represent loudspeakers located on the right hemisphere are mapped to the right output channel. The signals of these channels are weighted by a coefficient of 1. The channels that represent loudspeakers in the median plane are mapped to both output channels of the binaural signal. The input signals of these channels are weighted by a coefficient

$$a = 0.7071 \approx \frac{1}{\sqrt{2}}.$$

In addition, an energy equalization step is performed in the downmix. It adapts the bandwise energy of one downmix channel to be equal to the sum of the bandwise energy of the input signal channels that are contained in this downmix channel. This energy equalization is conducted by a bandwise multiplication with a real-valued coefficient

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$$c_{eq,k} = \sqrt{\frac{p_{in}^{k}}{p_{out}^{k}+\varepsilon}}.$$

The factor $c_{eq,k}$ is limited to an interval of [0.5, 2].The numerical constant $\varepsilon$ is introduced to avoid a division by zero. The downmix is also bandlimited to the band $K_{conv}$; the values in all higher frequency bands are set to zero.

The downmix and the $K_{LR}$ values of RT60 and energy are fed to the reverberator. In the reverberator a mono downmix of the stereo input is calculated. This is done by incoherently applying a 90° phase shift on one of the stereo input channels. This mono signal is then fed to a feedback delay loop in each frequency band $k$, which creates a decaying sequence of impulses. It is followed by parallel FIR decorrelators that distribute the wanted signal energy in a decaying manner into the intervals between the impulses and create incoherence between the output channels. A decaying filter tap density is applied to create the energy decay defined by the RT60 time. The filter tap phase operations are restricted to four options to implement a sparse and multiplier-free decorrelator. After the calculation of the reverberation an inter-channel coherence (ICC) correction step is included in the reverberator module for every QMF frequency band. In the ICC correction step frequency-dependent direct gains and crossmix gains are used to adapt the ICC. The processing in the reverberator is shown in detail in Figure 54.

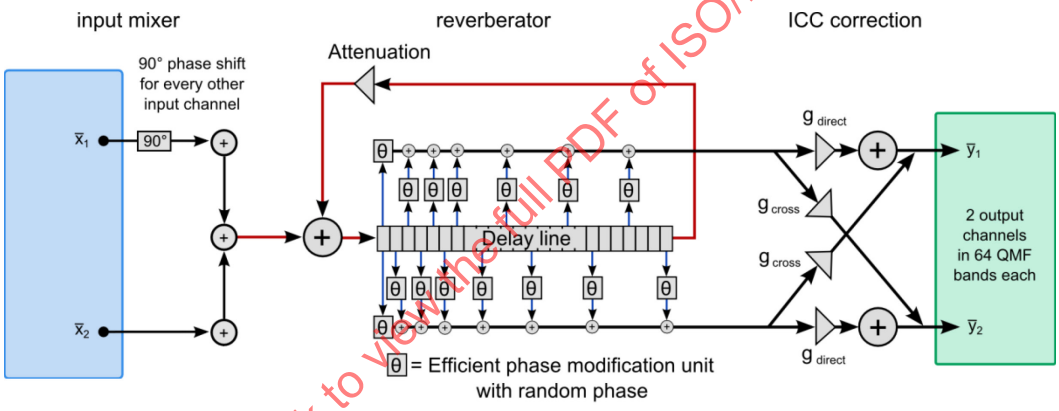



**Figure 54 — Processing in the reverberator**

The amount of energy and the reverberation times for the different frequency bands are contained in the input parameter set. The RT60 reverberation time and energy values are given at a number of $K_{LR}$ frequency points which are internally mapped to the $K = 64$ QMF frequency bands.

The reverberation is adaptively scaled according to a correlation measure of the input signal frame. The scaling factor is defined as a value in the interval of $[\sqrt{N_{DMX,act}}, N_{DMX,act}]$ linearly depending on a correlation coefficient $c_{corr}$ between 0 and 1. The coefficient $c_{corr}$ is set to the value 0.5.

$N_{DMX,act}$ is the number of channel that are active (carry signal energy) in the current audio frame and are downmixed to one output channel (e.g.the intersection of all values of one line of $M_{DMX}$ and the currently active audio channels).

The scaling factors are

$$\mathbf{c}_{scale} = [c_{scale,0}, c_{scale,1}] = \frac{\left[\sqrt{N_{DMX,act,0}} + c_{corr} \cdot \left(N_{DMX,act,0} - \sqrt{N_{DMX,act,0}}\right),\right.}{\left.\sqrt{N_{DMX,act,1}} + c_{corr} \cdot \left(N_{DMX,act,1} - \sqrt{N_{DMX,act,1}}\right)\right]}.$$

The number of active downmix channels $N_{DMX,act}$ may change over time.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

The scaling factors are smoothed over audio signal frames by a 1st order low pass filter resulting in smoothed scaling factors $\tilde{\mathbf{c}}_{\text{scale}} = [\tilde{c}_{\text{scale},0}, \tilde{c}_{\text{scale},1}]$. The scaling factors are initialized in the first audio input data frame.

13.2.4.2.5 **QTDL processing**

In the bands $K_{\text{conv}}$ to $K_{\max}$ –1, a QMF domain tapped delay line (QTDL) replaces both the VOFF processing and the Sparse Frequency Domain Reverberator.

The QTDL processing is done by performing one complex multiplication and delay.

$$\check{z}^{n,k}_{\text{i,QTDL}} = \sum_{m=0}^{N_{\text{in}}} g^{k}_{i,\mathbf{m}_{\text{conv}}[m]}\, \hat{y}^{\nu - d^{k}_{i,\mathbf{m}_{\text{conv}}[m]},k}_{\text{m}} \quad i \in \{0,1\}$$

$$\check{\mathbf{z}}^{n,k}_{\text{QTDL}} = [\check{z}^{n,k}_{0,\text{QTDL}}, \check{z}^{n,k}_{1,\text{QTDL}}]$$

For each QMF domain input signal with $L_n = 64$ timeslots, 64 QTDL processed output signals $\check{\mathbf{z}}^{n,k}_{\text{QTDL}}$ are returned corresponding to $L_n = 64$ timeslots. The remaining timeslots (corresponding to the last $d^{k}_{i,\mathbf{m}_{\text{conv}}[m]}$ samples) are stored and an overlap-add processing is carried out in the following frame(s).

13.2.4.2.6 **Mixing and Combination of Binaural Processed Outputs**

Next, the VOFF processed output $\tilde{\mathbf{z}}^{n,k}_{conv} = [\tilde{z}^{n,k}_{0,conv}, \tilde{z}^{n,k}_{1,conv}]$, the reverberator output $\hat{\mathbf{z}}^{n,k}_{\text{rev}} = [\hat{z}^{n,k}_{0,\text{rev}}, \hat{z}^{n,k}_{1,\text{rev}}]$ and the QTDL processed output $\check{\mathbf{z}}^{n,k}_{\text{QTDL}} = [\check{z}^{n,k}_{0,\text{QTDL}}, \check{z}^{n,k}_{1,\text{QTDL}}]$ for one QMF domain audio input frame are combined by a mixing process that bandwise adds up the two signals.

The late reverberation output is delayed by an amount of $d$ time slots in the mixing process. The delay $d$ takes into account the frequency-dependent transition time from early reflections to late reflections in the BRIRs and an initial delay of the reverberator of 20 QMF time slots, as well as an analysis delay of 0.5 QMF time slots for the QMF analysis of the BRIRs to ensure the insertion of the late reverberation at a reasonable time slot.

The combined signal $\hat{\mathbf{z}}^{n,k}$ at one time slot $n$ is calculated by $\hat{\mathbf{z}}^{n,k}_{\text{conv}} + \hat{\mathbf{z}}^{n-d,k}_{\text{rev}}$ in the bands up to $K_{\text{conv}}$ and is directly connected by $\check{\mathbf{z}}^{n,k}_{\text{QTDL}}$ in the bands between $K_{\text{conv}}$ and $K_{\max}$. For clarification, the bands beyond $K_{\max}$ are not processed in the binaural rendering and thus no signal is combined to the output signal in this band.

If *flag_HRIR* == 0

$$\hat{\mathbf{z}}^{n,k} = \begin{cases} \hat{\mathbf{z}}^{n,k}_{\text{conv}} + \hat{\mathbf{z}}^{n-d,k}_{\text{rev}} & 0 \le k < K_{\text{conv}} \\ \check{\mathbf{z}}^{n,k}_{\text{QTDL}} & K_{\text{conv}} \le k < K_{\max} \\ 0 & K_{\max} \le k < 64 \end{cases}$$

else

$$\hat{\mathbf{z}}^{n,k} = \begin{cases} \hat{\mathbf{z}}^{n,k}_{\text{conv}} & 0 \le k < K_{\text{conv}} \\ \check{\mathbf{z}}^{n,k}_{\text{QTDL}} & K_{\text{conv}} \le k < K_{\max} \\ 0 & K_{\max} \le k < 64 \end{cases}$$

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

13.2.4.2.7 **QMF synthesis of binaural QMF domain signal**

One 2-channel frame of $L_n = 32$ time slots of the QMF domain output signal $\hat{\mathbf{z}}^{n,k}$ is transformed to a 2-channel time domain signal frame with length $L$ by the QMF synthesis according to 4.6.18.4.2 of ISO/IEC 14496-3:2009 yielding the final time domain output signal $\tilde{\mathbf{z}}^{v} = [\tilde{z}_0^v, \tilde{z}_1^v]$.

## 13.3 Time-Domain Binaural Renderer

### 13.3.1 Introduction

The Time-Domain Binaural Renderer can be used for generating 3D Audio headphone signals for all types of input content (Channel and/or Object and/or HOA).

In the general case (Channel and/or Object and/or HOA input), the Time-Domain Binaural Renderer operates in the VLB (Virtual Loudspeaker Binaural) approach, which converts a loudspeaker signal $\mathbf{W}_{\mathrm{SPEAKER}}$ to the stereo output signal $\mathbf{W}_{\mathrm{LR}}$ using Time-Domain Virtual Loudspeaker binaural parameters for each loudspeaker position.

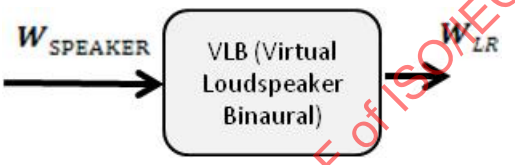



**Figure 55 — VLB approach for all types of input content (Channel and/or Object and/or HOA)**

However, in the specific case of HOA-only signals, the H2B (HOA to Binaural) approach can be also used, which directly converts the HOA signal to the stereo output signal using HOA-based binaural parameters.

The H2B approach can be used for HOA-only signals by applying a binaural filtering directly to the HOA components **C** to obtain the binaural signals $\mathbf{W}_{\mathrm{LR}}$

**Figure 56 — H2B approach for HOA-only signals**

When using Time-Domain Binauralization for HOA-only input signals, both VLB and H2B approaches shall be supported in the decoder. In this case, the approach used in the decoder for a given audio bitstream and given filters is selected depending on the availability of the filter format in the decoder (VLB, H2B, or both). It is noted that H2B filters can be obtained from VLB filters, but not conversely.

In the case where H2B filters are available, H2B methodology is used, otherwise VLB methodology is used.

Whatever the approach selected by the decoder (VLB or H2B), the binauralization is based on the filtering (with summation for each ear) of the input signals. In the VLB case, the inputs to the filters are the virtual loudspeakers signals, and in the H2B case, the inputs are the HOA components.

Globally, for each ear, this filtering process consists of:

— applying a specific Direct filter block, ideally modeling the direct path and early reflections (in effect also modeling the Head & Torso of the listener), to each input signal,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

— applying a common Diffuse filter $B_{mean}$ (possibly decomposed into M blocks), ideally modeling the diffuse part of the room response (including late reflections and reverb), to a weighted sum (taking into account the difference of energy between BRIRs) of the input signals.

— limiting the bandwidth of each block (to save computation) according to the cutting frequency provided (by the filter design, but also possibly by the decoder, e.g. limitation of bandwidth of the content)

The Direct and Diffuse filter coefficients, Diffuse weights, and cut-off frequencies are provided by an adaptive filter parametrization technique.

### 13.3.2 Definitions

#### 13.3.2.1 General Variables

In the following subclauses, the following variable conventions are used.

| | |
|---|---|
| $n$ | time index |
| $k$ | ear index (0=left, 1=right) |
| $u$ | frequency index |
| $L$ | number of input channels (virtual loudspeakers or HOA components). It is equal to the number of BRIRs (each BRIR containing a pair of impulse responses). |
| $l \in [1; L]$ | channel index |
| $I(l)$ | input signal for channel $l$ |
| $O^k$ | output signal for the left ($k = 0$) or right ($k = 1$) ear |
| $h_{k,l}$ | impulse response of the left ($k = 0$) or right ($k = 1$) BRIR $l$ |
| $h_{k,l}^{[n_0;n_1]}$ | taps $n_0$ to $n_1$ of impulse response of the left ($k = 0$) or right ($k = 1$) BRIR $l$ |

#### 13.3.2.2 Filter parametrization variables

| | |
|---|---|
| $bsDelay$ | initial delay of a BRIR set in time domain samples (propagation time) |
| $N_d$ | size in samples of the direct part of BRIRs |
| $A^k(l) \in \mathbb{R}^{N_d}$ | FIR coefficients for the direct block of the left ($k = 0$) or right ($k = 1$) BRIR $l$ |
| $M$ | number of diffuse blocks |
| $m \in [1; M]$ | block index |
| $\boldsymbol{B}_l^k \in \mathbb{R}^{N_d \cdot M}$ | FIR coefficients for the left ($k = 0$) or right ($k = 1$) diffuse filter of BRIR $l$ |
| $\boldsymbol{B}_{mean}^k \in \mathbb{R}^{N_d \cdot M}$ | FIR coefficients for the left ($k = 0$) or right ($k = 1$) average diffuse filter |
| $B_{mean}^k(m) \in \mathbb{R}^{N_d}$ | FIR coefficients for diffuse block $m$ of the left ($k = 0$) or right ($k = 1$) average diffuse filter |
| ${f_a}^k(l)$ | cut off frequency for direct part of the left ($k = 0$) or right ($k = 1$) BRIR $l$ |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| $f_b{}^k(m)$ | cut off frequency for the $m^{\text{th}}$ diffuse block of the left ($k = 0$) or right ($k = 1$) average diffuse filter |
| $Dw(l)$ | gain to apply to the input channel $l$ before filtering with average diffuse filters |
| $z^{-N_d \cdot m}$ | delay line operator for a delay of $m$ frequency blocks (i.e., $N_d m$ time samples) |
| $*_{[0;f]}$ | frequency-limited convolution operator (i.e., term-by-term multiplication in the frequency domain applied to a limited number of frequency bins, from 0 to $f$) |

#### 13.3.2.3 Function Definitions

The energy of a time-domain single-channel signal $x(n)$ is defined by :

$$E(x) = \sum_{n=0}^{+\infty} x^2(n)$$

The cumulative energy is defined by :

$$e(x,n) = \sum_{i=0}^{n} x^2(i)$$

The normalized cumulative energy is defined by :

$$e_{norm}(x,n) = \frac{e(x,n)}{E(x)}$$

The frequency-domain cumulative energy is defined by :

$$e_{freq}(x,u) = \frac{\sum_{i=0}^{u} X^2(i)}{\sum_{i=0}^{Nyquist} X^2(i)}$$

where $X(u)$ is the discrete Fourier transform of the $N$-point signal $x(n)$ :

$$X(u) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi \frac{u}{N} n}$$

and $Nyquist$ is the index of the Nyquist frequency, i.e., $\frac{N}{2}$ when $N$ is even.

### 13.3.3 Parameterization of Binaural Room Impulse Responses

#### 13.3.3.1 General

For each ear $k$ and channel $l$, the BRIR impulse response $h_{k,l}$ is parameterized as :

— an initial propagation delay $bsDelay$ (common for all BRIRs). This can be seen in Figure 57 as the initial 'quiet samples'.

— a direct filter block $A^k(l) \in \mathbb{R}^{N_d}$($N_d$ samples in the time domain). This can be seen in Figure 57 as the next block after bsDelay.

— $M$ diffuse filter blocks $B^k_{mean}(m) \in \mathbb{R}^{N_d}$, for $m \in [1; M]$ (common for all BRIRs). This can also been seen in Figure 57after the Direct block.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

— a diffuse gain $Dw(l)$ (common for left and right impulse responses)

The parametrization is composed of four stages that are described in details in the next sections :

— Propagation time calculation

— Direct/diffuse automatic segmentation

— Diffuse filters computation

— Cut-off frequencies computation

As a special case, if BRIRs are shorter than 1024 samples in the time domain, then the three first stages of parameterization are ignored and :

— $bsDelay$ is set to 0

— $N_d$ is set to the power of two above or equal to the BRIRs size

— $A^k(l) = h_{k,l}$ (plus additional zeros at the end to complete to $N_d$ samples)

— $M$ is set to 0 (i.e., no diffuse filter)

#### 13.3.3.2 Propagation time calculation

Starting from a set of original BRIRs, the offset *bsDelay* (in sample) is first computed to remove propagation time and samples with near-zero energy at the beginning of all BRIRs. For each ear $k$ and each channel $l$, the integer time index $d_{k,l}$ at which BRIR cumulative energy reaches $thres1 = 10^{-5}\ E_{max}$ is computed :

$$d_{k,l} = \min\left[\arg_n\left(\frac{e(h_{k,l},n)}{E_{max}} > thres1\right)\right]$$

where $E_{max}$ is the energy of the BRIR impulse response with maximum energy (among all filters) :

$$E_{max} = \max_{k,l}\left[\mathrm{E}(h_{k,l})\right]$$

The minimum of all time indexes is then selected as *bsDelay* :

$$bsDelay = \max_{k,l}\left[d_{k,l}\right]$$

All samples before *bsDelay* are set to zero in the BRIRs.

#### 13.3.3.3 Direct/diffuse automatic segmentation

Each BRIR is then automatically decomposed into a first block (direct and first reflections part) and $M$ diffuse blocks of the same size. This segmentation is illustrated on Figure 57.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

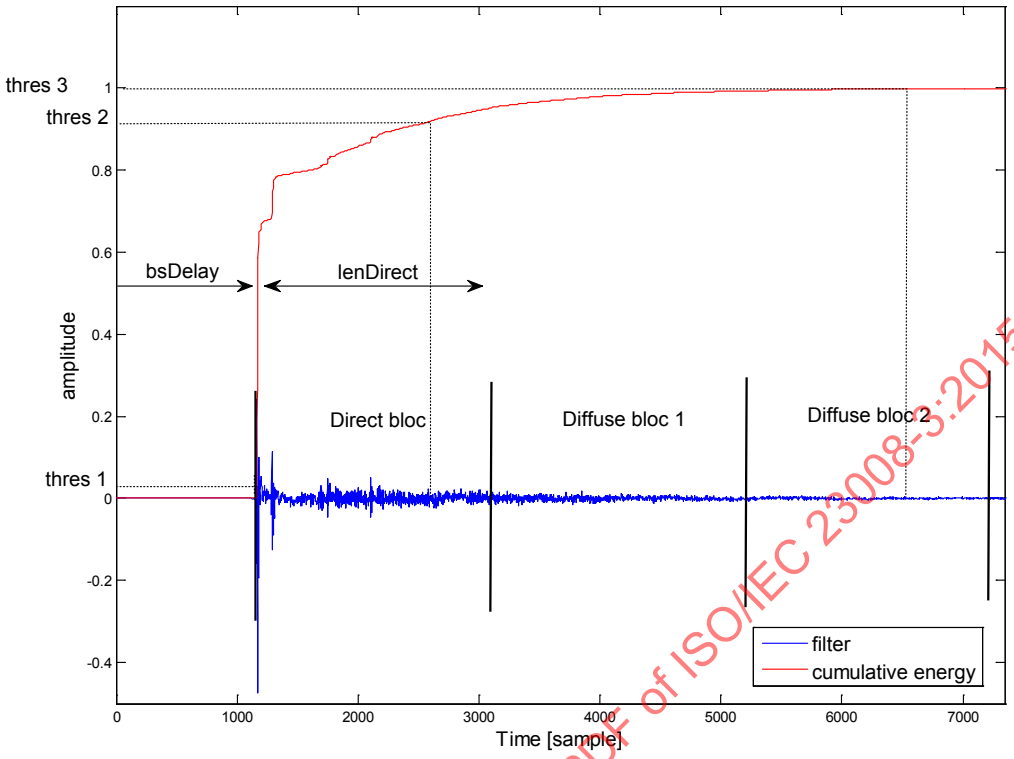



**Figure 57 — Automatic direct/diffuse segmentation for one BRIR**

The Direct part length is computed as follows. For each ear $k$ and each channel $l$, the integer time index $d_{k,l}$ at which BRIR normalized cumulative energy reaches $thres2 = 1 - 10^{-15/10}$ is computed :

$$d_{k,l} = \min\left[\arg_n\left(e_{norm}\left(h_{k,l}, n\right) \geq thres2\right)\right]$$

The maximum time index is selected and it is increased to the next power of two to get the direct part length

$$N_d = \text{nextpow2}(\max_{k,l}[d_{k,l}])$$

Finally the direct part length is imposed to be less than 8192 samples long :

$$N_d = min(8192,\ N_d\ )$$

For each ear $k$ and channel $l$, the direct filer $A^k(l)$ is then given by :

$$A^k(l) = h_{k,l}^{[bsDelay:bsDelay+N_d-1]}$$

The total number of diffuse blocks $M$ is computed as follows. For each ear $k$ and each channel $l$, the integer time index $d_{k,l}$ at which BRIR normalized cumulative energy reaches *thres3*=1-10$^{-19/10}$ is computed :

$$d_{k,l} = \min\left[\arg_n\left(\frac{e_{norm}(h_{k,l},n)}{E_{max}} > thres3\right)\right]$$

Then $M$ is given by :

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

$$M = \text{ceil}\left(\frac{\max_{k,l}\lfloor d_{k,l}\rfloor - (bsDelay + N_d - 1)}{N_d}\right)$$

where ceil is the round up function. If $M$ is zero, then there is no diffuse filter and the last steps of parametrization can be skiped. This is the case when *thres3* is reached before the end of the direct block (i.e., time index $bsDelay + N_d - 1$).

Otherwise the $M$-block long diffuse filter is computed for ear $k$ and channel $l$ by :

$$\boldsymbol{B}_l^k = h_{k,l}^{[bsDelay+N_d;\, bsDelay+(M+1).N_d-1]}$$

A smoothing window $w(n)$ is applied to fade out the last $N_{fade} = 512$ points of $\boldsymbol{B}_l^k$. Specifically we compute $w(n)$ by :

$$w(n) = \frac{cos(\pi.n/N_{fade})+1}{2} \text{ for } n = [0; N-1]$$

BRIR samples beyond $\boldsymbol{B}_l^k$ (i.e., time indexes superior to $bsDelay + (M+1).N_d - 1$) are ignored. If the end of $\boldsymbol{B}_l^k$ goes beyond the end of the BRIRs, then $\boldsymbol{B}_l^k$ is completed with zeros at the end (zero-padding) to reach $(M.N_d)$ samples. This is done before applying the smoothing window.

#### 13.3.3.4 Diffuse Filters Computation

To reduce computational cost, a single average diffuse filter $\boldsymbol{B}_{mean}^k$ is computed for ear $k$ by taking the mean of all normalized contributions per channel :

$$\boldsymbol{B}_{mean}^k = \frac{1}{L}\sum_{l=1}^{L}\frac{\boldsymbol{B}_l^k}{E(\boldsymbol{B}_l^k)}$$

where $\boldsymbol{B}_l^k$ is the $M$-block long diffuse filter for ear *k* and channel *l*, and $E(\boldsymbol{B}_l^k)$ its energy. $\boldsymbol{B}_{mean}^k$ is then cut into $M$ contiguous blocks $B_{mean}^k(m)$ of $N_d$ samples each for $m \in [1; M]$.

A compensation gain $Dw(l)$ is computed per channel $l$ by :

$$Dw(l) = \frac{2.G}{W^0(l) + W^1(l)}$$

where $G$ is a global attenuation gain of 6dB ( $G = 10^{-6/20}$ ) and $W^k(l)$ is a weight computed for ear $k$ and channel $l$ by:

$$W^k(l) = \frac{\sqrt{E(\boldsymbol{B}_{mean}^k)}}{\sqrt{E(\boldsymbol{B}_l^k)}}$$

#### 13.3.3.5 Cut-off frequencies computation

In typical BRIRs, high-frequency energy decreases faster than low-frequency energy. Cut-off frequencies are extracted for direct and diffuse block filters, to avoid multiplying frequency bins with low energy during processing.

Figure 58 shows an example of the decomposition of one BRIR (left and right) into 1 block of direct part, and 2 blocks of diffuse part, on a time frequency representation. Each block is $N_d$= 2048 samples long. The samples after $3N_d$ are not used. The frequencies above $f_a{}^l(l)$ and $f_a{}^r(l)$ in the direct block are not used. The

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

frequencies above $f_b{}^l(1)$ and $f_b{}^r(1)$ in the first diffuse block are not used. The frequencies above $f_b{}^l(2)$ and $f_b{}^r(2)$ in the second diffuse block are not used. During processing, unprocessed bins above cut-off frequencies are equivalent to a multiplication by 0 (but not from a complexity point of view) which can lead to some artifacts due to circular convolution inside a block. These artifacts can generally be kept low enough to avoid audible artifacts after the whole process.

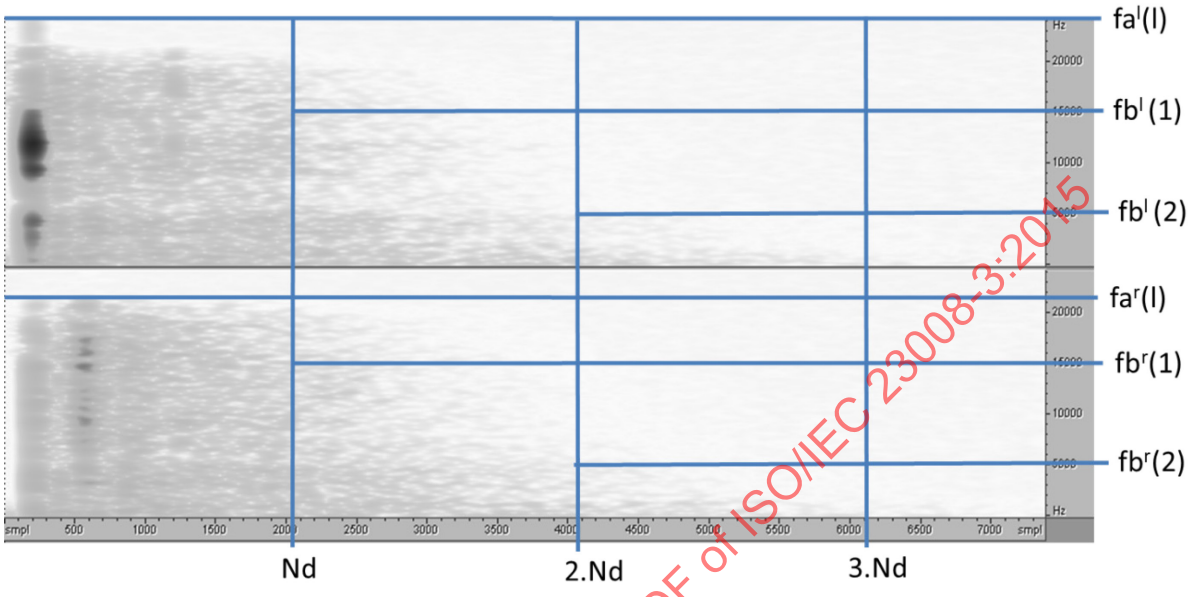



**Figure 58 — Time frequency representation of a pair of BRIR**

This phenomenon is exploited in the binaural renderer. For ear $k$ and channel $l$, the frequency index $\hat{d}_{k,l}$ at which the frequency-domain cumulative energy of the direct block reaches *thres4*=0.99 is computed :

$$\hat{d}_{k,l} = \min\left[\arg_u\left(e_{freq}(A^k(l), u) > thres4\right)\right]$$

Then the normalized cutoff frequency $f_a{}^k(l)$ is defined by :

$$f_a{}^k(l) = \frac{\hat{d}_{k,l}}{Nyquist}$$

In a final "quantization" stage, $f_a{}^k(l)$ is set to its closest value in vector [1/6 1/3 1/2 2/3 5/6 1].

For each channel l, each ear $k$ and each diffuse bloc $m$, the frequency index $\hat{d}_{l,k,m}$ at which the frequency-domain cumulative energy of the diffuse block $m$ reaches *thres4* is computed :

$$\hat{d}_{l,k,m} = \min\left[\arg_u\left(e_{freq}\left(B_l^k(m), u\right) > thres4\right)\right]$$

Then the normalized cutoff frequency $f_b{}^k(m)$ is defined by :

$$f_b{}^k(m) = \frac{\max_l \hat{d}_{l,k,m}}{Nyquist}$$

In a final "quantization" stage, $f_b{}^k(m)$ is set to its closest value in vector [1/6 1/3 1/2 2/3 5/6 1].

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

### 13.3.4 Time-Domain Binaural Processing

#### 13.3.4.1 General

The left/right output is given by the following formula:

$$O^k = \sum_{l=1}^{L}\left(A^k(l) *_{[0;\ldots;f_a{}^k(l)]} I(l)\right) + \sum_{m=1}^{M}\left(B^k_{mean}(m) *_{[0;\ldots;f_b{}^k(m)]}\left(z^{-N_d.m}.\sum_{l=1}^{L} D_w(l).I(l)\right)\right)$$

The processing is achieved in the DFT (discrete Fourier transform) domain based on a block-wise approach. The following constraints are associated to the algorithm:

— The number of samples $N_d$ of the direct part shall be equal to a power of $2$.

— The size of each of the $M$ diffuse block is equal to $N_d$.

— The frame size of the binauralization process is equal to $N_d$.

— The size of the delay line is a multiple of $Nd$ (delay of $1$ frequency block).

— The size of the DFT is $2.N_d$

The following figure shows the implementation with $L$ input signals and $M$ diffuse blocks:

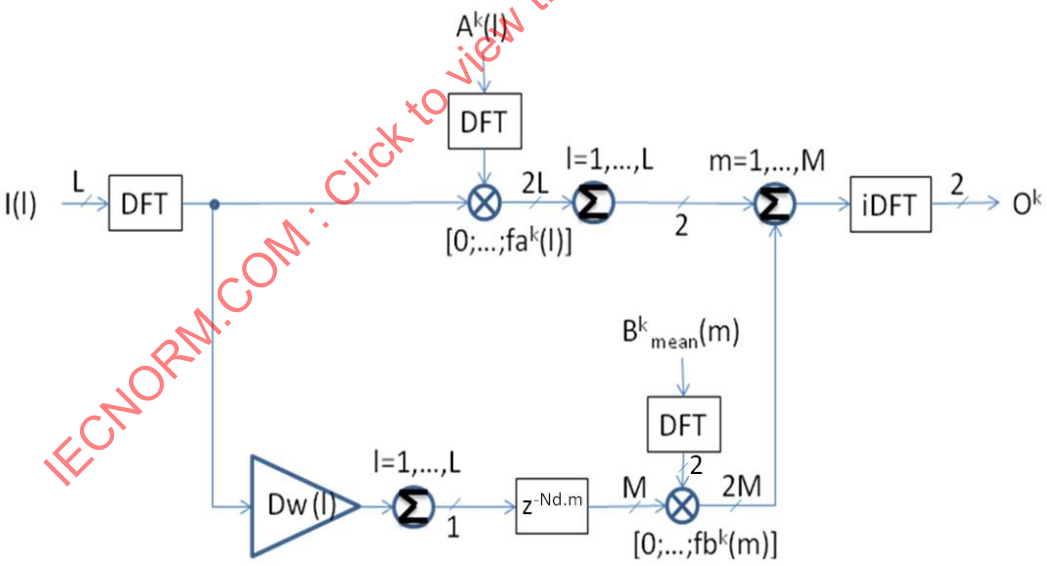



**Figure 59 — Block diagram description of the binauralization algorithm**

#### 13.3.4.2 Initialization

The $L$ filters of the Direct block $A^k(l)$ and the $M$ filters of the Diffuse blocks $B^k_{mean}(m)$ are provided as time domain blocks of size $N_d$. Each time domain filter is converted to the frequency domain in an initialization step.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$2L$ buffers of $2N_d$ complex data for Direct part, and $2M$ buffers of $2N_d$ complex data for Diffuse blocks are provided.

#### 13.3.4.3 Processing

The $L$ input signals (channel or HOA) are put in buffers of size $N_d$. Using an overlap-add/overlap-save technique, buffers of size $2N_d$ are created and then converted into the DFT domain.

For the Direct block and for each ear, these $L$ frequency buffers are multiplied term-by-term with the associated $L$ Direct filters from $0$ up to the frequency ${f_a}^k(l)$ and accumulated in a frequency output buffer previously initialized to zero.

These $L$ frequency buffers are also multiplied-accumulated by the Diffuse weights $Dw(l)$ for each ear to provide a weighted downmix buffer dedicated to feed successively all the Diffuse blocks.

For each of the $M$ Diffuse blocks and for each ear, the appropriately delayed weighted downmix buffer is multiplied term-by-term with the associated Diffuse filters from 0 up to the frequency ${f_b}^k(l)$ and accumulated in the frequency output buffer.

# 14 MPEG-H 3D audio stream (MHAS)

## 14.1 Overview

This clause defines a self-contained stream format to transport ISO/IEC 23008-3 (MPEG-H 3D Audio) data. The transport mechanism uses a packetized approach. Both, configuration data as well as coded audio payload data is embedded into separate packets. Synchronization and length information is added to enable a self-synchronizing syntax.

This stream format is intended to be used for the transmission over channels where no frame synchronization is available and it may be used for the transmission over channels with fixed frame synchronization.

## 14.2 Syntax

### 14.2.1 Main MHAS syntax elements

**Table 137 — Syntax of mpeghAudioStream()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpeghAudioStream()<br>{<br>  while (bitsAvailable() != 0) {<br>  mpeghAudioStreamPacket();<br>  }<br>} | | |

**Table 138 — Syntax of mpeghAudioStreamPacket()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpeghAudioStreamPacket()<br>{ | | |
| MHASPacketType = escapedValue(3,8,8); | **3,11,19** | **uimsbf** |
| MHASPacketLabel = escapedValue(2,8,32); | **2,10,42** | **uimsbf** |
| MHASPacketLength = escapedValue(11,24,24); | **11,35,59** | **uimsbf** |
| MHASPacketPayload(MHASPacketType); | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| |
|---|
| } |
| NOTE: With the given bit allocation, MHASPacketPayload() is always byte-aligned |

**Table 139 — Syntax of MHASPacketPayload()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| MHASPacketPayload(MHASPacketType) | | |
| { | | |
| switch (MHASPacketType) { | | |
| case PACTYP_SYNC: | | |
| **0xA5**; /* syncword*/ | **8** | **uimsbf** |
| break; | | |
| case PACTYP_MPEGH3DACFG: | | |
| mpegh3daConfig(); | | |
| break; | | |
| case PACTYP_MPEGH3DAFRAME: | | |
| mpegh3daFrame(); | | |
| break; | | |
| case PACTYP_FILLDATA: | | |
| for (i=0; i< MHASPacketLength; i++) { | | |
| **mhas_fill_data_byte(i)**; | **8** | **bslbf** |
| } | | |
| break; | | |
| case PACTYP_SYNCGAP: | | |
| syncSpacingLength = escapedValue(16,24,24); | **16,40,64** | **uimsbf** |
| break; | | |
| case PACTYP_MARKER: | | |
| for (i=0; i< MHASPacketLength; i++) { | | |
| **marker_byte(i)**; | **8** | **bslbf** |
| } | | |
| break; | | |
| case PACTYP_CRC16: | | |
| **mhasParity16Data**; | **16** | **bslbf** |
| break; | | |
| case PACTYP_CRC32: | | |
| **mhasParity32Data**; | **32** | **bslbf** |
| break; | | |
| case PACTYP_DESCRIPTOR: | | |
| for (i=0; i< MHASPacketLength; i++) { | | |
| **mhas_descriptor_data_byte(i)**; | **8** | **bslbf** |
| } | | |
| break; | | |
| case PACTYP_USERINTERACTION: | | |
| mpegh3daElementInteraction(); | | |
| break; | | |
| case PACTYP_LOUDNESS_DRC: | | |
| mpegh3daLoudnessDrcInterface(); | | |
| break; | | |
| case PACTYP_BUFFERINFO: | | |
| **mhas_buffer_fullness_present** | **1** | **uimsbf** |
| if (mhas_buffer_fullness_present) | | |
| mhas_buffer_fullness = escapedValue(15,24,32); | **15,39,71** | **uimsbf** |
| } | | |
| break; | | |
| } | | |
| ByteAlign(); | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

### 14.2.2 Subsidiary MHAS syntax elements

**Table 140 — Syntax of mpegh3daLoudnessDrcInterface()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daLoudnessDrcInterface()<br>{<br>  uniDrcInterface(); /* as defined in ISO/IEC 23003-4 */<br>} | | |

## 14.3 Semantics

### 14.3.1 mpeghAudioStreamPacket()

**MHASPacketType** This element specifies the payload type in the actual packet. The meaning of MHASPacketType is defined in Table 141. A decoder which does not support a certain MHASPacketType shall skip this packet and continue with the next package.

**Table 141 — Value of MHASPacketType**

| MHASPacketType | Value |
|---|---|
| PACTYP_FILLDATA | 0 |
| PACTYP_MPEGH3DACFG | 1 |
| PACTYP_MPEGH3DAFRAME | 2 |
| */* reserved for ISO use */* | 3-5 |
| PACTYP_SYNC | 6 |
| PACTYP_SYNCGAP | 7 |
| PACTYP_MARKER | 8 |
| PACTYP_CRC16 | 9 |
| PACTYP_CRC32 | 10 |
| PACTYP_DESCRIPTOR | 11 |
| PACTYP_USERINTERACTION | 12 |
| PACTYP_LOUDNESS_DRC | 13 |
| PACTYP_BUFFERINFO | 14 |
| */* reserved for ISO use */* | 15-127 |
| */* reserved for use outside of ISO scope */* | 128-261 |
| */* reserved for ISO use */* | 262-389 |
| */* reserved for use outside of ISO scope */* | 390-517 |
| NOTE: Application-specific MHASPacketType values are mandated to be in the space reserved for use outside of ISO scope. These are skipped by a decoder as a minimum of structure is required by the decoder to skip these extensions. | |

**MHASPacketLabel** This element provides an indication on which packets belong together. For example, with using different labels, different MPEG-H 3D audio configuration structures may be assigned to particular sequences of MPEG-H 3D audio access units.
If this field is set to ‘0’, the packet is of general interest and is related to the complete stream.

**MHASPacketLength** This element indicates the length of the MHASPacketPayload() in Bytes.

MHASPacketPayload() The payload for the actual MHASPacket.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

### 14.3.2 MHASPacketPayload()

| | |
|---|---|
| mpegh3daConfig() | An MPEG-H 3D audio configuration structure as defined in 5.2.2.1. |
| mpegh3daFrame() | An MPEG-H 3D audio payload as defined in 5.2.3.1. |
| ByteAlign() | Up to 7 fill bits to achieve byte alignment with respect to the beginning of the syntactic element in which ByteAlign() occurs. |
| **syncSpacingLength** | the length in Bytes between the last two **MHASPacketType** PACTYP_SYNC. |
| **mhasParity16Data** | a 16-bit field that contains the CRC value that yields a zero output of the 16 registers in the decoder with the polynomial: $x^{16} + x^{15} + x^5 + 1$ and the initial state of the shift register of 0xFFFF. |
| **mhasParity32Data** | a 32-bit field that contains the CRC value that yields a zero output of the 32 registers in the decoder with the polynomial: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ and the initial state of the shift register of 0xFFFFFFFF. |
| **mhas_fill_data_byte** | 8-bit data elements, no restrictions apply |
| mpegh3daElementInteraction() | MPEG-H 3D audio element interaction structure as defined in 17.7.4 |
| mpegh3daLoudnessDrcInterface() | An MPEG-H 3D Loudness and DRC interface as defined in 14.2.2. |
| **mhas_buffer_fullness_present** | a bit signaling the presence of mhas_buffer_fullness |
| **mhas_buffer_fullness** | data element indicating the state of the bit reservoir in the course of encoding the access unit. It is transmitted as the number of available bytes in the bit reservoir. The state of the bit reservoir is derived according to 5.6.2. |

## 14.4 Description of MHASPacketTypes

### 14.4.1 PACTYP_FILLDATA

The MHASPacketType PACTYP_FILLDATA provides the possibility to add fill data to adjust the instantaneous bit-rate. This may be desirable in certain real-time applications where data is transmitted over a constant rate channel.

MHASPacketLabel shall be 0, because packets of this type do not relate to certain payload data.

It is expected that decoders neglect the data transmitted in packets of type PACTYP_FILLDATA. Furthermore, intermediate tools that are processing an MHAS streams are allowed to remove such packets from the stream.

NOTE: It is allowed to set MHASPacketLength to 0. This yields in a minimum packet size of 2 bytes.

### 14.4.2 PACTYP_MPEGH3DACFG

MHAS Packets of the MHASPacketType PACTYP_MPEG3DACFG embed an MPEG-H 3D audio configuration structure, mpegh3daConfig(), in the MHASPacketPayload().

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

MHASPacketLabel indicates the ID used for this configuration. This configuration shall be used for packets of type PACTYP_MPEGH3DAFRAME that have the same ID, so that a unique link between configuration data and payloads is possible.

If MHASPacketType of type PACTYP_MPEGH3DACFG is not present, the mpegh3daConfig() should be conveyed through out-band means, such as session announcement/description/control protocols.

### 14.4.3 PACTYP_MPEGH3DAFRAME

MHAS Packets of the MHASPacketType PACTYP_MPEG3DAFRAME embed a frame of MPEG-H 3D audio, mpegh3daFrame(), in the MHASPacketPayload().

MHASPacketLabel indicates the ID used for this payload. Packets of type PACTYP_MPEGH3DACFG that have the same ID carry the corresponding configuration, so that a unique link between configuration data and payloads is possible.

### 14.4.4 PACTYP_SYNC

The syncword payload for ISO/IEC 23008-3 MPEG-H Audio streams is '1010 0101'.

For this packet type, MHASPacketLabel has no meaning and shall be set to 0.

NOTE: The complete syncword is not only determined by the syncword payload, but by the complete mpeghAudioStreamPacket() with the MHASPacketType of PACTYP_SYNC, including all its fields. Therefore, a packet of type PACTYP_SYNC is 0xC001A5.

### 14.4.5 PACTYP_SYNCGAP

The MHASPacketType PACTYP_SYNCGAP may be used to improve synchronization to a stream. It directly follows a MHASPacketType PACTYP_SYNC.

For this packet type, MHASPacketLabel has no meaning and shall be set to 0.

### 14.4.6 PACTYP_MARKER

#### 14.4.6.1 General

The MHASPacketType PACTYP_MARKER indicates a marker event in the stream. The event type is signaled in the packet payload.

**Table 142 — Meaning of marker_byte**

| Value of marker_byte | Meaning |
|---|---|
| 0x01 | Configuration Change Marker |
| 0x02 | Random Access / Immediate Playout Marker |
| 0x03 | Program Boundary Marker |
| 0x04 ... 0xff | *reserved* |

#### 14.4.6.2 Configuration Change Marker

When the first marker_byte of the packet payload is "0x01", a configuration change occurred and evaluating the configuration structure mpegh3daConfig() in the following MHASPacketType PACTYP_MPEG3DACFG is mandatory.

For packets of PACTYP_MPEG3DACFG without a preceding configuration change marker packet, evaluation of the configuration structure is not required.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

#### 14.4.6.3 Random Access / Immediate Playout Marker

When the first marker_byte of the packet payload is “0x02”, the following packet of type PACTYP_MPEG3DAFRAME with identical MHASPacketLabel is encoded following the rules given in 5.7.

#### 14.4.6.4 Program Boundary Marker

When the first marker_byte of the packet payload is “0x03”, a program boundary occurs at this point in time and all following packets belong to a new program.

### 14.4.7 PACTYP_CRC16 and PACTYP_CRC32

The MHASPacketType PACTYP_CRC16 and PACTYP_CRC32 may be used for detection of errors in the subsequent MHAS packet with MHASPacketLabel set to the same value. It shall be directly followed by the MHAS packet which its CRC value refers to. This may be beneficial when an MHAS stream is conveyed over an error prone channel.

The error detection method uses one of the generator polynomial and associated shift register states as defined for mhasParity16Data or mhasParity32Data respectively.

All bits of the MHASPacketPayload() (including the ByteAlign()) of the protected MHAS packet are included into the CRC-check.

In the case where there are no errors, each of the outputs of the shift register shall be zero. At the CRC encoder the **mhasParity16Data** / **mhasParity32Data** field is encoded with a value such that this is ensured.

### 14.4.8 PACTYP_DESCRIPTOR

The PACTYP_DESCRIPTOR may be used to embed MPEG-2 TS/PS descriptors in MHAS streams. Data conveyed as **mhas_descriptor_data_byte** has the same syntax and semantics as defined for descriptor() in ISO/IEC 13818-1.

For this packet type and for descriptors transmitted in the first descriptor loop in the TS_program_map_section() (see ISO/IEC 13818-1), MHASPacketLabel shall be 0.

For this packet type and for descriptors assigned to one elementary stream (i.e. the second descriptor loop in the TS_program_map_section() ), MHASPacketLabel have the same value as in the PACTYP_CONFIG from the associated elementary stream.

### 14.4.9 PACTYP_USERINTERACTION

The MHASPacketType PACTYP_USERINTERACTION may be used to feed element interaction data to the decoder.

For this packet type, MHASPacketLabel shall have the same value as the packet of MHASPacketType PACTYP_MPEGH3DACFG, which the user interaction data refers to.

### 14.4.10 PACTYP_LOUDNESS_DRC

The MHASPacketType PACTYP_LOUDNESS_DRC may be used to feed Loudness and DRC control data to the decoder.

For this packet type, MHASPacketLabel has the same value as the packet of MHASPacketType PACTYP_MPEGH3DACFG, which the control data refers to.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

#### 14.4.11 PACTYP_BUFFERINFO

The MHASPacketType PACTYP_BUFFERINFO may be used to indicate the buffer fullness of the encoded stream or sub-stream with MHASPacketLabel set to the same value.

If MHASPacketLabel is set to 0, the information in the packet relates to the overall stream including all available sub-streams.

If present, this packet shall be placed into the stream at that point of time it provides valid information; i.e. when present after one packet of MHASPacketType PACTYP_MPEGH3DAFRAME, it shall signal the buffer state after encoding of that access unit.

### 14.5 Application Examples

#### 14.5.1 Light-weighted broadcast

Certain applications require a minimum overhead in terms of bitrate, while synchronization time is less critical. (e.g. Internet Radio)

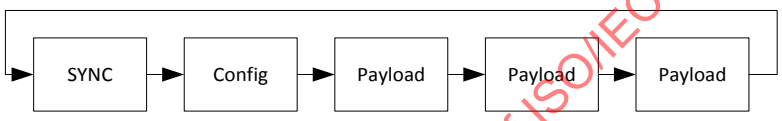



**Figure 60 — Example 1**

#### 14.5.2 MPEG-2 Transport Stream

When embedding MPEG-H 3D Audio streams into MPEG-2 Transport streams, fast synchronization to a stream at random access points is most important, while bitrate overhead is usually less critical. Therefore, the configuration structure is sent on a regular basis, typically twice a second.

To improve synchronization to the stream packets of type PACTYP_SYNC may be embedded more frequently and in addition the PACTYP_SYNCGAP type may be used as well.

The figures below indicate some possible sequences of packet types when the MPEG-H Audio Stream is intended to be embedded into the MPEG-2 Transport Stream.

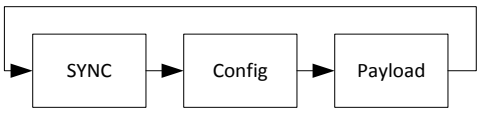



**Figure 61 — Example 2**

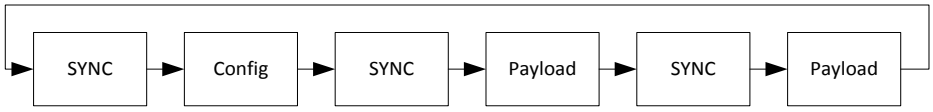



**Figure 62 — Example 3**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

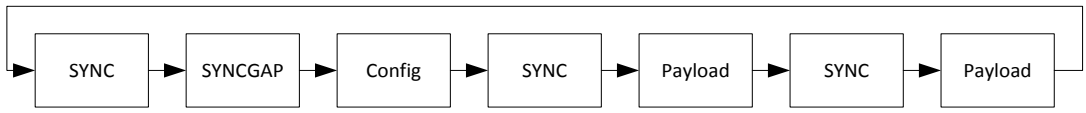



**Figure 63 — Example 4**

## 14.6 Multi-Stream Delivery and Interface

This subclause addresses the situation in which multiple incoming input streams form one program. These incoming streams may be merged into one MHAS stream prior to decoding. By assigning each of the incoming streams (i.e. packets of type PACTYP_MPEGH3DAFRAME) and their related configuration structures (i.e. packets of type PACTYP_MPEGH3DACFG) a unique **MHASPacketLabel**, sub-streams within one MHAS stream are formed. As the sub-streams are generated by the same encoder, it is presumed that various incoming streams, though potentially delivered over different transmission channels, are completely aligned and have no phase offset. Compensation for different transmission delays shall be accomplished by the systems layer and is not covered by this specification.

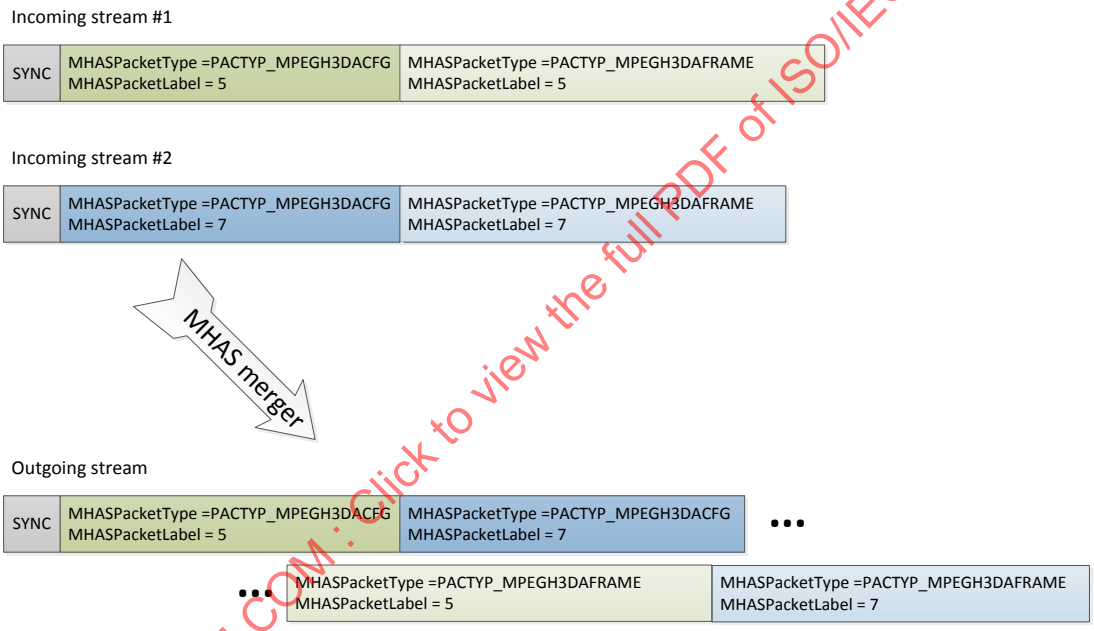



**Figure 64 — Merging two MHAS streams**

The MHAS packets of the incoming streams shall be merged into the outgoing steam such that all MHAS packets that belong to the same time instance and thus form one Access Unit are consecutively written into the outgoing stream.

In general, no specific order of MHAS sub stream packets that belong to one Access Units needs to be maintained. However it is recommended to order the sub-streams in ascending order of the packet label number. By this each received new packet with a higher label belongs to the same time slot than the previous packets.

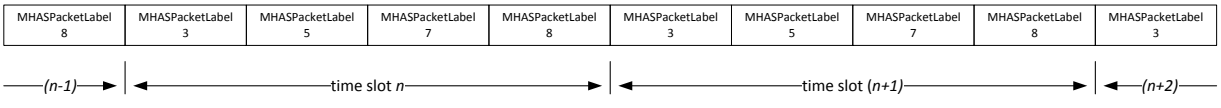



**Figure 65 — Time Slots with Multiple MHAS Packets**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

Though it may be desirable, it is not required that the sub-stream with **mae_isMainStream** set to '1' has the lowest **MHASPacketLabel** number.

As specified in to 14.3.1 incoming packets with **MHASPacketLabel** set to '0' are related to the complete stream, at least packets of type PACTYP_MPEGH3DAFRAME shall not be labeled with '0' when more than one sub-stream is present in the MHAS stream.

The multi-stream enabled decoder shall be capable of handling those streams with sub-streams labeled with different **MHASPacketLabels**. By utilizing the field **mae_bsMetaDataElementIDoffset** in mae_AudioSceneInfo() (see 15.3), the decoder shall be enabled to arrange the sub-streams in the right order before decoding.

EXAMPLE:

The receiver has several incoming streams, from which he chooses the main stream containing the channel bed, additional effect channels and the main dialog in language 1 and the stream #N containing the audio description in language 1. These two MHAS streams are merged by an MHAS merger into one stream containing two sub-streams:

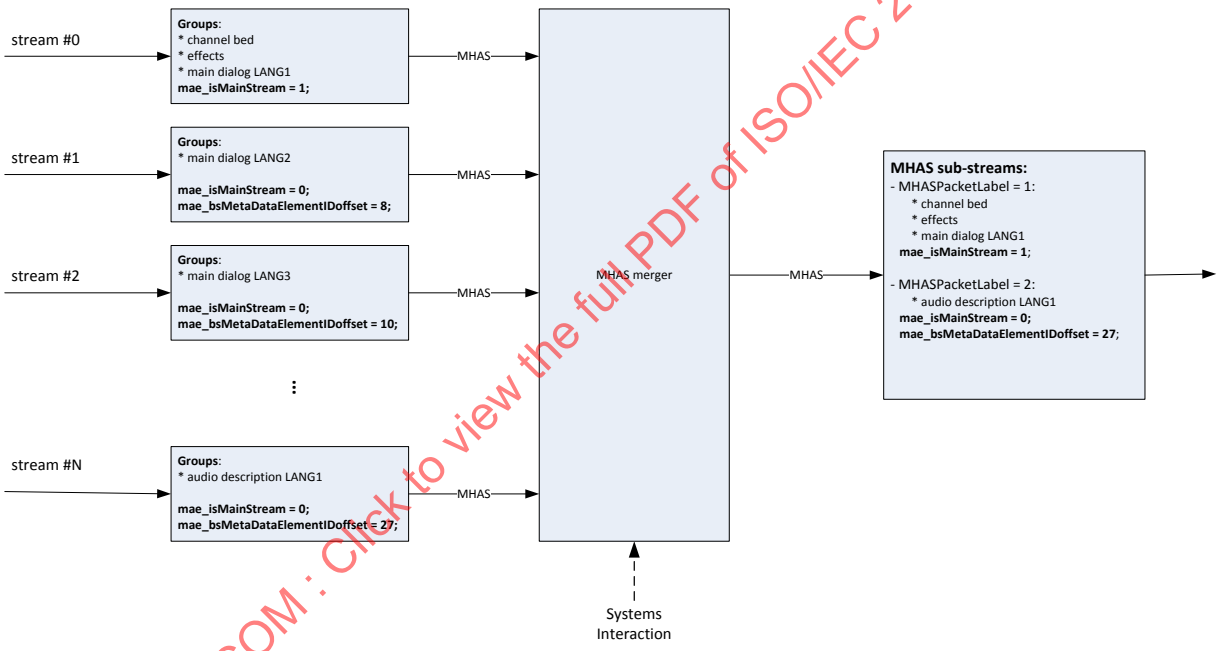



**Figure 66 — Example of switching and merging multiple incoming streams**

The combined stream has the following structure:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

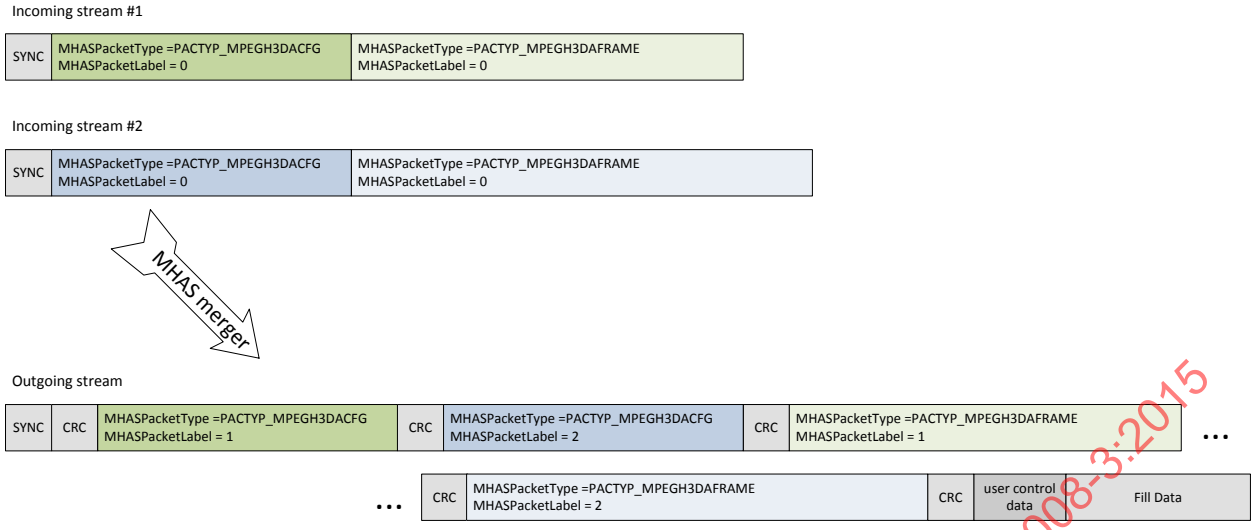



**Figure 67 — Example output bit stream from MHAS merger**

Additionally, the MHAS merger might also include or discard additional packages such as PACTYP_CRC32 as required by the subsequent transmission line.

# 15 Metadata Audio Elements (MAE)

## 15.1 Introduction

The set of metadata consists of

1) Descriptive metadata: Information about the existence of objects inside the bitstream and high-level properties of objects

2) Restrictive metadata: Information of how interaction is possible or enabled by the content creator

3) Positional metadata and ability to render to specific speakers and to signal channel content as objects

4) Structural metadata: Grouping and combination of objects

The Metadata is organized as groups or switch groups of elements. All static metadata is organized in the mae_AudioSceneInfo() in the mpegh3daConfigExtension(). The Main Audio Scene describes the full scene in terms of the static metadata of all elements.

**Groups of Elements**

Related elements are associated with a group. These elements belong together and can only be manipulated together. Elements of a group can no longer be interactively changed on their own (e.g. if a group is switched off, all its child elements are switched off), but behave as a union. Examples are channel-based recordings, e.g. an AB recording where the two recorded signals belong together and should only be manipulated as a pair. This grouping allows for signaling of stems and submixes by gathering the dedicated elements in groups.

**Switch Groups of Elements**

It should be possible to define special groups, where just one or a selected number of its elements can be switched on at a time. This group is called switch group here. This concept ensures that the audience could enable exactly one out of many elements of the same kind if reasonable. An example is the possibility to produce and transmit a multitude of language tracks.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

### Group Presets

Group presets define a combination of groups in an audio scene. A group preset contains a list of groups and an associated on/off-status for each of these groups (presets' conditions) which are not modifiable by the user. With group presets a content creator or an application can provide a restricted number of meaningful rendering options to the user.

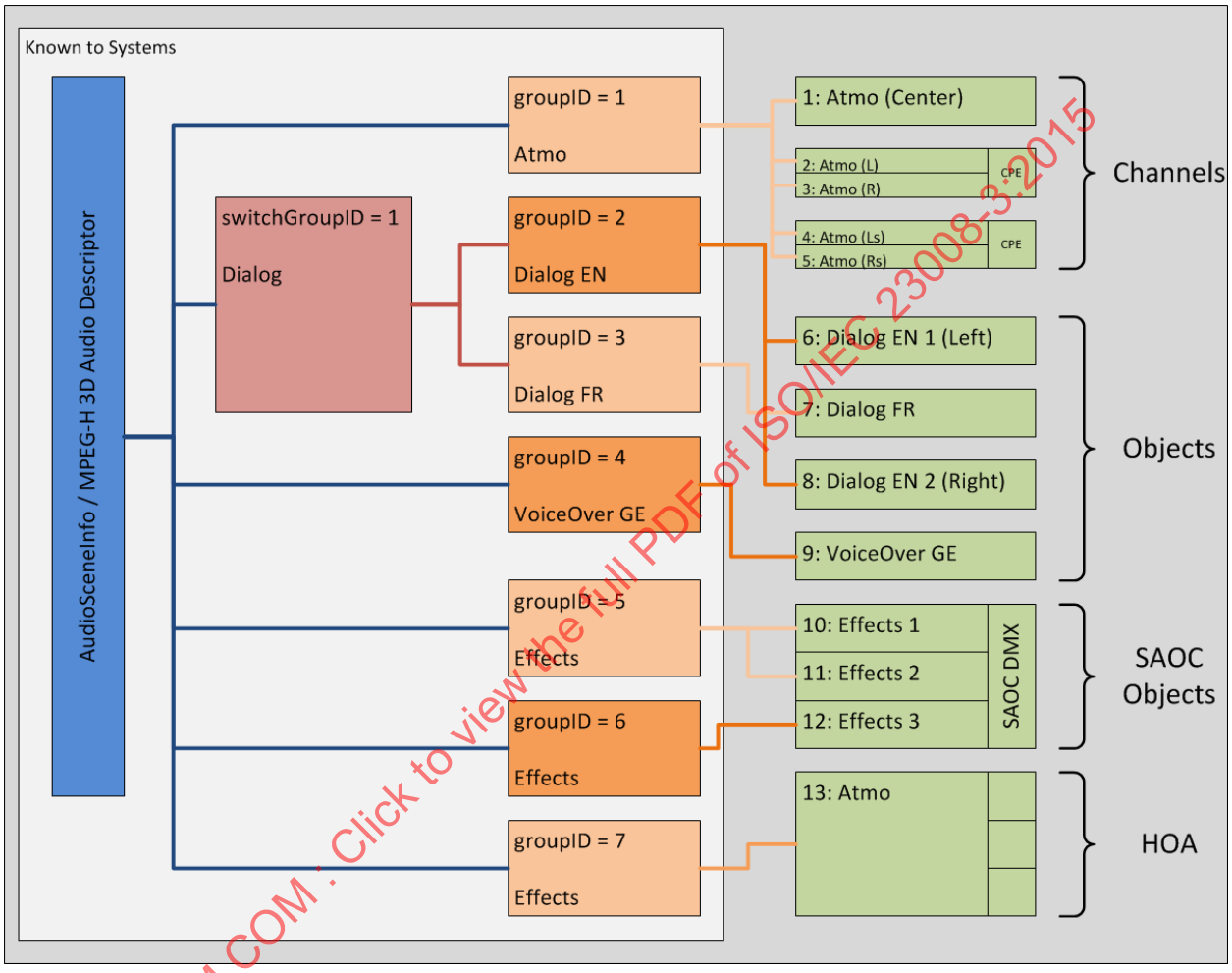



**Figure 68 — Example structure of Metadata Elements**

## 15.2 Syntax

**Table 143 — Syntax of mae_AudioSceneInfo()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_AudioSceneInfo()<br>{ | | |
| **mae_isMainStream;** | **1** | **bslbf** |
| if (mae_isMainStream) { | | |
| **mae_audioSceneInfoIDPresent**; | **1** | **bslbf** |
| if (mae_audioSceneInfoIDPresent) { | | |
| **mae_audioSceneInfoID**; | **8** | **uimsbf** |
| } | | |
| **mae_numGroups;** | **7** | **uimsbf** |
| mae_GroupDefinition( mae_numGroups ); | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| **mae_numSwitchGroups;** | **5** | **uimsbf** |
| mae_SwitchGroupDefinition( mae_numSwitchGroups ); | | |
| **mae_numGroupPresets;** | **5** | **uimsbf** |
| mae_GroupPresetDefinition( mae_numGroupPresets ); | | |
| mae_Data(); | | |
| mae_metaDataElementIDoffset = 0; | | |
| } | | |
| else { | | |
| **mae_bsMetaDataElementIDoffset;** | **7** | **uimsbf** |
| mae_metaDataElementIDoffset = mae_bsMetaDataElementIDoffset + 1; | | |
| } | | |
| } | | |

**Table 144 — Syntax of mae_Data()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_Data() | | |
| { | | |
| **mae_numDataSets;** | **4** | **uimsbf** |
| for (dscr = 0; dscr < mae_numDataSets; dscr ++) { | | |
| **mae_dataType**[dscr]**;** | **4** | **uimsbf** |
| **mae_dataLength**[dscr]**;** | **16** | **uimsbf** |
| switch (mae_dataType[dscr] ) { | | |
| | | |
| case ID_MAE_GROUP_DESCRIPTION: | | |
| mae_Description( ID_MAE_GROUP_DESCRIPTION ); | | |
| break; | | |
| case ID_ MAE_SWITCHGROUP_DESCRIPTION: | | |
| mae_Description( ID_MAE_SWITCHGROUP_DESCRIPTION ); | | |
| break; | | |
| case ID_ MAE_GROUPPRESET_DESCRIPTION: | | |
| mae_Description( ID_MAE_GROUPPRESET_DESCRIPTION ); | | |
| break; | | |
| case ID_ MAE_GROUP_CONTENT: | | |
| mae_ContentData(); | | |
| break; | | |
| case ID_MAE_GROUP_COMPOSITE: | | |
| mae_CompositePair() | | |
| break; | | |
| case ID_MAE_SCREEN_SIZE: | | |
| mae_ProductionScreenSizeData(); | | |
| break; | | |
| default: | | |
| while (mae_dataLength[dscr] ) { | | |
| **tmp;** | **8** | **uimsbf** |
| } | | |
| break; | | |
| } | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 145 — Syntax of mae_GroupDefinition()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_GroupDefinition( numGroups ) | | |
| { | | |
| for ( grp = 0; grp < numGroups; grp++ ) { | | |
| **mae_groupID**[grp]; | **7** | **uimsbf** |
| **mae_allowOnOff**[grp]; | **1** | **bslbf** |
| **mae_defaultOnOff**[grp]; | **1** | **bslbf** |
| | | |
| **mae_allowPositionInteractivity**[grp]; | **1** | **bslbf** |
| if (mae_allowPositionInteractivity[grp] ) { | | |
| **mae_interactivityMinAzOffset**[grp]; | **7** | **uimsbf** |
| **mae_interactivityMaxAzOffset**[grp]; | **7** | **uimsbf** |
| **mae_interactivityMinElOffset**[grp]; | **5** | **uimsbf** |
| **mae_interactivityMaxElOffset**[grp]; | **5** | **uimsbf** |
| **mae_interactivityMinDistFactor**[grp]; | **4** | **uimsbf** |
| **mae_interactivityMaxDistFactor**[grp]; | **4** | **uimsbf** |
| } | | |
| **mae_allowGainInteractivity**[grp]; | **1** | **bslbf** |
| if (mae_allowGainInteractivity[grp] ) { | | |
| **mae_interactivityMinGain**[grp]; | **6** | **uimsbf** |
| **mae_interactivityMaxGain**[grp]; | **5** | **uimsbf** |
| } | | |
| | | |
| **mae_groupPriority**[grp]; | **3** | **uimsbf** |
| **mae_closestSpeakerPlayout**[grp]; | **1** | **bslbf** |
| | | |
| **mae_bsGroupNumMembers**[grp]; | **7** | **uimsbf** |
| **mae_hasConjunctMembers[grp]**; | **1** | **bslbf** |
| | | |
| if ( mae_hasConjunctMembers[grp] ) { | | |
| **mae_startID**[grp]; | **7** | **uimsbf** |
| } | | |
| else { | | |
| for ( obj = 0; obj < mae_bsGroupNumMembers + 1; obj++ ) { | | |
| **mae_metaDataElementID**[grp][obj]; | **7** | **uimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**Table 146 — Syntax of mae_SwitchGroupDefinition()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_SwitchGroupDefinition( numSwitchGroups ) | | |
| { | | |
| for ( swgrp = 0; swgrp < numSwitchGroups; swgrp ++ ) { | | |
| **mae_switchGroupID**[swgrp]; | **5** | **uimsbf** |
| **mae_switchGroupAllowOnOff**[swgrp]; | **1** | **bslbf** |
| if (mae_switchGroupAllowOnOff[swgrp]) { | | |
| **mae_switchGroupDefaultOnOff**[swgrp]; | **1** | **bslbf** |
| } | | |
| **mae_bsSwitchGroupNumMembers**[swgrp]; | **5** | **uimsbf** |
| for ( grp = 0; grp < mae_bsSwitchGroupNumMembers + 1; grp++ ) { | | |
| **mae_switchGroupMemberID**[swgrp][grp]; | **7** | **uimsbf** |

© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| }<br>**mae_switchGroupDefaultGroupID**[swgrp];<br>}<br>} | **7** | **uimsbf** |

**Table 147 — Syntax of mae_Description()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_Description( type )<br>{<br>**mae_bsNumDescriptionBlocks;**<br>for ( n = 0; n < mae_bsNumDescriptionBlocks + 1; n++ ) {<br>if ( type == ID_MAE_GROUP_DESCRIPTION ) {<br>**mae_descriptionGroupID**[n];<br>}<br>else if ( type == ID_MAE_SWITCHGROUP_DESCRIPTION ) {<br>**mae_descriptionSwitchGroupID**[n];<br>}<br>else if ( type == ID_MAE_GROUPPRESET_DESCRIPTION ) {<br>**mae_descriptionGroupPresetID**[n];<br>}<br>**mae_bsNumDescLanguages[n];**<br>for (i=0; i<mae_bsNumDescLanguages[n] + 1; i++) {<br>**mae_bsDescriptionLanguage**[n][i];<br>**mae_bsDescriptionDataLength**[n][i];<br>for ( c = 0; c < mae_bsDescriptionDataLength[n][i] + 1; c++ ) {<br>**mae_descriptionData**[n][i][c];<br>}<br>}<br>}<br>} | **7**<br>**7**<br>**5**<br>**5**<br>**4**<br>**24**<br>**8**<br>**8** | **uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**uimsbf** |

**Table 148 — Syntax of mae_ContentData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_ContentData()<br>{<br>**mae_bsNumContentDataBlocks;**<br>for ( n = 0; n < mae_bsNumContentDataBlocks + 1; n++ ) {<br>**mae_ContentDataGroupID**[n];<br>**mae_contentKind**[n];<br>**mae_hasContentLanguage**[n];<br>if (mae_hasContentLanguage[n] ) {<br>**mae_contentLanguage**[n];<br>}<br>}<br>} | **7**<br>**7**<br>**4**<br>**1**<br>**24** | **uimsbf**<br>**uimsbf**<br>**uimsbf**<br>**bslbf**<br>**uimsbf** |

**Table 149 — Syntax of mae_CompositePair()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_CompositePair()<br>{<br>**mae_bsNumCompositePairs;** | **7** | **uimsbf** |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| for (sa=0; sa<mae_bsNumCompositePairs + 1; sa++) { | | |
| **mae_CompositeElementID** [sa][0]; | **7** | **uimsbf** |
| **mae_CompositeElementID** [sa][1]; | **7** | **uimsbf** |
| } | | |
| } | | |

**Table 150 — Syntax of mae_GroupPresetDefinition()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_GroupPresetDefinition(numGroupPresets) | | |
| { | | |
| for ( gp = 0; gp < numGroupPresets; gp++ ) { | | |
| **mae_groupPresetID**[gp]; | **5** | **uimsbf** |
| **mae_groupPresetKind**[gp]; | **5** | **uimsbf** |
| **mae_bsGroupPresetNumConditions**[gp]; | **4** | **uimsbf** |
| for ( cnd = 0; cnd < mae_bsGroupPresetNumConditions + 1; cnd++ ) { | | |
| **mae_groupPresetGroupID**[gp][cnd]; | **7** | **uimsbf** |
| **mae_groupPresetConditionOnOff**[gp][cnd]; | **1** | **bslbf** |
| if (mae_groupPresetConditionOnOff[gp][cnd]) { | | |
| **mae_groupPresetDisableGainInteractivity**[gp][cnd]; | **1** | **bslbf** |
| **mae_groupPresetGainFlag**[gp][cnd]; | **1** | **bslbf** |
| if ( mae_groupPresetGainFlag[gp][cnd] ) { | | |
| **mae_groupPresetGain**[gp][cnd]; | **8** | **uimsbf** |
| } | | |
| **mae_groupPresetDisablePositionInteractivity**[gp][cnd]; | **1** | **bslbf** |
| **mae_groupPresetPositionFlag**[gp][cnd]; | **1** | **bslbf** |
| if ( mae_groupPresetPositionFlag [gp][cnd] ) { | | |
| **mae_groupPresetAzOffset**[gp][cnd]; | **8** | **uimsbf** |
| **mae_groupPresetElOffset**[gp][cnd]; | **6** | **uimsbf** |
| **mae_groupPresetDistFactor**[gp][cnd]; | **4** | **uimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| } | | |

**Table 151 — Syntax of mae_ProductionScreenSizeData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mae_ProductionScreenSizeData() | | |
| { | | |
| **hasNonStandardScreenSize;** | **1** | **bslbf** |
| If (hasNonStandardScreenSize) { | | |
| **bsScreenSizeAz;** | **9** | **uimsbf** |
| **bsScreenSizeTopEl;** | **9** | **uimsbf** |
| **bsScreenSizeBottomEl;** | **9** | **uimsbf** |
| } | | |
| } | | |

## 15.3 Semantics

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| **mae_isMainStream** | This flag signals if the actual MPEG-H stream is the main stream. |
| **mae_audioSceneInfoIDPresent** | This flag defines if an ID for the current audio scene is present in the bitstream. |
| **mae_audioSceneInfoID** | This field defines the ID for the current audio scene. This field can take values between 0 and 255. A value of 0 means that the ID shall not be evaluated. |
| **mae_numGroups** | This field signals the number of groups in the overall audio scene (complete number of groups in the main stream plus all possible additional streams). This field shall take values between 1 and 127, a minimum number of one group and a maximum number of 127 groups are assumed. |
| **mae_numSwitchGroups** | This field signals the number of switch groups in the overall scene (complete number of switch groups in the main stream plus all possible additional streams). This field can take values between 0 and 31, resulting in a maximum number of 31 switch groups. |
| **mae_numGroupPresets** | Number of defined group presets. This field can take values between 0 and 31, resulting in a maximum number of 31 group presets. A group preset is a subset of all groups of an overall scene where the on/off status of each of these groups is bound to a condition. With group presets it is possible to define a controlled behaviour in dependence of the on/off status of some of the groups in an overall audio scene. A group preset is valid if all its associated conditions are true (logical AND of the conditions yields 1), i.e. if the on/off status of all associated groups is conform to the defined conditions.<br><br>An example is a group preset with the associated groups $mae_groupID_1$ and $mae_groupID_3$. This group preset gets valid if the group with $mae_groupID_1$ is switched on and the group with $mae_groupID_3$ is switched off: validity(group preset) = ((status($mae_groupID_1$)==on) && !(status($mae_groupID_3$)==on))<br>The validity status of a group preset can be connected to a special treatment, e.g. it can be connected to a defined DRC sequence which is only applied if the group preset is valid. |
| **mae_bsMetaDataElementIDoffset** | This field defines the offset for the first metadata element of the current MPEG-H data stream. It is zero if the stream is the main stream. |
| **mae_numDataSets** | This field defines the number of data sets that are following in the bitstream. |
| **mae_dataType** | For each data element this field defines the type of description that follows in the bitstream. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 152 — Value of mae_dataType**

| mae_dataType | value | meaning |
|---|---|---|
| ID_MAE_GROUP_DESCRIPTION | 0 | Group description follows in the bitstream |
| ID_MAE_SWITCHGROUP_DESCRIPTION | 1 | Switch group description follows in the bitstream |
| ID_ MAE_GROUP_CONTENT | 2 | Group content information follows in the bitstream |
| ID_MAE_GROUP_COMPOSITE | 3 | Composite pair information follows in the bitstream |
| ID_MAE_SCREEN_SIZE | 4 | Information about the local screen size follows in the bitstream |
| ID_MAE_GROUPPRESET_DESCRIPTION | 5 | Group preset description follows in the bitstream |
| reserved | 6 - 15 | n/a |

**mae_dataLength** This field defines the length in bytes of the data element that follows in the bit stream.

**mae_groupID** This field uniquely defines an ID for a group of metadata elements. This field can take values between 0 and 127.

**mae_allowOnOff** This flag defines if the audience is allowed to switch a metadata element group on and off (enable/disable playback).

**mae_defaultOnOff** This field defines the default status of a metadata element group, i.e. if this group is switched on (=1) or off (=0) by default.

**mae_allowPositionInteractivtiy** This flag defines if the audience is allowed to change the position of the elements of a metadata element group.

**mae_interactivityMinAzOffset** This field defines the minimum azimuth offset for changing the position of the members of a metadata element group, e.g. changing the original azimuth by a minimum offset of -30°. This field can take values between MinAzOffset = -180° and MinAzOffset = 0°:

MinAzOffset = –1.5 · mae_interactivityMinAzOffset

**mae_interactivityMaxAzOffset** This field defines the maximum azimuth offset for changing the position of the members of a metadata element group, e.g. changing the original azimuth by a maximum offset of +30°. This field can take values between MaxElOffset = 0° and MaxElOffset =180°:

MaxAzOffset = +1.5 · mae_interactivityMaxAzOffset

**mae_interactivityMinElOffset** This field defines the minimum elevation offset for changing the position of the members of a metadata element group, e.g. changing the original elevation by a minimum offset of -30°. This field can take values between MinElOffset = -90° and MinElOffset = 0°:

MinElOffset = –3 · mae_interactivityMinElOffset

**mae_interactivityMaxElOffset** This field defines the maximum elevation offset for changing the position of the members of a metadata element group, e.g. changing the original elevation by a maximum offset of +30°. This field can take values between MaxElOffset = 0° and MaxElOffset = 90°:

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

MaxElOffset = +3 · mae_interactivityMaxElOffset

**mae_interactivityMinDistFactor**

This field defines the minimum distance change factor for interactively changing the position of the members of a metadata element group. The field describes a multiplication factor by which the original distance is changed, e.g. multiplied by a minimum factor of 0.5. This field can take values between 0 and 15 resulting in MinDistFactor between 0.00025 and 8:

$$MinDistFactor = 2^{(mae_interactivityMinDistFactor - 12)}$$

**mae_interactivitiyMaxDistFactor**

This field defines the maximum distance change factor for interactively changing the position of the members of a metadata element group. The field describes a multiplication factor by which the original distance is changed, e.g. multiplied by a maximum factor of 4. This field can take values between 0 and 15 resulting in MaxDistFactor between 0.00025 and 8:

$$MaxDistFactor = 2^{(mae_interactivityMaxDistFactor - 12)}$$

**mae_allowGainInteractivity**

This flag defines if the audience is allowed to change the gain of a metadata element group.

**mae_interactivityMinGain**

This field defines a minimum gain of the members of a metadata element group. The field can take values between
MinGain = –63 dB
and
MinGain = 0 dB
in 1 dB steps, with

MinGain in dB = **mae_interactivityMinGain** – 63

If **mae_interactivityMinGain** is set to 0, MinGain shall be set to minus infinity dB.

**mae_interactivityMaxGain**

This field defines a maximum gain of the members of a metadata element group. The field can take values between
MaxGain = 0dB
and
MaxGain =31dB
in 1dB steps, with

MaxGain in dB = **mae_interactivityMaxGain**

The value of MinGain and MaxGain define the interval in dB of allowed interactivity gain changes relative to the OAM gain if OAM data is present for the current group. For example, with a MinGain of -2dB and a MaxGain of 6dB, the gain of the elements in the group shall be interactively altered only in the interval between -2dB and +6dB relative to their current OAM gain in dB. For groups without OAM data, the MinGain and MaxGain values define the interval in dB of allowed interactivity gain changes relative to the current audio gain (amplitude of the audio samples).

**mae_groupPriority**

This field defines the priority of the group. It can take integer values between 0 and 7. The group may be discarded from rendering and decoding if the prioity is lower than 7. If groups are discarded, the groups with lowest priority should be discarded first.

**mae_closestSpeakerPlayout**

This flag defines that the members of the metadata element group should not be rendered but directly be played back by the loudspeakers

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | |
|---|---|
| | which are nearest to the geometric position of the members as defined in 18.5. |
| **mae_bsGroupNumMembers** | This field signals the number of members of a group of metadata elements. The field can take values between 0 and 127 resulting in a maximum number of 128 members. |
| **mae_hasConjunctMembers** | This flag defines if all members of the metadata element group are coded consecutively in the bitstream. |
| **mae_startID** | If the members of the element group are coded consecutively this field defines the offset for the first metadata element of this group. |
| **mae_metaDataElementID** | This field uniquely defines an ID for a metadata element (i.e. channels, dynamic objects, SAOC channels, SAOC objects, HOA). This field can take values between 0 and 127, resulting in a maximum of 128 metadata elements. One group shall only contain elements of the same signal type (static objects (channel-based recordings without accompanying OAM data), objects (audio tracks with accompanying OAM data), SAOC channels, SAOC object, HOA). |
| **mae_switchGroupID** | This field uniquely defines an ID for a switch group of metadata elements groups. |
| **mae_switchGroupAllowOnOff** | This flag defines if the audience is allowed to completely disable the playback of the switch group. If the flag is set to zero, then one member of the group is always played back, if the flag is set to one, then either none or one member of the group is played back. |
| **mae_switchGroupDefaultOnOff** | This flag defines if the switch group is enabled or disabled for playback by default. If the flag is enabled by default, then the default member of the group is played back in the initial setting. |
| **mae_bsSwitchGroupNumMembers** | This field signals the number of members of a switch group. It can take values between 0 and 31, resulting in a maximum number of 32 members. If a group is a member of a switch group, the mae_allowOnOff field should be ignored during processing and playback. |
| **mae_switchGroupMemberID** | This field specifes the groupIDs of the members of the switch group. |
| **mae_switchGroupDefaultGroupID** | This field signals default member of the switch group (i.e. if a switch group is selected for playback by the audience, the default member is played back until the audience switches to another member). |
| **mae_bsNumDescriptionBlocks** | This field specifies the number of description blocks. |
| **mae_descriptionGroupID** | This field specifies the mae_groupID of the group to which the description block applies. |
| **mae_descriptionSwitchGroupID** | This field specifies the mae_switchGroupID of the switch group to which the description block applies. |
| **mae_descriptionGroupPresetID** | This field specifies the mae_groupPresetID of the group preset to which the description block applies. |
| **mae_bsNumDescLanguages** | This field specifies the number of available languages for the description text. |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**mae_bsDescriptionLanguage** This 24-bit field identifies the language of the description text of a metadata element group. It contains a 3-character code as specified by ISO 639-2. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO/IEC 8859-1 and inserted in order into the 24-bit field. EXAMPLE: French has 3-character code “fre”, which is coded as: “0110 0110 0111 0010 0110 0101”.

**mae_bsDescriptionDataLength** This field defines the length of the following group description in the bit stream.

**mae_descriptionData** This field contains a description of a metadata element group or a switch group, i.e. a string describing the content by a high-level description. The format shall follow UTF-8 according to ISO/IEC 10646.

**mae_bsNumContentDataBlocks** This field specifies the number of ContentData blocks.

**mae_contentDataGroupID** This field specifies the mae_groupID of the group to which the ContentData block applies.

**mae_contentKind** This field defines the kind of content of a metadata element group,

**Table 153 — Value of mae_contentKind**

| mae_contentKind | description |
|---|---|
| 0 | undefined |
| 1 | complete main |
| 2 | dialogue |
| 3 | music |
| 4 | effect |
| 5 | mixed |
| 6 | LFE |
| 7 | voiceover |
| 8 | spokensubtitle |
| 9 | audiodescription/visually impaired |
| 10 | commentary |
| 11 | hearing impaired |
| 12 | emergency |
| 13-15 | reserved |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**mae_hasContentLanguage** This field defines if the actual metadata element group has a language assigned to its content.

**mae_contentLanguage** This 24-bit field identifies the language of a metadata element group. It contains a 3-character code as specified by ISO 639-2. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO/IEC 8859-1 and inserted in order into the 24-bit field. EXAMPLE: French has 3-character code “fre”, which is coded as: “0110 0110 0111 0010 0110 0101”.

**mae_bsNumCompositePairs** This is the number of CompositePairs

**mae_CompositeElementID** This field uniquely defines an ID for a metadata element. This field can take values between 0 and 127, resulting in a maximum of 128 metadata


© ISO/IEC 2015 – All rights reserved

elements. The metadata element in position 0, is an *independent* object and the metadata element in position 1 is a *dependent* object. These two objects form together the CompositePair. This dependent object will be given the metadata of the independent object. The dependent object and the independent object represent a composite object pair combining the dependent object and the independent object in time or frequency domain. Each element may be either a discrete object or an SAOC object.

**mae_groupPresetID** This field uniquely defines an ID for a group preset. This field can take values between 0 and 31. The group preset with mae_groupPresetID == 0 shall be selected as default.

**mae_groupPresetKind** This field defines the kind of content of a group preset.

**Table 154 — Value of mae_groupPresetKind**

| mae_groupPresetKind | description |
|---|---|
| 0 | undefined |
| 1 | integrated TV loudspeaker |
| 2 | high quality loudspeaker |
| 3 | mobile speakers |
| 4 | mobile headphones |
| 5 | hearing impaired (light) |
| 6 | hearing impaired (heavy) |
| 7 | visually impaired / audio description |
| 8 | spoken subtitles |
| 9 | loudness/DRC |
| 10-25 | /* reserved for ISO use */ |
| 26-30 | /* reserved for use outside of ISO scope */ |
| 31 | other |

**mae_bsGroupPresetNumConditions**
This field defines the number of group conditions that are associated with a group preset. The field takes values between 0 and 15; a minimum of 1 condition and a maximum of 16 conditions are assumed. A condition is a combination of a mae_groupID and an on/off status.

**mae_groupPresetGroupID** This field specifies the groups associated with a group preset.

**mae_groupPresetConditionOnOff**
This flag describes the required on/off status of a group associated with a group preset. If the flag is 1, the associated group has to be switched on to validate the group preset. If the flag is 0, the associated group has to be switched off.

**mae_groupPresetDisableGainInteractivity**
This field defines if the gain interactivity of the currently referenced group shall be disabled (flag is equal to 1) or shall stay enabled (flag is equal to 0) if the preset is chosen/valid.

**mae_groupPresetGainFlag** This field defines whether the corresponding preset specifies an initial gain of the members of a metadata element group. It shall only be 1 if the flag mae_allowPositionInteractivtiy of the corresponding group is set to 1.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**mae_groupPresetGain** The field defines the initial gain of the members of a metadata element group when the corresponding preset is selected.

groupPresetGain in dB = 0.5 · (**mae_groupPresetGain** – 255) + 32

**mae_groupPresetDisablePositionInteractivity**
This field defines if the position interactivity of the currently referenced group shall be disabled (flag is equal to 1) or shall stay enabled (flag is equal to 0) if the present is chosen/valid.

**mae_groupPresetPositionFlag**
This field defines whether initial position interactivity data (azimuth offset, elevation offset and distance factor) is present that shall be applied to the members of a metadata element group. It shall only be 1 if the flag mae_allowPositionInteractivity of the corresponding group is set to 1.

**mae_groupPresetAzOffset** This field defines the additional azimuth offset that shall be applied to the currently referenced group if the preset is chosen/valid. This field can take values between PresetAdditionalAzOffset = -180° and AzOffset = +180°:

PresetAdditionalAzOffset =1.5 · **mae_groupPresetAzOffset**

**mae_groupPresetElOffset** This field defines the additional elevation offset that shall be applied to the currently referenced group if the preset is chosen/valid. This field can take values between PresetAdditionalElOffset = -90° and ElOffset = +90°:

PresetAdditionalElOffset = 3 · **mae_groupPresetElOffset**

**mae_groupPresetDistFactor** This field defines the additional distance change factor that shall be applied to the currently referenced group if the preset is chosen/valid. This field can take values between 0 and 15 resulting in PresetAdditionalDistFactor between 0.00025 and 8:

$$\text{PresetAdditionalDistFactor} = 2^{(\text{mae_groupPresetDistFactor}-12)}$$

**hasNonStandardScreenSize** This flag specifies whether a nominal screen size is defined that is different than the standard screen size. The definition is done via viewing angles corresponding to the screen edges. In case hasNonStandardScreenSize is zero, the following values are used as default (assuming a 4k display and an optimal viewing distance):

$\varphi_{\text{left}}^{\text{nominal}}$ = 29.0°
$\varphi_{\text{right}}^{\text{nominal}}$ = -29.0°
$\theta_{\text{top}}^{\text{nominal}}$ = 17.5°
$\theta_{\text{bottom}}^{\text{nominal}}$ = -17.5°

**bsScreenSizeAz** This field defines the azimuth corresponding to the left and right screen edge:

$\varphi_{\text{left}}^{\text{nominal}}$ = 0.5 · bsScreenSizeAz
$\varphi_{\text{left}}^{\text{nominal}}$ = min (max ( $\varphi_{\text{left}}^{\text{nominal}}$ , 0), 180);
$\varphi_{\text{right}}^{\text{nominal}}$ = -0.5 · bsScreenSizeAz
$\varphi_{\text{right}}^{\text{nominal}}$ = min (max ( $\varphi_{\text{right}}^{\text{nominal}}$ , -180), 0);

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**bsScreenSizeTopEl** This field defines the elevation corresponding to the top screen edge:

$\theta_{top}^{nominal}$ = 0.5 · (bsScreenSizeTopEl – 255)

$\theta_{top}^{nominal}$ = min (max ( $\theta_{top}^{nominal}$ , -90), 90);

**bsScreenSizeBottomEl** This field defines the elevation corresponding to the bottom screen edge:

$\theta_{bottom}^{nominal}$ = 0.5 · (bsScreenSizeBottomEl – 255)

$\theta_{bottom}^{nominal}$ = min (max ( $\theta_{bottom}^{nominal}$ , -90), 90);

## 15.4 Definition of mae_metaDataElementIDs

The definition of metadata groups (mae_GroupDefinition) references signals via the identifiers mae_metaDataElementID. These identifiers are calculated using the following pseudo code:

```
mae_metaDataElementID = mae_metaDataElementIDoffset;
for ( grp = 0; grp < bsNumSignalGroups + 1; grp++)  {
    if ( SignalGroupType[grp] == SignalGroupTypeChannels ) {
        for ( id = 0; id < bsNumberOfSignals[grp] + 1; id++ ) {
            mae_metaDataElementID++;
        }
   }
   else if ( SignalGroupType[grp] == SignalGroupTypeObject ) {
        for ( id = 0; id < bsNumberOfSignals[grp] + 1; id++ )  {
            mae_metaDataElementID++;
        }
   }
   else if ( SignalGroupType[grp] == SignalGroupTypeSAOC ) {
        for ( id = 0; id < numSpeakers + bsNumSaocObjects; id++ )  {
            mae_metaDataElementID++;
        }
    }
    else if ( SignalGroupType[grp] == SignalGroupTypeHOA ) {
        mae_metaDataElementID++;
    }
}
```

with `numSpeakers` as used when evaluating SAOC3DgetNumChannels(saocChannelLayout).

# 16 Loudspeaker Distance Compensation

At the end of the loudspeaker feed processing chain a loudspeaker compensation shall be applied as follows if no external compensation is conducted. If the distances of the local loudspeakers to the center of the listening area (‘sweet spot’) differ with respect to each other, distance compensation shall be applied. It applies trim parameters (compensation gains as well as compensation delays) to counter the effects of non-uniform loudspeaker distances.

The trim parameters

$T_{d,A}$: trim delay in samples for each output channel *A*,

$T_{g,A}$: trim gain (linear gain value) for each output channel *A*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

shall be computed as follows as a function of the loudspeaker distances in $trim_A$, the distances given in meters:

$$T_{d,A} = \text{round}\left( \frac{\max_n trim_n - trim_A}{340 / f_s} \right) \text{ with output sampling rate } f_s \text{ in Hz,}$$

$$T_{g,A} = \frac{\sqrt{trim_A}}{\sqrt{\max_n trim_n}}.$$

Note that $\max_n trim_n$ denotes the distance of the loudspeaker located farest away from the sweet spot. Thus, all trim delays are non-negative and no trim gain exceeds the neutral gain value 1.0.

If the distances of the loudspeakers of the reproduction setup are non-uniform, the derived trim parameters shall be applied accordingly: The loudspeaker signal of output channel $A$ shall be delayed by $T_{d,A}$ time domain samples and the signal shall also be multiplied by the linear gain $T_{g,A}$.

Besides the distance compensation, the loudspeaker calibration gains shall be applied here.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

# 17 Interfaces to the MPEG-H 3D audio decoder

## 17.1 General

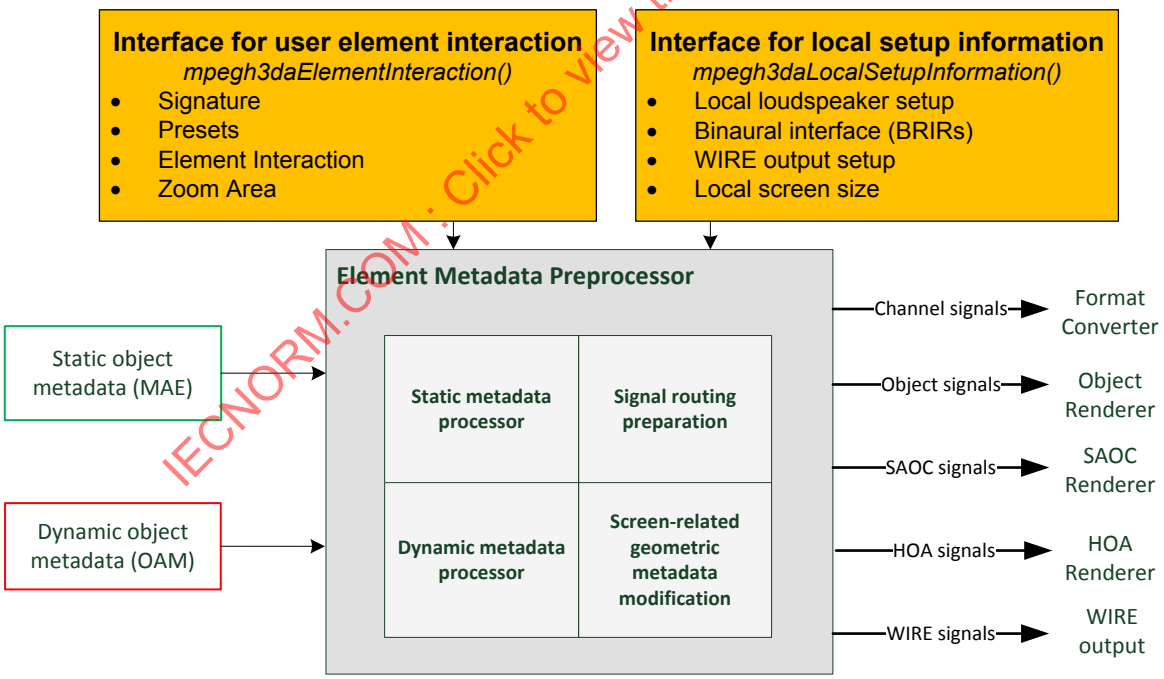



**Figure 69 — Interfaces to MPEG-H 3D audio decoder**


© ISO/IEC 2015 – All rights reserved

## 17.2 Interface for local setup information

### 17.2.1 General

The mpegh3daLocalSetupInformation() syntax element defines the formatting of local setup information for the rendering. It gives the rendering type and the screen information if available. Two different rendering types are defined: Loudspeaker Rendering and Binaural Rendering.

### 17.2.2 WIRE output

In addition to the signaling of local loudspeakers or the binaural configuration, it is also possible to signal WIRE outputs. WIRE outputs are arbitrary output devices (e.g. headphone outputs or auxiliary speakers) that exist in addition to the regular playback device. The output after the decoder then shall first contain the regular speaker signals and the WIRE outputs. A WIRE output can be single channel or multichannel.

The element groups in a WIRE output should be processed with the same Loudness/DRC processing as if they would be part of the regular speaker signals. The application of a peak limiter at the end of the processing chain is highly recommended.

Only channels or objects can be routed to WIRE outputs.

### 17.2.3 Syntax for local setup information

**Table 155 — Syntax for mpegh3daLocalSetupInformation()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daLocalSetupInformation() | | |
| { | | |
| **bsRenderingType**; | **1** | **uimsbf** |
| switch (bsRenderingType) { | | |
| case 0: | | |
| LoudspeakerRendering(); | | |
| case 1: | | |
| BinauralRendering(); | | |
| } | | |
| **bsNumWIREoutputs**; | **16** | **uimsbf** |
| if (bsNumWIREoutputs > 0) { | | |
| for (n = 0; n < bsNumWIREoutputs; n++) { | | |
| **WireID**[n]; | **16** | **uimsbf** |
| } | | |
| } | | |
| **hasLocalScreenSizeInformation;** | **1** | **bslbf** |
| if (hasLocalScreenSizeInformation) { | | |
| LocalScreenSizeInformation(); | | |
| } | | |
| } | | |

### 17.2.4 Semantics for local setup information

**bsRenderingType** Defines the type of rendering (loudspeaker rendering or binaural rendering).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**bsNumWIREoutputs** This field defines the number of WIRE outputs. A maximum of 65535 WIRE outputs is assumed; therefore the field can take values between 0 and 65535.

**WireID** This field contains an identifier for a WIRE output.

**hasLocalScreenSizeInformation** Flag that defines if information about the local screen size is available. If yes, the local screen size information follows afterwards.

## 17.3 Interface for local loudspeaker setup and rendering

### 17.3.1 General

The LoudspeakerRendering() syntax element defines the signaling of the local loudspeaker setup.

The local loudspeaker setup for loudspeaker reproduction is signaled using the syntax element SpeakerConfig3D() (see 5.2.2.2). The speaker setup can be given by either:

— a CICPspeakerLayoutIdx (speakerLayoutType = 0), or

— a list of CICPspeakerIdx (speakerLayoutType = 1), or

— a mixed list of speaker positions in terms of azimuth and elevation and CICPspeakerIdx (speakerLayoutType = 2).

In case speakerLayoutType=0 or speakerLayoutType=1 is used, the exact position may be signaled in addition for each speaker. In that way it is possible to signal standard speaker layouts with speaker misplacement. In this case, the decoder is able to utilize the implicit semantic information about the setup.

The Format Converter provides optimized behavior if a standard setup plus offset values is provided as setup information. It therefore takes offset values from the nominal speaker positions as an input parameter (see 10.3.4). In case that speakerLayoutType<=1 is used and exact position information are transmitted, the exact position information shall be used to calculate the offsets from the nominal speaker positions. These offsets for azimuth and elevation angles are defined as the difference between signaled position and nominal position. The nominal positions are defined in ISO/MPEG 23001-8.

If speakerLayoutType=0 (CICPspeakerLayoutIdx) is used, the order of channels is implicitly given and the order of output channels then follows the order of channels in Table 95. If the channel order of a standard setup is permuted (with respect to the order given in Table 95), then speakerLayoutType=1 shall be used for signaling, with a different order of CICPspeakerIdx than in Table 95. Signaling of speakerLayoutType=1 shall also be used for setups that are not standard layouts but do consist of standard loudspeakers. Completely arbitrary setups shall be signaled by speakerLayoutType=2.

If speakerLayoutType=1 or speakerLayoutType=2 is used, the output of the decoder then shall follow the order given in the LoudspeakerRendering() syntax element.

In addition to the layout information, a distance to the reference point (‘sweet spot’) is defined for each speaker which shall be directly used by the renderer in the loudspeaker compensation processing (see clause 16).

Further, a loudspeaker calibration gain value is defined for each speaker which shall be directly used by the renderer in the loudspeaker compensation processing (see clause 16).

### 17.3.2 Syntax for local loudspeaker signaling

**Table 156 — Syntax for LoudspeakerRendering()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

```
LoudspeakerRendering()                                          
{                                                               
    bsNumLoudspeakers;                                  16      uimsbf
    localLoudspeakerSetup = SpeakerConfig3d();                  
    hasLoudspeakerDistance;                             1       bslbf
    hasLoudspeakerCalibrationGain;                      1       bslbf
    for (n = 0; n < bsNumLoudspeakers; n++)  {                  
        if (speakerLayoutType <= 1) {                           
            hasKnownPosition[n];                        1       bslbf
            if (hasKnownPosition[n]) {                          
                loudspeakerAzimuth[n];                  9       uimsbf
                loudspeakerElevation[n];                8       uimsbf
            }                                                   
        }                                                       
        if (hasLoudspeakerDistance) {                           
            loudspeakerDistance[n];                     10      uimsbf
        }                                                       
        if (hasLoudspeakerCalibrationGain) {                    
            loudspeakerCalibrationGain[n];              7       uimsbf
        }                                                       
    }                                                           
    externalDistanceCompensation;                       1       bslbf
}                                                               
```

### 17.3.3 Semantics for local loudspeaker signaling

**bsNumLoudspeakers** Number of loudspeakers in the local reproduction setup.

**hasKnownPosition** This flag defines if an explicit signaling of the known position of the loudspeaker is following in the bit stream.

**loudspeakerAzimuth** This field defines the azimuth angle of a speaker. It can take values between -180° and 180° in 1° steps.

Azimuth = (loudspeakerAzimuth – 256)

Azimuth = min (max (Azimuth, -180), 180)

**loudspeakerElevation** This field defines the elevation angle of a speaker. It can take values between -90° and 90° in 1° steps.

Elevation = (loudspeakerElevation – 128)

Elevation= min (max (Elevation, -90), 90)

**loudspeakerDistance** This field defines the distance of a loudspeaker to the reference point centered in the loudspeaker setup in cm. The field can take values between 0 and 1023, corresponding to 0 to 1023 cm in 1cm steps.

**loudspeakerCalibrationGain** This field defines a loudspeaker calibration gain in dB. The field can take values between 0 and 127, corresponding to dB values between Gain = -32 dB and Gain = 31.5 dB in 0.5 dB steps:

Gain [dB] = 0.5 × (loudspeakerGain – 64)

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**externalDistanceCompensation**

This flag defines whether loudspeaker compensation shall be a applied to the decoder output signals. If this flag is 1, the loudspeaker compensation shall be omitted and loudspeakerDistance and loudspeakerCalibrationGain shall not be applied in the decoder..

## 17.4 Interface for binaural room impulse responses (BRIRs)

### 17.4.1 Introduction

The BRIR/HRTF data for the binaural rendering can be fed to the decoder by using the syntax element BinauralRendering(). It supports the description of:

— raw FIR filters (for both virtual loudspeakers and HOA-to-Binaural approaches)

— low-level parameters associated to the Time-Domain Binaural Renderer (for both virtual loudspeakers and HOA-to-Binaural approaches) and the Frequency-Domain Binaural Renderer

In case the impulse responses are described as virtual loudspeaker filters, the measurement setup of the impulse responses is defined, i.e. the positions of speakers during the measurement procedure or sound sources in a room simulation.

The measurement setup of the selected impulse response set is signaled using the bitstream syntax element SpeakerConfig3d(). In case a CICPspeakerLayoutIdx is used for signaling the measurement setup, the order of the BRIR pairs has to follow the order of channels in Table 95; otherwise speakerLayoutType=1 or speakerLayoutType=2 shall be used to signal a different BRIR order.

The binaural data should contain at least one representation that is supported by the binaural renderer, otherwise no output is provided.

### 17.4.2 Syntax of Binaural Renderer Interface

**Table 157 – Syntax of BinauralRendering()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| BinauralRendering() | | |
| { | | |
| **bsFileSignature;** | **32** | **bslbf** |
| **bsFileVersion;** | **8** | **uimsbf** |
| **bsNumCharName**; | **8** | **uimsbf** |
| for ( i=0; i<bsNumCharName; i++ ) { | | |
| **bsName[i]**; | **8** | **bslbf** |
| } | | |
| **useHeadTrackingMode;** | **1** | **bslbf** |
| **bsNumBinauralDataRepresentation**; | **4** | **uimsbf** |
| for (r = 0; r < bsNumBinauralDataRepresentation; r++) { | | |
| **brirSamplingFrequencyIndex;** | **5** | **uimsbf** |
| if (brirSamplingFrequencyIndex == 0x1f ) { | | |
| **brirSamplingFrequency;** | **24** | **uimsbf** |
| } | | |
| **isHoaData;** | **1** | **bslbf** |
| if (isHoaData) { | | |
| **hoaOrderBinaural** = escapedValue(3,5,0); | **3,8** | **uimsbf** |
| nBrirPairs = (hoaOrderBinaural+1)^2; | | |
| } else { | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| MeasurementSetup = SpeakerConfig3d(); | | |
| if (speakerLayoutType == 0) { /* See SpeakerConfig3d() */ | | |
| **nBrirPairs** = escapedValue(5,8,0)+1; | **5,13** | **uimsbf** |
| } else { | | |
| nBrirPairs = numSpeakers; /* See SpeakerConfig3d() */ | | |
| } | | |
| } | | |
| **bsBinauralDataFormatID;** | **2** | **uimsbf** |
| ByteAlign(); | | |
| switch (bsBinauralDataFormatID) { | | |
| case 0: | | |
| BinauralFIRData(); | | |
| break; | | |
| case 1: | | |
| FdBinauralRendererParam(); | | |
| break; | | |
| case 2: | | |
| TdBinauralRendererParam(); | | |
| break; | | |
| } | | |
| } | | |
| } | | |
| NOTE: useHeadTrackingMode = 1 is reserved. | | |

**Table 158 — Syntax of BinauralFirData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| BinauralFirData() | | |
| { | | |
| **bsNumCoefs**; | **24** | **uimsbf** |
| for (pos = 0; pos < nBrirPairs; pos++) { | | |
| for (i = 0; i < bsNumCoefs; i++) { | | |
| **bsFirCoefLeft[pos][i];** | **32** | **bslbf** |
| **bsFirCoefRight[pos][i];** | **32** | **bslbf** |
| } | | |
| } | | |
| **bsAllCutFreq;** | **32** | **bslbf** |
| if (bsAllCutFreq == 0) { | | |
| for (pos = 0; pos < nBrirPairs; pos++) { | | |
| **bsCutFreqLeft[pos]**; | **32** | **bslbf** |
| **bsCutFreqRight[pos]**; | **32** | **bslbf** |
| } | | |
| } else { | | |
| for (pos = 0; pos < nBrirPairs; pos++) { | | |
| bsCutFreqLeft [pos] = bsAllCutFreq; | | |
| bsCutFreqRight [pos] = bsAllCutFreq; | | |
| } | | |
| } | | |
| } | | |

**Table 159 — Syntax of FdBinauralRendererParam()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| FdBinauralRendererParam() | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

| | | |
|---|---|---|
| { | | |
| **flagHrir;** | **1** | **bslbf** |
| **dInit;** | **10** | **uimsbf** |
| **kMax;** | **6** | **uimsbf** |
| **kConv;** | **6** | **uimsbf** |
| **kAna;** | **6** | **uimsbf** |
| VoffBrirParam(); | | |
| if (flagHrir==0) { | | |
| SfrBrirParam(); | | |
| } | | |
| QtdlBrirParam(); | | |
| } | | |

**Table 160 — Syntax of VoffBrirParam()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| VoffBrirParam() | | |
| { | | |
| **nBitNFilter;** | **4** | **uimsbf** |
| **nBitNFft;** | **3** | **uimsbf** |
| **nBitNBlk;** | **3** | **uimsbf** |
| for (k=0; k<kMax ; k++) { | | |
| **nFilter[k];** | **nBitNFilter** | **uimsbf** |
| **nFft[k];** | **nBitNFft** | **uimsbf** |
| fftLength = pow(2, nFft[k]); | | |
| **nBlk[k];** | **nBitNBlk** | **uimsbf** |
| for (b=0; b<nBlk[k]; b++) { | | |
| for (nr=0; nr< nBrirPairs; nr++) { | | |
| for (v=0; v<fftLength; v++) { | | |
| **VoffCoeffLeftReal[k][b][nr][v];** | **32** | **bslbf** |
| **VoffCoeffLeftImag[k][b][nr][v];** | **32** | **bslbf** |
| **VoffCoeffRightReal[k][b][nr][v];** | **32** | **bslbf** |
| **VoffCoeffRightImag[k][b][nr][v];** | **32** | **bslbf** |
| } | | |
| } | | |
| } | | |
| } | | |
| } | | |

**Table 161 — Syntax of SfrBrirParameter()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| SfrBrirParam() | | |
| { | | |
| for (k=0; k<kAna ; k++) { | | |
| **fcAna[k];** | **32** | **bslbf** |
| **rt60[k];** | **32** | **bslbf** |
| **nrgLr[k];** | **32** | **bslbf** |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 162 — Syntax of QtdlBrirParameter()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| QtdlBrirParam() | | |
| { | | |
| for (k=0; k<kMax-kConv ; k++) { | | |
| **nBitQtdlLag[k];** | **4** | **uimsbf** |
| for (nr=0; nr< nBrirPairs; nr++) { | | |
| **QtdlGainLeftReal[k][nr];** | **32** | **bslbf** |
| **QtdlGainLeftImag[k][nr];** | **32** | **bslbf** |
| **QtdlGainRightReal[k][nr];** | **32** | **bslbf** |
| **QtdlGainRightImag[k][nr];** | **32** | **bslbf** |
| **QtdlLagLeft[k][nr];** | **nBitQtdlLag[k]** | **uimsbf** |
| **QtdlLagRight[k][nr];** | **nBitQtdlLag[k]** | **uimsbf** |
| } | | |
| } | | |
| } | | |

**Table 163 — Syntax of TdBinauralRendererParam()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| TdBinauralRendererParam() | | |
| { | | |
| **bsDelay;** | **16** | **uimsbf** |
| **bsDirectLen**; | **16** | **uimsbf** |
| **bsNbDiffuseBlocks;** | **8** | **uimsbf** |
| for ( chan=0; chan< nBrirPairs; chan++ ) { | | |
| **bsFmaxDirectLeft[chan]**; | **32** | **bslbf** |
| **bsFmaxDirectRight[chan]**; | **32** | **bslbf** |
| } | | |
| for ( block=0; block< bsNbDiffuseBlocks; block++ ) { | | |
| **bsFmaxDiffuseLeft[block]**; | **32** | **bslbf** |
| **bsFmaxDiffuseRight[block]**; | **32** | **bslbf** |
| } | | |
| for ( chan=0; chan< nBrirPairs; chan++ ) { | | |
| **bsWeights[chan]**; | **32** | **bslbf** |
| } | | |
| for ( chan=0; chan< nBrirPairs; chan++ ) { | | |
| for ( tap=0; tap< bsDirectLen; tap++ ) { | | |
| **bsFIRDirectLeft[chan][tap]**; | **32** | **bslbf** |
| **bsFIRDirectRight[chan][tap]**; | **32** | **bslbf** |
| } | | |
| } | | |
| for ( bloc=0; bloc< bsNbDiffuseBlocks; bloc++ ) { | | |
| for ( tap=0; tap< bsDirectLen; tap++ ) { | | |
| **bsFIRDiffuseLeft[bloc][tap]**; | **32** | **bslbf** |
| **bsFIRDiffuseRight[bloc][tap]**; | **32** | **bslbf** |
| } | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

### 17.4.3 Semantics

**bsFileSignature** A string of 4 ASCII characters that reads “BRIR”.

© ISO/IEC 2015 – All rights reserved

**bsFileVersion** File version indication. This document describes version 1 of the file format.

**bsNumCharName** Number of ASCII characters in the BRIR name.

**bsName** BRIR name.

**useHeadTrackingMode** This flag defines if a headtracker is connected and the binaural rendering should be processed in a special headtracking mode.

**bsNumBinauralDataRepresentation**
Total number of representations of the filters.

**brirSamplingFrequencyIndex**
This index determines the sampling frequency of the BRIRs. The value of brirSamplingFrequencyIndex and their associated sampling frequencies are specified in Table 164.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**Table 164 — Value and meaning of brirSamplingFrequencyIndex**

| brirSamplingFrequencyIndex | sampling frequency |
|---|---|
| 0x00 | 96000 |
| 0x01 | 88200 |
| 0x02 | 64000 |
| 0x03 | 48000 |
| 0x04 | 44100 |
| 0x05 | 32000 |
| 0x06 | 24000 |
| 0x07 | 22050 |
| 0x08 | 16000 |
| 0x09 | 12000 |
| 0x0a | 11025 |
| 0x0b | 8000 |
| 0x0c | 7350 |
| 0x0d | reserved |
| 0x0e | reserved |
| 0x0f | 57600 |
| 0x10 | 51200 |
| 0x11 | 40000 |
| 0x12 | 38400 |
| 0x13 | 34150 |
| 0x14 | 28800 |
| 0x15 | 25600 |
| 0x16 | 20000 |
| 0x17 | 19200 |
| 0x18 | 17075 |
| 0x19 | 14400 |
| 0x1a | 12800 |
| 0x1b | 9600 |
| 0x1c | reserved |
| 0x1d | reserved |
| 0x1e | reserved |
| 0x1f | escape value |
| NOTE: The values of **brirSamplingFrequencyIndex** 0x00 up to 0x0e are identical to those of the **samplingFrequencyIndex** 0x0 up to 0xe contained in the AudioSpecificConfig() specified in ISO/IEC 14496-3:2009 | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

**brirSamplingFrequency** BRIR sampling frequency as unsigned integer value in case brirSamplingFrequencyIndex equals zero.

**isHoaData** Set to 1, this element indicates that corresponding BRIR coefficients are represented in the spherical harmonics (HOA) domain. Otherwise corresponding BRIR coefficients are represented in the virtual loudspeaker domain.

**hoaOrderBinaural** This element determines the HOA order of the filter dataset. The ordering of components is given by the “Single Index Designation” specified in ISO/IEC 14496-11, Table 6.

**nBrirPairs** The number of HRTF pairs that are transmitted, i.e. virtual speaker positions or number of channels or number of HOA components to be filtered by a pair of binaural filters.

© ISO/IEC 2015 – All rights reserved

**bsBinauralDataFormatID** Indicates the representation type of the BRIR set (FIR, FD parameterized or TD parameterized).

**bsNumCoefs** Indicates the number of coefficients of a FIR filter.

pos BRIR index ranged from 0 to nBrirPairs-1.

i Sample index ranged from 0 to bsNumCoefs-1.

**bsFirCoefLeft** FIR coefficient for left ear (interpreted as float in IEEE 754 format).

**bsFirCoefRight** FIR coefficient for right ear (interpreted as float in IEEE 754 format).

**bsAllCutFreq** if non-zero, the frequency above which left and right filters values can be neglected are the same for all filters and this value is bsAllCutFreq.

**bsCutFreqLeft** Frequency above which left filter value can be neglected.

**bsCutFreqRight** Frequency above which right filter value can be neglected.

nr BRIR order index ranged from 0 to nBrirPairs-1.

**flagHrir** Indicates whether inputted impulse response is HRIR or BRIR.

**Table 165 — Value of flagHrir**

| flagHrir | Meaning |
|---|---|
| 0 | BRIR, the sparse reverberator is switched on |
| 1 | HRIR, the sparse reverberator is switched off |

**dInit** Value of the propagation time.

**kMax** The maximum processing band.

**kConv** Number of bands used for convolution.

**kAna** Number of analysis bands used in analysis of late reverberation analysis.

**nBitNFilter** Number of bits for nFilter.

**nBitNFft** Number of bits for nFftr.

**nBitNBlk** Number of bits for n_block.

k Band index.

fftLength Length of VOFF Coefficient.

b Block index.

v VOFF coefficient index.

**nFilter[k]** Filter length per band for VOFF.

**nFft[k]** The length of the FFT for each band is expressed as a power of 2, where nFft[k] is the exponent. i.e. $2^{nFft[k]}$ = FFT length per band for VOFF.

**nBlk[k]** Number of blocks per band for VOFF.

**VoffCoeffLeftReal[k][b][nr][v]** Real values of left VOFF coefficients (interpreted as float in IEEE 754 format).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

**VoffCoeffLeftImag[k][b][nr][v]**
Imaginary values of left VOFF coefficients (interpreted as float in IEEE 754 format).

**VoffCoeffRightReal[k][b][nr][v]**
Real values of right VOFF coefficients (interpreted as float in IEEE 754 format).

**VoffCoeffRightImag[k][b][nr][v]**
Imaginary values of right VOFF coefficients (interpreted as float in IEEE 754 format).

**fcAna[k]** Center frequencies of the late reverberation analysis frequency bands (interpreted as float in IEEE 754 format).

**RT60[k]** Reverberation times RT60 in seconds of the late reverberation in the late reverberation analysis bands (interpreted as float in IEEE 754 format).

**NRG[k]** Energy values that represent the energy (amplitude to the power of two) of the late reverberation part of one BRIR in the late reverberation analysis bands (interpreted as float in IEEE 754 format).

**nBitQtdlLag[k]** Number of bits used for convolution.

**QtdlGainLeftReal[k][nr]** Real values of left QTDL gains(interpreted as float in IEEE 754 format).

**QtdlGainLeftImag[k][nr]** Imaginary values of left QTDL gains(interpreted as float in IEEE 754 format).

**QtdlGainRightReal[k][nr]** Real values of right QTDL gains(interpreted as float in IEEE 754 format).

**QtdlGainRightImag[k][nr]** Imagniary values of right QTDL gains(interpreted as float in IEEE 754 format).

**QtdlLagLeft[k][nr][l]** Left lag values in samples for QTDL.

**QtdlLagRight[k][nr][l]** Right lag values in samples for QTDL.

chan index of channel.

block index of diffuse block.

tap index of filter tap.

**bsDelay** delay in sample to apply at the beginning of output signals (to compensate BRIRs propagation time removed at parametrization).

**bsDirectLen** size in samples of the direct part of parameterized BRIRs.

**bsNbDiffuseBlocks** number of blocks in diffuse part of parameterized BRIRs.

**bsFmaxDirectLeft** cut off frequency of left direct part, given by a number between 0 and 1, where 1 corresponds to the Nyquist frequency (interpreted as float in IEEE 754 format).

**bsFmaxDirectRight** cut off frequency of right direct part, given by a number between 0 and 1, where 1 corresponds to the Nyquist frequency (interpreted as float in IEEE 754 format).

**bsFmaxDiffuseLeft** cut off frequency of left diffuse part, given by a number between 0 and 1, where 1 corresponds to the Nyquist frequency (interpreted as float in IEEE 754 format).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**bsFmaxDiffuseRight** cut off frequency of right diffuse part, given by a number between 0 and 1, where 1 corresponds to the Nyquist frequency (interpreted as float in IEEE 754 format).

**bsWeights** gain to apply to the input channels before filtering with diffuse part of the impulse response (interpreted as float in IEEE 754 format).

**bsFIRDirectLeft** FIR coefficients of direct part of left parameterized BRIRs (interpreted as float in IEEE 754 format).

**bsFIRDirectRight** FIR coefficients of direct part of right parameterized BRIRs (interpreted as float in IEEE 754 format).

**bsFIRDiffuseLeft** FIR coefficients of diffuse part of left parameterized BRIRs (interpreted as float in IEEE 754 format).

**bsFIRDiffuseRight** FIR coefficients of diffuse part of right parameterized BRIRs (interpreted as float in IEEE 754 format).

## 17.5 Interface for local screen size information

### 17.5.1 General

In both rendering cases (binaural rendering or loudspeaker rendering) local screen size information can be signaled in the mpegh3daLocalSetupInformation() syntax element. This data shall be used for screen-related element remapping and zooming in the decoder if screen-related elements are present in the audio scene.

### 17.5.2 Syntax

**Table 166 — Syntax of LocalScreenSizeInformation()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| LocalScreenSizeInformation() | | |
| { | | |
| **isCenteredInAzimuth;** | **1** | **bslbf** |
| if ( isCenteredInAzimuth ) { | | |
| **bsLocalScreenSizeAz;** | **9** | **uimsbf** |
| } else { | | |
| **bsLocalScreenSizeLeftAz;** | **10** | **uimsbf** |
| **bsLocalScreenSizeRightAz;** | **10** | **uimsbf** |
| } | | |
| **hasLocalScreenElevationInformation;** | **1** | **bslbf** |
| if ( hasLocalScreenElevationInformation ) { | | |
| **bsLocalScreenSizeTopEl;** | **9** | **uimsbf** |
| **bsLocalScreenSizeBottomEl;** | **9** | **uimsbf** |
| } | | |
| } | | |

### 17.5.3 Semantics

**isCenteredInAzimuth** This flag defines if the screen is centered in azimuthal direction with respect to the listening position (sweet spot).

**bsLocalScreenSizeAz** This field defines the azimuth corresponding to the left and right screen edge if the screen is centered in azimuth:

$\varphi_{left}$ = 0.5 · bsLocalScreenSizeAz

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

$\varphi_{left}$ = min (max ( $\varphi_{left}$ , 0), 180);

$\varphi_{right}$ = -0.5 · bsLocalScreenSizeAz

$\varphi_{right}$ = min (max ( $\varphi_{right}$ , -180), 0);

**bsLocalScreenSizeLeftAz** This field defines the azimuth corresponding to the left screen edge if the screen is not centered in azimuth:

$\varphi_{left}$ = 0.5 · (bsLocalScreenSizeLeftAz – 511)

$\varphi_{left}$ = min (max ( $\varphi_{left}$ , -180), 180);

**bsLocalScreenSizeRightAz** This field defines the azimuth corresponding to the right screen edge if the screen is not centered in azimuth:

$\varphi_{right}$ = 0.5 · (bsLocalScreenSizeRightAz – 511)

$\varphi_{right}$ = min (max ( $\varphi_{right}$ , -180), 180);

**hasLocalScreenElevationInformation**
This flag defines if elevation information is available.

**bsLocalScreenSizeTopEl** This field defines the elevation corresponding to the top screen edge:

$\theta_{top}$ = 0.5 · (bsLocalScreenSizeTopEl – 255)

$\theta_{top}$ = min (max ( $\theta_{top}$ , -90), 90);

**bsLocalScreenSizeBottomEl** This field defines the elevation corresponding to the bottom screen edge:

$\theta_{bottom}$ = 0.5 · (bsLocalScreenSizeBottomEl – 255)

$\theta_{bottom}$ = min (max ( $\theta_{bottom}$ , -90), 90);

## 17.6 Interface for signaling of local zoom area

### 17.6.1 General

A local zoom area can be signaled to the decoder by the LocalZoomAreaSize() interface.

### 17.6.2 Syntax

**Table 167 — Syntax of LocalZoomAreaSize()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| LocalZoomAreaSize() | | |
| { | | |
| **bsZoomAzCenter;** | **10** | **tcimsbf** |
| **bsZoomAz;** | **10** | **uimsbf** |
| **bsZoomElCenter;** | **9** | **tcimsbf** |
| **bsZoomEl;** | **10** | **uimsbf** |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

### 17.6.3 Semantics

**bsZoomAzCenter** This field defines the azimuth corresponding to the center position of left and right edges of a zoom area in azimuth. This field can take integer values between -360 and +360.

$$\varphi_{center}^{ZoomArea} = 0.5 \times \text{bsZoomAzCenter}$$

$$\varphi_{center}^{ZoomArea} = \min(\max(\varphi_{center}^{ZoomArea}, -180), +180)$$

**bsZoomAz** This field defines the azimuth corresponding to the offset from the center position to left and right edges of a zoom area in azimuth. This field can take integer values between 0 and +720.

$$\varphi_{offset}^{ZoomArea} = 0.125 \times \text{bsZoomAz}$$

$$\varphi_{offset}^{ZoomArea} = \min(\max(\varphi_{offset}^{ZoomArea}, 0), 90)$$

**bsZoomElCenter** This field defines the elevation corresponding to the center position of top and bottom edges of a zoom area in elevation. This field can take integer values between -180 and +180.

$$\theta_{center}^{ZoomArea} = 0.5 \times \text{bsZoomElCenter}$$

$$\theta_{center}^{ZoomArea} = \min(\max(\theta_{center}^{ZoomArea}, -90), +90)$$

**bsZoomEl** This field defines the elevation corresponding to the offset from the center position to top and bottom edges of a zoom area in elevation. This field can take integer values between 0 and +720.

$$\theta_{offset}^{ZoomArea} = 0.125 \times \text{bsZoomEl}$$

$$\theta_{offset}^{ZoomArea} = \min(\max(\theta_{offset}^{ZoomArea}, 0), 90)$$

## 17.7 Interface for user interaction

### 17.7.1 Introduction

This subclause describes a syntax for a normative interface for user interactivity or communication with the application.

### 17.7.2 Definition of User Interaction Categories

Different user interaction categories are defined according to the metadata audio element syntax. All modifications are defined on the level of a group of elements, because groups gather related elements that shall only be manipulated jointly. The possible modification is defined according to the Metadata Audio Element fields

— **On/Off interactivity**: A group of elements is switched on or off (playback is enabled or disabled).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

— **Position interactivity**: The position in 3D space of all elements of a group is changed (azimuth, elevation and distance are modified). The interaction happens by offset values (azimuth, elevation) or a multiplication factor (distance) to allow for an unchanged relative inter-element relationship.

— **Gain interactivity**: The level/gain of all elements of a group is changed. The interaction happens by an additional decibel value that is added to all elements of a group, such that the relative inter-element relationship is kept unchanged.

— **WIRE interactivity**: The audio content of the elements of a group are routed to a WIRE output (e.g. content for hearing impaired or an additional language track, a WIRE output is a generic output in addition to the connected standard output device (loudspeakers or headphones)). No metadata is sent along the unrendered audio content.

### 17.7.3 Definition of an Interface for User Interaction

The mpegh3daElementInteraction() syntax element provides an interface for all allowed kinds of user interaction. Two interaction modes are defined in the interface.

The first mode is an advanced interaction mode, where the interaction can be signaled for each element group that is present in the audio scene. This mode enables the user to freely choose which groups to play back and to interact with all of them (within the restrictions of allowances and ranges defined in the metadata and the restrictions of switch group definitions).

The second mode is a basic interaction mode, where the user may choose one of the group presets that are defined in the metadata audio element syntax. With a group preset, the on/off statuses of the groups that are referenced in the conditions of the chosen preset are defined and cannot be changed by the user. The user may only change the on/off status of the other groups that are not referenced in the conditions of the chosen preset. Position and gain of all groups can be changed according to the defined restrictions and ranges.

A signature is introduced in the interface. The electronic signature can be used to identify and authenticate a particular person, device or software as the origin of the user interaction data.

### 17.7.4 Syntax of interaction interface

**Table 168 — Syntax of mpegh3daElementInteraction()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| mpegh3daElementInteraction() | | |
| { | | |
| **ei_InteractionSignatureDataLength;** | **8** | **uimsbf** |
| if (ei_InteractionSignatureDataLength >0) { | | |
| **ei_InteractionSignatureDataType;** | **8** | **uimsbf** |
| for ( c = 0; c < ei_InteractionSignatureDataLength ; c++ ) { | | |
| **ei_InteractionSignatureData**[c]; | **8** | **uimsbf** |
| } | | |
| } | | |
| ElementInteractionData(); | | |
| **hasLocalZoomAreaSize**; | **1** | **bslbf** |
| if (hasLocalZoomAreaSize) { | | |
| LocalZoomAreaSize(); | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**Table 169 — Syntax of ElementInteractionData()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| ElementInteractionData() | | |
| { | | |
| **ei_interactionMode;** | **1** | **bslbf** |
| **ei_numGroups;** /* Channel, Object, HOA, SAOC */ | **7** | **uimsbf** |
| if ( ei_interactionMode == 0) { | | |
| ei_GroupInteractivityStatus( ei_numGroups ); | | |
| } else { | | |
| **ei_groupPresetID;** | **5** | **uimsbf** |
| ei_GroupInteractivityStatus( ei_numGroups ); | | |
| } | | |
| } | | |

**Table 170 — Syntax of ei_GroupInteractivityStatus()**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| ei_GroupInteractivityStatus ( numGroups ) | | |
| { | | |
| for (grp = 0; grp < numGroups; grp++) { | | |
| **ei_groupID**[grp]; | **7** | **uimsbf** |
| **ei_onOff**[grp]; | **1** | **bslbf** |
| **ei_routeToWIRE**[grp]; | **1** | **bslbf** |
| if ( ei_routeToWIRE[grp] == 1 ) { | | |
| **routeToWireID**[grp]; | **16** | **uimsbf** |
| } | | |
| if ( ei_onOff [grp] == 1) { | | |
| **ei_changePosition**[grp]; /* position change */ | **1** | **bslbf** |
| if ( ei_changePosition[grp] ) { | | |
| **ei_azOffset**[grp]; | **8** | **uimsbf** |
| **ei_elOffset**[grp]; | **6** | **uimsbf** |
| **ei_distFact**[grp]; | **4** | **uimsbf** |
| } | | |
| **ei_changeGain;** /* gain change */ | **1** | **bslbf** |
| if ( ei_changeGain ) { | | |
| **ei_gain**; | **7** | **uimsbf** |
| } | | |
| } | | |
| } | | |
| } | | |

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

### 17.7.5 Semantics of interaction interface

**ei_InteractionSignatureDataLength** This field defines the length of the following interaction signature in Byte.

**ei_InteractionSignatureDataType** This field defines the type of signature. The following values are possible:


© ISO/IEC 2015 – All rights reserved

**Table 171 — Value of ei_InteractionSignatureDataType**

| value | meaning |
|---|---|
| 0 | Generic String in UTF-8 according to ISO/IEC 10646 |
| 1-127 | Reserved for ISO use. See NOTE |
| 128-255 | Reserved for use outside of ISO scope |
| NOTE It is expected that a registration authority will maintain a list of values of the ei_InteractionSignatureDataType and its meaning. | |

**ei_InteractionSignatureData** This field contains a signature defining the originator of the interaction data.

**hasLocalZoomAreaSize** Flag that defines if information about the local zoom area size is available. If this flag is enabled, Object Remapping for Zooming is applied according to 18.4.

**ei_interactionMode** Flag that defines if the advanced interaction type or the basic interaction mode is chosen.
A value of 0 indicates the advanced interaction mode.
A value of 1 indicates the basic interaction mode.

**ei_numGroups** This field contains the number of groups in the audio scene.

**ei_groupPresetID** This field contains a mae_groupPresetID that is defined in the audio scene. This ID reflects the user's preset choice.

**ei_groupID** mae_groupID for the current group for which the interaction is described.

**ei_routeToWIRE** This field defines if the audio content of the group should be routed to a WIRE output.

**ei_routeToWireID** ID of the WIRE output where the group should be routed to.

**ei_onOff** Defines the on/off status of the current group. In case the basic interaction mode (interaction on group presets) is chosen, this value has to be identical to the defined on/off status of the group with ei_groupID if this group is part of the conditions of the chosen group preset with ei_groupPresetID. For basic interaction mode it is not allowed to signal a different on/off status here. For all groups that are not part of the conditions of the chosen group preset, the on/off status may arbitrarily be signaled.

**ei_changePosition** This flag defines if the position of the group elements has been changed (azimuth and/or elevation and/or distance have been modified by the user).

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

**ei_azOffset**

The change of azimuth is given as an offset. This field can take values between AzOffset=-180° and AzOffset=180°:

AzOffset = $1.5 \times$ (ei_azOffset – 128)

AzOffset = min (max (AzOffset, -180), 180);

**ei_elOffset**

The change of azimuth is given as an offset. This field can take values between ElOffset=-90° and ElOffset=90°:

ElOffset = $3 \times$ (ei_elOffset – 32)

ElOffset = min (max (ElOffset, -90), 90);

**ei_distFact**

The distance interactivity is given as a multiplication factor. The field can take values between 0 to 15 resulting in DistFactor between 0.00025 and 8:

DistFactor = $2^{(ei_distFactor-12)}$

DistFactor = min( max (DistFactor, 0.00025), 8);

**ei_changeGain**

This flag defines if the gain/level of the group elements has been changed (the gain has been modified by the user)

**ei_gain**

This field defines an additional gain of the members of the current group. The field can take values between 0 and 127 representing gain values between
Gain = -63 dB and Gain = 31 dB in 1 dB steps, with

Gain [dB] = ei_gain – 64

Gain [dB] = min( max (Gain, -63), 31);

If ei_gain is set to 0, Gain shall be set to minus infinity dB.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

### 17.8 Interface for loudness normalization and dynamic range control (DRC)

The interface for loudness normalization and dynamic range control is fully specified in ISO/IEC 23003-4.

## 18 Application and processing of local setup information and interaction data

### 18.1 *Element Metadata* Preprocessing

A processing block is defined that processes the user interaction interface data and prepares the audio elements for rendering and play-out. This “Element Metadata Preprocessor” also applies the logic that is imposed by the user interaction and the element metadata.


© ISO/IEC 2015 – All rights reserved

The following processing structure and order of processing the interface data (interactivity interface mpegh3daElementInteraction() and interface for local setup information mpegh3daLocalSetupInformation()) and metadata audio elements is defined. An overview is given in Figure 70.

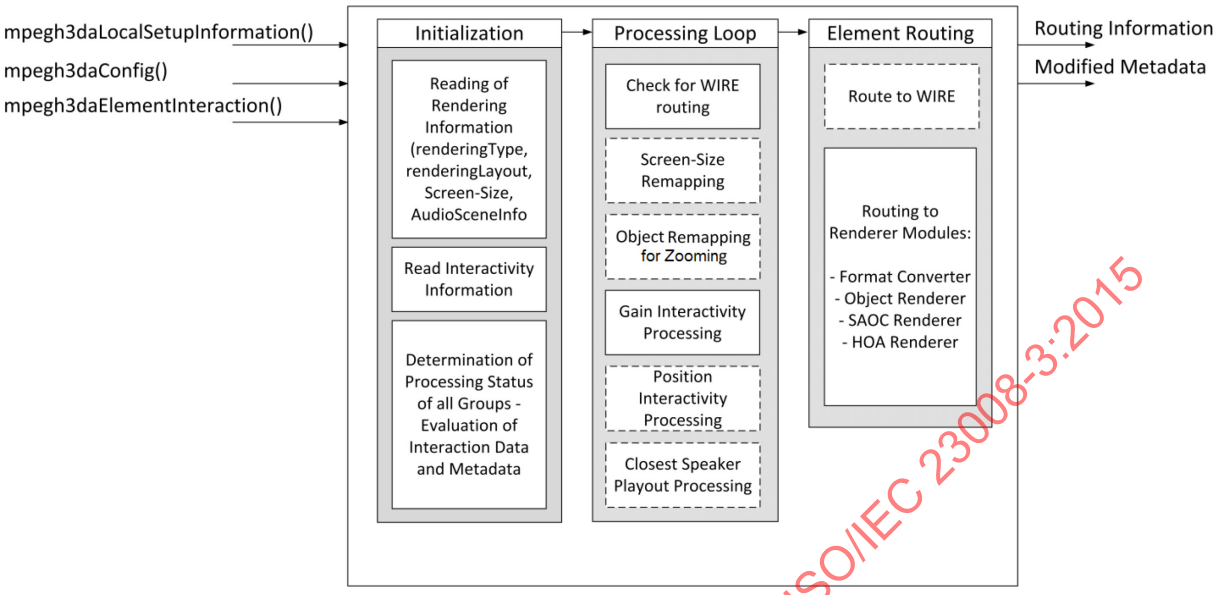



**Figure 70 — Processing of Metadata and Interface Data**

As a first step the rendering type, rendering layout and screen size information is determined. Then, the syntax element mae_AudioSceneInfo() is retrieved. Afterwards, the interactivity data is read from the interactivity interface.

The interaction type (basic or advanced) is determined. Next, it is determined which element groups have to be processed. This is defined by their on/off status, the WIRE routing information and the implicit logic by switch group definitions.

In the basic interaction mode, the user can choose one of the defined group presets. The on/off statuses of the groups that are referenced in the chosen group preset's conditions are set according to these conditions. Any discrepancy to these values by the values in the ei_GroupInteractivityStatus() syntax element shall be ignored.

For the groups that are not referenced in the chosen group preset's conditions, the status is set to the mae_defaultOnOff value as a default, but the user can choose the on/off status to his preference if the definition of the group allows for it and no switch group logic is violated.

The following groups have to be processed:

— All groups that are switched on and are not member of a switch group.

— All groups that are switched on and that are member of a switch group if the switch group condition is fulfilled (a maximum of one member of this switch group is switched on).

— All groups that are switched off but are default members of a switch group from whose members no other member is marked as switched on. The on/off status of this default group shall then be set to 1 (on).

— All groups that are switched off and are marked to be routed to a WIRE output.

NOTE The relevant groups are known at this point in time so the core decoder could be controlled such that irrelevant elements are not decoded.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved

Then the relevant groups are prepared for playout and rendering. Therefore, updated output element characteristics are calculated for each element of each group: Gain, azimuth, elevation, and distance. Azimuth and elevation either represent the element's position or the element's speaker's position (in case the mae_closestSpeakerPlayout flag is set to one). In case of groups with SignalGroupTypeChannels, azimuth and elevation contain the unmodified speaker's position.

For the determination of the updated element characteristics, it shall first be checked if the element group are marked to be routed to a WIRE output.

If a group is not routed to a WIRE output (non-WIRE group), the Screen-Related Remapping is conducted for all group members that are marked as screen-related. If a group is marked as to be routed to a WIRE output (WIRE group), this step should be skipped.

Afterwards, the gain interactivity data is processed (for non-WIRE groups and WIRE groups). Therefore, it is first checked if the current group allows for gain interactivity (mae_allowGainInteractivity flag is equal to 1).

If that is the case and if the basic interaction mode is active, it is then checked if the currently active/valid preset disables the gain interactivity.

If the gain interactivity is still enabled, the output gain is calculated for each element of the group, taking into account the interactivity gain modification, and either the OAM gain (if available) or the current gain of an element (in case no OAM data is available). If the basic interaction mode is enabled, the possible group gain value of the currently active/valid preset is also taken into account.

The overall gain modification (interactivity gain in dB plus group gain in dB) shall be restricted according to the given mae_interactivityMinGain and mae_interactivityMaxGain values. If no gain interaction is allowed, the output gain shall either contain 0dB (in case no OAM data is available) or the OAM gain in dB.

If the group is marked as WIRE output, then no further interactivity processing shall be applied.

If the group is not marked as WIRE output, then the application of further processing depends on the signal type. If the group contains channel-based elements (SignalGroupTypeChannels), then no further interactivity processing shall be applied. Instead the position of the predefined associated speaker shall be determined from the audioChannelLayout information for each element of the group. It is assumed that the member elements in the member list are ordered with respect to the channel ordering signaled in the syntax element Signals3d().

If the group is accompanied by OAM data, then the position interactivity shall be processed. Therefore, it is first checked if the current group allows for position interactivity (mae_allowPositionInteractivity flag is equal to1).

If that is the case and if basic interaction mode is active, it is then checked if the currently active/valid preset disables the position interactivity.

If the position interactivity is still enabled for the current group, azimuth, elevation and distance values are determined for each element of the group. The calculation of the output values takes into account the interactivity modification (offset values and distance multiplication factor) and the values from the OAM data. If the basic interaction mode is enabled, the possible values for a group azimuth offset, group elevation offset and group distance factor defined in the preset shall be taken into account.

The overall interactivity modification (interactivity modification plus the corresponding values in the preset) shall be restricted according to the given interactivity ranges.

As a next step, the closest-speaker playout processing shall be conducted. If the group is marked as closest-speaker playout, then the position of the closest speaker to the output position data is determined for each member element as defined in 18.5. No rendering shall be applied; therefore it has to be ensured that the determined speaker position exists in the reproduction speaker setup.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015


© ISO/IEC 2015 – All rights reserved

After processing the interactivity data, the routing information for the elements is determined. If an element group is marked as WIRE output, it should be directly sent to the according WIRE ID. The output gain is sent along and shall be applied before WIRE output. Metadata regarding gain interactivity as well as Loudness/DRC processing should be applied before outputting the audio signal. Content that is routed to a WIRE output should be processed with the same Loudness/DRC processing as if they would be part of the regular output signals. The application of a peak limiter at the end of the processing chain is highly recommended. The signals on the WIRE output should have the same delay as signals after the mixer.

If a group is not marked as WIRE output, then the output depends on the signal type.

Channel-based groups are marked to be sent to the format converter, object-based groups are marked to be sent to the object renderer, SAOC-based groups are marked to be sent to the SAOC renderer, HOA-based groups are marked to be sent to the HOA renderer.

In addition to the decoded audio data, each renderer gets azimuth, elevation, distance (either element's position or element's speaker's position), gain and the rendering layout. The SAOC renderer and the HOA renderer might need additional side information.

The application of the output element gains shall happen before playout.

## 18.2 Interactivity Limitations and Restrictions

### 18.2.1 General Information

The application of user interactivity is restricted by the metadata audio element syntax that is defined for the groups of elements. There, the interaction possibilities may be limited for each group by the metadata fields mae_allowOnOff, mae_allowGainInteractivity and mae_allowPositionInteractivity.

In addition the interaction possibilities are restricted by the logic of groups and switch groups.

### 18.2.2 WIRE Interactivity

WIRE interactivity is only defined if the local reproduction setup and the setup of connected WIRE outputs (including WIRE IDs) are known at the user interface or application. A WIRE output can be a device that supports a single channel or a multichannel signal.

WIRE output are only allowed for groups of type SignalGroupTypeChannels or SignalGroupTypeObject.

The decoder output is a single channel or multichannel signal that is sent to the WIRE output. The number of channels corresponds to the number of members of the element group that is routed to a WIRE output. If multiple groups are routed to the same WIRE output, then the number of channels corresponds to the total number of elements in these groups.

The channel-order corresponds to the order of elements in the group's member list. If multiple groups are routed to the same WIRE output the order of channels is defined with ascending mae_groupID and within each group by the order of elements in the group's member list.

If the number of elements that are routed to a WIRE output does not correspond to the number of channels that the WIRE output could handle (e.g. sending a channel-based element group with stereo dialogue to a WIRE headset with just one output channel), the application has to handle this discrepancy.

As also switched-off groups can be routed to a WIRE output (e.g. for support of alternative languages or voice-over content over WIRE), the audio decoder shall allow output of switched-off groups which are routed to WIRE outputs.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23008-3:2015

© ISO/IEC 2015 – All rights reserved